Dynamic Edge-centric Resource Provisioning for Online and Offline Services Co-location

Tao Ouyang, Kongyange Zhao, Xiaoxi Zhang, Zhi Zhou, Xu Chen
School of Computer Science and Engineering, Sun Yat-sen University, China
Email: {ouyt9, zhaokyg}@mail2.sysu.edu.cn, {zhangxx89, zhouzhi9, chenxu35}@mail.sysu.edu.cn

***Abstract*—Due to the penetration of edge computing, a wide variety of workloads are sunk down to the network edge to alleviate huge pressure of the cloud. With the presence of high input workload dynamics and intensive edge resource contention, it is highly non-trivial for an edge proxy to optimize the scheduling of heterogeneous services with diverse QoS requirements. In general, online services should be quickly completed in a quite stable running environment to meet their tight latency constraint, while offline services can be processed in a loose manner for their elastic soft deadlines. To well coordinate such services at the resource-limited edge cluster, in this paper, we study an edge-centric resource provisioning optimization for dynamic online and offline services co-location, where the proxy seeks to maximize timely online service performances while maintaining satisfactory long-term offline service performances. However, intricate hybrid couplings for provisioning decisions arise due to heterogeneous constraints of the co-located services and their different time-scale performances. We hence first propose a reactive provisioning approach without requiring a prior knowledge of future system dynamics, which leverages a Lagrange relaxation for devising constraint-aware stochastic subgradient algorithm to deal with the challenge of hybrid couplings. To further boost the performance by integrating the powerful machine learning techniques, we also advocate a predictive provisioning approach, where the future request arrivals can be estimated accurately. With rigorous theoretical analysis and extensive trace-driven evaluations, we show the superior performance of our proposed algorithms for online and offline services co-location at the edge.**

I. Introduction

Edge computing has benefited a broad range of delay-sensitive applications such as auto-driving, online education, video analytics. In general, these applications are more likely to be interactive so that the workloads highly depend on the human activities, which further exacerbate the resource inefficiency issues [1]. Meanwhile, with massive user data generated at the edge of network, the edge storage paradigm becomes popular, which enables a cost-effective and privacy-friendly processing for data-intensive tasks at the edge, such as DNN model training for federated learning [2]. However, such huge increases in user demands dramatically promote diversification and temporal imbalance of edge workloads [3]. Thus, the proxy faces the grand challenge of efficiently scheduling a wide variety of heterogeneous workloads with diverse QoS requirements within a resource-limited edge cluster.

Co-locating online and offline services has been regarded as an economic solution to boost both the quality of service and resource utilization in the modern cloud data centers [4]. Particularly, compared with prior online services, offline batch jobs are usually regarded as secondary background tasks to improve resource efficiencies whenever server capacity is underutilized. As a complement to cloud computing, the edge platforms can also follow this running pattern to optimize the system efficiency. Nevertheless, a major difference between cloud and edge is the resource bottleneck, i.e., the edge cluster typically hosts much fewer servers as constrained by the physical infrastructure size (e.g. Alibaba ENS [1]). Thus, rather than primarily targeting on the system resource utilization improvement, the joint optimization of online and offline service performances is more desirable due to edge resource capacity constraint. Consequently, a thorny problem arises: *How can the edge proxy properly allocate the limited edge resources for the co-located online and offline services to jointly boost their heterogeneous QoS performances?*

To address this challenging issue, an elastic management for edge resource provisioning is vital. More explicitly, the edge proxy allows the number of VM instances or containers to be scaled up or down with time-varying demands to improve the service performances. However, regardless of operation cost such as extra wear and tear on the hardware, frequent provisioning adjustment would considerably deteriorate the real-time performance of the delay-sensitive online edge services. For example, the startup of a VM instance or container should load the necessary codes for running (e.g. language runtime libraries), which causes extra long time relative to the service execution. Indeed, an important observation in [5] demonstrates that the function execution times are of the same order of magnitude as the cold start times. As for offline services, such as data-intensive cluster applications (e.g. model training, data analytics [6]), they are far more delay-tolerant than online services, and the running time can even reach to an order of magnitude of hours or days. In this way, the edge proxy can leverage such characteristics to enable efficient opportunistic resource sharing under a regime of online and offline edge services co-location.

However, optimizing such provisioning decisions adaptively over the run time is non-trivial. On the one hand, serious tensions among different time-scale performances exist under highly dynamic workloads and intensive resource contention at the edge clusters. Thus, how to well navigate such an inherent performance trade-off is a huge challenge for long-term performance optimization. On the other hand, the stringent requirements of stochastically-arriving online services impose varying restrictions on their timely resource provisioning, which in turn induces intricate couplings between online and

This work was supported in part by the National Science Foundation of China (No. U20A20159, No. 61972432); Guangdong Basic and Applied Basic Research Foundation (No.2021B151520008); the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.2017ZT07X355). Corresponding Author: Xu Chen.

offline services for sharing limited edge resources. To tackle such challenging hybrid couplings, we develop an efficient dynamic online optimization framework of constraint-aware stochastic subgradient algorithms based on the key principle of Lagrange relaxation. We elaborate algorithm designs under the settings of reactive and predictive resource provisioning.The main contributions of this work are listed as follows:

- Based on the characteristics of online and offline edge services, we formulate a novel dynamic edge-centric resource provisioning optimization, where the proxy tries to maximize timely online service performances while maintaining satisfactory long-term offline service performances with the limited edge resources.
- We leverage the idea of Lagrange relaxation, and develop a constraint-aware stochastic subgradient algorithm based reactive provisioning approach without requiring a prior knowledge of future system dynamics. Based on some useful insights that capture the characterizations of heterogeneous resource provisioning constraints, we further quantify the performance gap between our online algorithm and the offline optimal solution.
- We further extend our reactive provisioning approach for an enhanced design, which allows the proxy to integrate the powerful machine learning mechanisms for predictive resource provisioning. Rigorous theoretical performance analysis is provided to verify its effectiveness.
- The trace-driven simulation demonstrates the efficiency of the proposed algorithms in both reactive and predictive settings, e.g. achieving 13.5% performance gains in term of timely online service completion rate than the dynamic resource partition algorithm in literature under the given average offline service throughput requirement.

II. Related work

Cluster Scheduling for Heterogeneous Services: To efficiently deal with heterogeneous workloads, online services need to be quickly processed while the execution of offline services can be deferred based on their requirements. On this basis, a hybrid datacenter scheduling [6] is proposed to achieve low delay for short tasks while maintaining high resource utilization by an efficient resource partition for online and offline services. The work [4] runs offline services as background tasks to improve the resource utilization, and proposes a container-based preemption approach to achieve both responsiveness and high utilization. An elastic GPU cluster scheduler for deep learning is proposed in [7], which exploits cluster-level capacity loading and job-level elasticity to tackle the problems of low inference utilization and long training queuing time. A new YARN scheduler based on task-dependency and resource-demand in [8] is proposed to improve resource utilization and reduce the makespan. Major works of cluster scheduling mainly focus on the task preemptions or priority of online services, while the negative impact of provisioning changing are less considered.

Switching Costs for Online Services: A proactive solution called fixed keep-alive policy is adopted in AWS and Azure

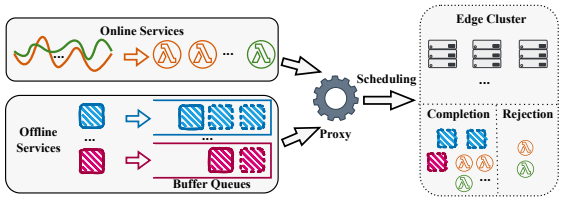

Fig. 1. An edge platform for online and offline service co-location.

(e.g. OpenWhisk) to reduce cold start at the expense of over-provisioning. Based on the workload characterization, a dynamic hybrid histogram policy is proposed in [9], which optimizes the pre-warming window and keep-alive window to alleviate the cold starts with fewer resources. A different line of works incorporates such negative impacts as additional constant costs [10] or switching costs [11], [12] into the objective function (e.g. perceived delay or operational cost). Compared with the above works, we consider a more flexible and practical scheduling form, i.e., stable service runtime constraints for online services by limiting new container initiation within a given satisfaction range.

Resource Provisioning at the Edge: Major works focus on resource provisioning for delay-sensitive applications (i.e., online services) [11] [13] [14] [15], which aim to minimize the overall latency with low operation costs or high resource utilization. Few works consider both online and offline services co-location at the edge. In [16], a delay-aware Lyapunov based algorithm is proposed to minimize the long-term operational cost, while ensuring diverse QoS for heterogeneous IoT applications. By considering the cloud-assisted scenarios in [17], an efficient provisioning algorithm is proposed by leveraging the piece-wise convex property to minimize system cost with delay-tolerant and delay-sensitive constraints. However, the long-run impacts due to resource provisioning switching on online services in the co-location are not modeled. Thus, to well characterize the key features of online and offline services in co-location, a full-fledged edge-centric resource provisioning modeling and optimization framework is essential.

III. System Model and problem formulation

As illustrated in Fig. 1, we consider M types of online services and N types of offline services at an edge cluster, denoted as the set $\mathcal{M}$ and $\mathcal{N}$, respectively. To enable elastic service management, we assume that the edge resources are virtualized into a set $\mathcal{J}$ of containers or lightweight VMs to provision the needed runtime environments for different applications. Practical container configuration usually contains a small set of resource configuration, such as CPU and memory, but we adopt one constant c_j to denote the occupied amount of computing resources for configuration $j \in \mathcal{J}$ for simplicity. Note that other dimensional resources (e.g. memory) can be set in proportion to unit resource c_j, such as AWS Lambda allocates CPU power with linear to the amount of memory configured [10]. In terms of a container, better service performance (e.g., throughput) can be achieved with larger unit resource. Besides, the mutual interference among multiple services is not considered in our model, since

it can be largely alleviated by existing techniques such as resource isolation [18] [19]. Similar to [16] [17], we adopt a discrete time-slotted model, denoted as $\mathcal{T} = \{0, 1, 2, \ldots, T\}$, to characterize the system dynamics, and each slot matches the time scale where resource provisioning decisions are updated.

A. Edge Resource Provisioning Decisions

The proxy should carefully choose configuration combinations to balance the trade-off among heterogeneous service performances for efficient resource provisioning. For instance, for DNN based inference tasks, the performance highly depends on the assigned resources and DNN model type within a container configuration j. Thus, we denote a resource provisioning vector for each online service $m \in \mathcal{M}$ as $\boldsymbol{x}_m^t = (x_{m,1}^t, \ldots, x_{m,J}^t)$, where each entry $x_{m,j}^t$ is the amount of container configuration $j \in \mathcal{J}$ assigned to the service m. Further, we use stacked vector $\boldsymbol{x}^t = (\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_M^t)$ to denote the total decisions for the online services $\mathcal{M}$ at slot t. The resource provisioning vector $\boldsymbol{x}^t$ satisfies:

$$\sum_{j \in \mathcal{J}} [x_{m,j}^t - x_{m,j}^{t-1}]^+ \leq \Delta_m, m \in \mathcal{M}, t \in \mathcal{T}, \tag{1}$$

$$0 \leq x_{m,j}^t \leq X_{m,j}^{max}, m \in \mathcal{M}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{2}$$

where $[x]^+ = \max\{0, x\}$ and Δ_m is a presumed constant on the magnitude of the resource provisioning change for the online service m. The first constraint (1) is a stable service runtime constraint, which reflects the latency sensitivity for online services. By setting a moderate value of Δ_m to control the spurt in resource provisioning, the negative impact of frequent time-consuming cold-start operations on the delay-sensitive online services (e.g. launching many new containers on spot) can be largely alleviated[1]. For simplicity, we denote $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_M)$ as the total control parameters for all online services. The second constraint (2) is a quota constraint, which ensures that at most $X_{m,j}^{max}$ amount of the service m within resource configuration j can be assigned. Likewise, for each offline service $n \in \mathcal{N}$, the resource provisioning vector is denoted as $\boldsymbol{y}_n^t = (y_{n,1}^t, \ldots, y_{n,J}^t)$, and $\boldsymbol{y}^t = (\boldsymbol{y}_1^t, \ldots, \boldsymbol{y}_N^t)$. And its quota constraint must satisfy:

$$0 \leq y_{n,j}^t \leq Y_{n,j}^{max}, n \in \mathcal{N}, j \in \mathcal{J}, t \in \mathcal{T}. \tag{3}$$

Besides, the total provisioning resource for both decisions $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ should not exceed the maximum edge capacity C, i.e.,

$$\sum_{j \in \mathcal{J}} c_j \Big(\sum_{m \in \mathcal{M}} x_{m,j}^t + \sum_{n \in \mathcal{N}} y_{n,j}^t \Big) \leq C, \forall t \in \mathcal{T}. \tag{4}$$

Note that in our edge-centric resource provisioning, offline services will not be offloaded to the remote public cloud for process, due to the privacy issues (e.g. data leakage) and operational cost (e.g. huge expense of bandwidth resources caused by cloud data transmission) [2]. While, the edge cluster can process cloud offline services for utilization improvements. Thus, we assume the proxy can preset higher quotas for offline services in general, i.e., the maximum quota of offline services holds $\sum_{j \in \mathcal{J}} \sum_{n \in \mathcal{N}} c_j Y_{n,j}^{max} \geq C$. Besides, we do not impose the switching constraints such as (1) for offline service provisioning, due to their delay-tolerant nature. For ease of exposition, the feasible domain of provisioning vector $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ are denoted as $\mathcal{X}^t$ and $\mathcal{Y}_0$[2].

B. Heterogeneous Service Performance Modeling

First, we introduce a random state vector $\boldsymbol{\sigma}^t$ to characterize all sources of randomness at current slot t, which is a sequence of random vector with an unknown distribution over whole time horizon. Then, we respectively define the utility functions for online and offline services to distinguish their different time-scale performance in the slotted-structure model.

Due to latency-sensitive property, online services should be completed at the end of each time slot as many as possible. Formally, let the utility function $f_m(\boldsymbol{x}_m^t, \boldsymbol{\sigma}^t)$ be online service m within resource provisioning vector $\boldsymbol{x}_m^t$ over the system state $\boldsymbol{\sigma}^t$, and $f_m^t(\boldsymbol{x}_m^t)$ be realized of $f_m(\boldsymbol{x}_m^t, \boldsymbol{\sigma}^t)$ at slot t for simplicity. Taking video stream inference services as an instance, we regard the utility function $f_m^t(\boldsymbol{x}_m^t)$ as the percentage of timely completion rate for online service m:

$$f_m^t(\boldsymbol{x}_m^t) = \frac{100 H_m \times \min\{A_m^t, \sum_{j \in \mathcal{J}} a_{m,j}^t x_{m,j}^t\}}{A_m^t + \beta}, \tag{5}$$

where the dynamic workload arrivals A_m^t of service m, and unit processed amount[3] $a_{n,j}^t$ of configuration j depend on the system randomness $\boldsymbol{\sigma}^t$. Besides, the weight H_m denotes the priorty of online service m, and the small constant β avoids a zero denominator in utility function.

Similarly, the utility function of offline service n is denoted as $g_n(\boldsymbol{y}_n^t, \boldsymbol{\sigma}^t)$ with a realized value at each t denoted as $g_n^t(\boldsymbol{y}_n^t)$. For example, the utility function $g_n^t(\boldsymbol{y}_n^t)$ can be adopted as a linear function to characterize the processing throughput at slot t for offline service n, i.e.,

$$g_n^t(\boldsymbol{y}_n^t) = \sum_{j \in \mathcal{J}} p_{n,j} y_{n,j}^t + b_n^t, \tag{6}$$

where parameter $p_{n,j}$ is the averaged unit throughput of offline service n within configuration j, and b_n^t is a random noise due to edge resource adjustment dynamics. Note that (5) and (6) present a special case of f_m^t and g_n^t in edge computing services. For the general cases, we assume that the utility functions satisfy the following regularity conditions, which are widely used in existing studies [20]–[23].

Assumption 1. *Basic assumptions of utility functions*

- *Given the system state $\boldsymbol{\sigma}^t$, the utility functions $f_m^t(\boldsymbol{x}_m^t)$, $g_n^t(\boldsymbol{y}_n^t)$ are non-decreasing, differentiable, concave, bounded, and satisfy $f_m^t(\mathbf{0}) = 0$, $g_n^t(\mathbf{0}) = 0$;*
- *The partial derivative of utility function $f_m^t(\cdot)$ is bounded, i.e., $|\nabla f_m^t(\cdot)| \leq U_g, \forall t$, where $U_g > 0$. And an upper bound for any offline service exists, i.e., $g_n(\boldsymbol{y}, \boldsymbol{\sigma}) \leq U_y$;*
- *The mean ergodicity of the utility functions of online and offline services hold in the appropriate sense:*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} F^t(\boldsymbol{x}^t) \overset{as}{=} \lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[F^t(\boldsymbol{x}^t)] := \overline{F},$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} g_n^t(\boldsymbol{y}_n^t) \overset{as}{=} \lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[g_n^t(\boldsymbol{y}_n^t)] =: \overline{B}_n, \forall n,$$

[1]Particularly, Δ_m in (1) depends on its workload characteristic and average arrival rate, and the empirical value setting is out of scope. Besides, the constant Δ_m can be easily extended to a random variable Δ_m^t with lower and upper bounds, as well the system randomness $\boldsymbol{\sigma}^t$. And the following proposed algorithms and theoretical analysis still work.

[2]The feasible domain $\mathcal{X}^t$ is varying due to the switching constraint while the feasible domain $\mathcal{Y}_0$ is fixed.

[3]The averaged value of parameter $a_{n,j}^t$ can be measured beforehand based on the service workload and the configuration type.

where expectations are over the distribution of randomness $\boldsymbol{\sigma}^t$, and the possible randomness of the decisions.

C. Problem Formulation

At the co-located edge cluster, the proxy seeks to maximize the timely online service performance while maintaining a satisfactory long-term offline performance, i.e.,

$$\begin{aligned}\mathbf{P_0}: \quad F^* = &\max_{\boldsymbol{x}^t, \boldsymbol{y}^t} \lim_{T\to\infty} \frac{1}{T}\sum_{t\in\mathcal{T}} F^t(\boldsymbol{x}^t)\\ \text{s.t.} \quad &(1),(2),(3),(4),\\ &\lim_{T\to\infty}\frac{1}{T}\sum_{t\in\mathcal{T}} g_n^t(\boldsymbol{y}_n^t) \geq B_n, \forall n\in\mathcal{N}, \end{aligned} \tag{7}$$

where the total real-time performance of online services at slot t can be denoted as $F^t(\boldsymbol{x}^t) = \sum_{m\in\mathcal{M}} f_m^t(\boldsymbol{x}_m^t)$, and long-term constraints (7) connect the utility function and expected performance B_n for offline services based on their latency-tolerant properties. It implies the lowest acceptable time-averaged performance for offline service n. Note that the long-term performance constraint can be rewritten as the stability of dynamic buffer queue, which can be denoted as

$$Q_n^{t+1} = \max[Q_n^t - g_n^t(\boldsymbol{y}_n^t) + B_n^t, 0], \tag{8}$$

where the desirable served amount B_n^t of offline service n depends on the system randomness. If its buffer queue is stable, i.e., $\lim_{T\to\infty}\frac{1}{T}\sum_{t\in\mathcal{T}}\mathbb{E}\{Q_n^t\} < \infty$, the long-term performance constraint (7) is satisfied [24], where B_n is the expectation of the desirable served amount B_n^t. Similar to many studies for resource provisioning [16] [17], we formulate the continuous decision variables above. This is mainly due to the fact that fine-grained configurations such as lightweight containers, kernels, serverless functions are being widely adopted in edge computing. Hence, the number of virtualized resources at the edge can be large, and continuous resource provisioning decisions can serve as a good approximation in practice. Note that our framework can be also generalized to the case of discrete decisions by integrating the rounding techniques, e.g. [25]. To guarantee the feasibility of provisioning policy within limited edge resources, the value of given B_n must satisfy the following assumption:

Assumption 2. *(Slater condition). There exist an $\epsilon > 0$ and a stationary policy of partial decisions $\overline{\boldsymbol{y}} \in \mathcal{Y}_0$ satisfying*

$$\mathbb{E}_{\boldsymbol{\sigma}}[B_n - g_n(\overline{\boldsymbol{y}}_n, \boldsymbol{\sigma})] \leq -\epsilon, \quad \forall n \in \mathcal{N}.$$

However, as implementing offline management policies is practically infeasible due to complexity and lack of whole future information. It calls for an online approach that can adjust resource provision decisions on the fly[4] to accomodate system dynamics. Thus, an online framework is desired, which makes irrevocable provisioning decisions based only on the current and limited predictive future information. However, devising efficient online algorithm is highly non-trivial due to the complicated couplings: 1) Constraint (1) imposes challenging temporal couplings across consecutive time slots for online services; 2) Constraint (7) implies long-run time-average requirement, which leads to complex temporal couplings across the whole time horizon for offline services; 3) Constraint (4) further induces the intricate resource contention couplings between both online and offline services.

[4]Note that the offline policy requires the future system information to optimize the resource provisioning decisions over the long run. While online policy optimizes the resource scheduling based on only current system information (e.g. current input workloads). Thus, the physical meanings of "online" and "offline" for resource provisioning policies are different from the meanings of "online" and "offline" for edge services.

IV. Reactive resource provisioning policy

In this section, we consider a reactive resource provisioning policy, where the proxy first observes the current system state $\boldsymbol{\sigma}^t$, and then optimizes the resource provisioning decisions based on the utility functions. To address challenges of the hybrid couplings mentioned above, we first leverage the idea of Lagrange relaxation to decouple the long-term constraint into a series of independent optimization problems. And accordingly, we propose an efficient constraint-aware stochastic subgradient algorithm to form the reactive resource provisioning policy.

A. Problem Relaxation

Based on the mentioned assumptions in Section III-B, we first consider a more tractable problem $\mathbf{P_1}$ as follows:

$$\begin{aligned}\mathbf{P_1}: \quad -\overline{F}^* := &\min_{\boldsymbol{x}^t, \boldsymbol{y}^t} -\overline{F}\\ \text{s.t.} \quad &(2),(3),(4),\\ &B_n - \overline{B}_n \leq 0, \forall n \in \mathcal{N}. \end{aligned} \tag{9}$$

Compared with the original one $\mathbf{P_0}$, the switching constraints (1) of online services are removed and long-term performance constraints (7) are replaced by (9) in $\mathbf{P_1}$. After such transformations, the hybrid temporal couplings for provisioning decisions have been largely simplified, i.e., the varying domain for online services caused by their switching constraints becomes fixed, similar to offline services. Accordingly, only the follow-up impacts for long-term offline service performance need to be tackled on the fly within the resource-limited edge cluster.

Clearly, based on the assumption 1, the problem $\mathbf{P_1}$ is a relaxed version of problem $\mathbf{P_0}$, since any feasible provisioning solution $\{\boldsymbol{x}^t, \boldsymbol{y}^t\}_{t=1}^T$ for $\mathbf{P_0}$ in expectation is also feasible to $\mathbf{P_1}$. It in turn implies that $\overline{F}^* \geq F^*$, due to $T \to \infty$. Particularly, the optimal solution to $\mathbf{P_1}$ can be achieved by a stationary randomized policy that determines per-slot provisioning decisions purely as a function (possibly randomized) of the current state $\boldsymbol{\sigma}^t$ [24, Theorem 4.5]. By queue-based relaxation techniques [21], the problem of enforcing the long-term offline service performance, i.e., (9) can be transformed into queue recursion optimization with their Lagrange multipliers $\boldsymbol{\lambda} := [\lambda_1, \cdots, \lambda_N]^\top$. In this way, the long-term offline service performance can be incorporated into the timely online performance, where the Lagrange multipliers $\boldsymbol{\lambda}$ act as vital weight parameters to balance the online and offline service performance at each slot. Correspondingly, the proxy should minimize the Lagrangian over the all provisioning decisions:

$$D(\boldsymbol{\lambda}) := \min_{\boldsymbol{x}^t\in\mathcal{X}_-, \boldsymbol{y}^t\in\mathcal{Y}_0} -\overline{F} + \sum_{n\in\mathcal{N}} \lambda_n(B_n - \overline{B}_n), \tag{10}$$

where $\mathcal{X}_-$ is a fixed domain of online services by omitting the switching constraint (1), i.e., $\mathcal{X}_- = \{\boldsymbol{x} | x \text{ satisfy } (2),(4)\}$, and the dual problem is $\max_{\boldsymbol{\lambda}\geq 0} D(\boldsymbol{\lambda})$. Fortunately, by the

Lagrange dual approach, the multiple standard subgradient iteration of $\boldsymbol{\lambda}$ are guaranteed to converge to a neighborhood of the optimal multipliers $\boldsymbol{\lambda}^*$ for the dual problem with the distribution of system randomness [22]. Thus, the long-term performance of offline services are well decoupled for $\mathbf{P_1}$.

However, the long-term system dynamics are typically unknown and hard to predict in practice, making it hard to perform the subgradient iteration algorithm. Fortunately, a stochastic subgradient approach is widely adopted to derive the stochastic estimates on the fly. A major difference in our problem is that switching constraints of online services should be incorporated to ensure their stringent requirements.

B. Constraint-aware Stochastic Subgradient Algorithm

Driven by $\mathbf{P_1}$, we now add switching constraints back to guarantee the stable runtime environment of online services. Similar to the (10), the decoupled joint optimization of online and offline services at slot t is denoted as follows:

$$\Gamma^* := \min_{\boldsymbol{x}^t \in \mathcal{X}^t, \boldsymbol{y}^t \in \mathcal{Y}_0} -F^t(\boldsymbol{x}^t) + \sum_{n\in\mathcal{N}} \lambda_n^t (B_n - g_n^t(\boldsymbol{y}_n^t)). \quad (11)$$

Correspondingly, the Lagrange multipliers of offline services $\boldsymbol{\lambda}^t$ are updated with a static step size μ as follows:[5]:

$$\lambda_n^{t+1} = [\lambda_n^t + \mu(B_n - g_n^t(\boldsymbol{y}_n^t))]^+, \forall n \in \mathcal{N}. \quad (12)$$

Here, λ_n^t denotes the stochastic estimates of the Lagrange multipliers in the standard subgradient iteration in problem $\mathbf{P_1}$, which characterizes the follow-up impacts of offline service performance (or its buffer) at current slot t. Particularly, a large value of λ_n^t indicates more resources should be allocated to offline service n currently, since large amounts of offline services have been deferred yet. Note that previous provisioning decisions $\boldsymbol{x}^{t-1}$ in the switching constraints are not the optimization variables at current slot, which can be treated as a series of determined constants. Since the $[\cdot]^+$ operator is a convex function, the switching term (i.e., $\sum_{j\in\mathcal{J}}[x_{m,j}^t - x_{m,j}^{t-1}]^+$) is also a convex function. In this regard, (11) is a convex program, which can be efficiently solved in polynomial time by existing solvers [26]. The detailed procedure is summarized in Algorithm 1. In this way, the edge proxy can adaptively adjust resource provisioning to accommodate system dynamics based on the Lagrange multipliers, as shown in Fig. 2.

C. Performance Analysis

To facilitate the performance analysis, we first introduce a constructed problem $\mathbf{P_2}$ to act as the bridge that connects our constructed minimization problem (11) to the problem $\mathbf{P_0}$:

$$\mathbf{P_2}: \quad \tilde{\Gamma}^* := \min_{\tilde{\boldsymbol{x}}^t \in \mathcal{X}_-, \tilde{\boldsymbol{y}}^t \in \mathcal{Y}_0} -F^t(\tilde{\boldsymbol{x}}^t) + \sum_{n\in\mathcal{N}} \lambda_n^t (B_n - g_n^t(\tilde{\boldsymbol{y}}_n^t)). \quad (13)$$

Our solved problem used in Algorithm 1 is different from $\mathbf{P_1}$ in that: the Lagrange multiplier λ_n^t is updated according to (12) rather than the standard subgradient iteration algorithm. Besides, compared with (11), switching constraints (1) are absent in $\mathbf{P_2}$. Therefore, the value of $\mathbf{P_2}$ is a lower bound

[5]The (unknown) expected amount B_n can be replaced by the desirable served one B_n^t. It would not affect the following performance analysis.

Algorithm 1: Reactive Resource Provisioning Policy

Initialization: a provisioning policy $\boldsymbol{x}^0 = \boldsymbol{y}^0 = \mathbf{0}$, step-size μ, and Lagrange multipliers $\boldsymbol{\lambda}^0 = \mathbf{0}$

for *each time slot* $t = 1, 2, \ldots$ **do**

  Acquire the Lagrange multiplier $\boldsymbol{\lambda}^t$ of offline services based on (12), current system state $\boldsymbol{\sigma}^t$ and utility functions

  Solve the decoupled joint optimization (11) to obtain decisions $\boldsymbol{x}^t$, $\boldsymbol{y}^t$, and perform them

end

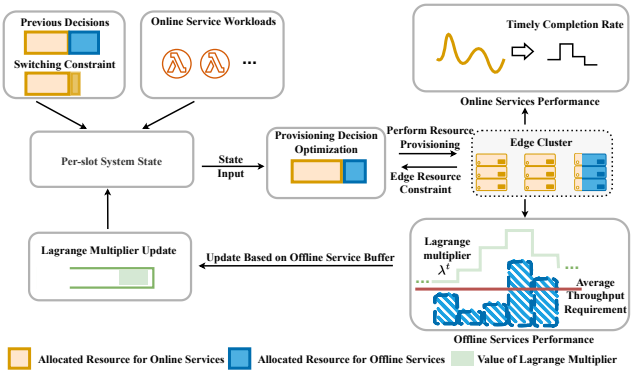

Fig. 2. Illustrative block diagram of reactive resource provisioning framework.

of our online decoupled problem, i.e., $\tilde{\Gamma}^* \leq \Gamma^*$, and their optimal gap is defined as $\delta(\boldsymbol{\Delta}) = \Gamma^* - \tilde{\Gamma}^*$.

To well quantify this gap $\delta(\boldsymbol{\Delta})$, we use some interesting insights to eliminate the impact of Lagrange multipliers $\boldsymbol{\lambda}^t$, since their intermediate results may be considerably large and hard to directly bounded over the long-run. A nice property is that decoupled instantaneous optimizations (11) and (13) can be separately optimized for partial decisions $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ once their total assigned resources are determined. More explicitly, let $\pi^t := \{\boldsymbol{x}^t, \boldsymbol{y}^t\}$ be the optimal solution of (11), and $\tilde{\pi}^t := \{\tilde{\boldsymbol{x}}^t, \tilde{\boldsymbol{y}}^t\}$ the optimal solution of (13). Besides, the total assigned resources in (11) are denoted as $C_x^t := \boldsymbol{c}^\top \boldsymbol{x}^t$ and $C_y^t := \boldsymbol{c}^\top \boldsymbol{y}^t$, respectively. Similarly, the total assigned resources in (13) are denoted as $\tilde{C}_x^t := \boldsymbol{c}^\top \tilde{\boldsymbol{x}}^t$ and $\tilde{C}_y^t := \boldsymbol{c}^\top \tilde{\boldsymbol{y}}^t$. Then we define the following sub-problems, respectively:

Definition 1. *The sub-problem optimization* $\mathbf{S_1}(C_y)$*, sub-problem optimization* $\mathbf{S_2^C}(C_x)$*, and sub-problem optimization* $\mathbf{S_2}(C_x)$ *are denoted as*

$$\mathbf{S_1}(C_y) := \min_{\boldsymbol{y}^t \in \mathcal{Y}_0} \sum_{n\in\mathcal{N}} -\lambda_n^t g_n^t(\boldsymbol{y}_n^t), \quad \text{s.t.} \sum_{n\in\mathcal{N}} \sum_{j\in\mathcal{J}} c_j z_{n,j}^t = C_y;$$

$$\mathbf{S_2^C}(C_x) := \min_{\boldsymbol{x}^t \in \mathcal{X}_-} -F^t(\boldsymbol{x}^t), \quad \text{s.t.} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} c_j x_{m,j}^t = C_x;$$

$$\mathbf{S_2}(C_x) := \min_{\boldsymbol{x}^t \in \mathcal{X}^t} -F^t(\boldsymbol{x}^t), \quad \text{s.t.} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} c_j x_{m,j}^t = C_x.$$

Lemma 1. *The optimal properties of sub-problems exist in terms of their solutions and function values:*

- *The resource allocations for both problem (11) and (13) satisfy* $C_x^t + C_y^t = C$*, and* $\tilde{C}_x^t + \tilde{C}_y^t = C$ *for all slot* t*;*

- *The optimal value of problem* $\mathbf{P_2}$ *is equal to the sum of optimal values of two sub-problem, i.e.,* $\mathbf{S_1}$ *and* $\mathbf{S_2^C}$ *with corresponding resource assignment* $\tilde{C}_y^t$ *and* $\tilde{C}_x^t$ *for any slot t, i.e.,*$\tilde{\Gamma}^* = \mathbf{S_1}(\tilde{C}_y^t) + \mathbf{S_2^C}(\tilde{C}_x^t)$*, as well for problem* $\mathbf{P_0}$*, i.e.,* $\Gamma^* = \mathbf{S_1}(C_y^t) + \mathbf{S_2}(C_x^t)$*;*
- *The optimal values of all sub-problems* $\mathbf{S_1}$*,* $\mathbf{S_2^C}$ *and* $\mathbf{S_2}$ *would decrease as the resource allocation increases, i.e., if* $C_1 \geq C_2$*, then* $\mathbf{S_1}(C_1) \leq \mathbf{S_1}(C_2), \mathbf{S_2}(C_1) \leq \mathbf{S_2}(C_2), \mathbf{S_2^C}(C_1) \leq \mathbf{S_2^C}(C_2)$.

Proof. For the first item, if the assigned total resource is not equal to the edge resource capacity, i.e., $C_x^t + C_y^t \neq C$ or $\tilde{C}_x^t + \tilde{C}_y^t \neq C$, it indicates that the remaining resources can be allocated to further minimize the total cost even when the previous partial decisions $\boldsymbol{x}^t$ are equal to $\mathbf{0}$, since the edge resource satisfies $\sum_{m\in\mathcal{M}} \min_{m\in\mathcal{M},j\in\mathcal{J}}\{\Delta_m c_j\} + \sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{J}} c_j Y_{n,j}^{max} \geq C$. Thus, the policy π^t and $\tilde{\pi}^t$ would be contrary to their optimality assumption. Obviously, the objective functions for both problem (11) and (13) are a linear combination of utility function $f_m^t(\cdot)$ and $g_n^t(\cdot)$ for all services. Besides, the optimal policies π^t and $\tilde{\pi}^t$ are feasible to their corresponding sub-problems. Once assigned resources C_x^t, C_y^t, $\tilde{C}_x^t$, and $\tilde{C}_y^t$ are determined, the optimal solutions derived by corresponding sub-problems should be equal to the optimal policies π^t and $\tilde{\pi}^t$, unless there exist multiple different optimal policies of joint optimization. Otherwise, it contradicts the optimality of policies π^t and $\tilde{\pi}^t$. Thus, the second item can be proved. The third item can be easily proved due to the non-decreasing property of utility functions $f_m^t(\cdot)$ and $g_n^t(\cdot)$. In other words, higher service performance can be achieved with larger assigned resources. □

Thus, the optimal gap between $\tilde{\Gamma}^*$ and Γ^* can be quantified by their sub-optimal problems, which only depend on either decision $\boldsymbol{x}^t$ or $\boldsymbol{y}^t$. Then, the optimal gap satisfies as follows:

Lemma 2. *The optimal gap* $\delta(\boldsymbol{\Delta})$ *can be upper-bounded with only respect to the objective values of online services, i.e.,*

$$\delta(\boldsymbol{\Delta}) \leq \begin{cases} \mathbf{S_2}(\tilde{C}_x^t) - \mathbf{S_2^C}(\tilde{C}_x^t), & if \quad C_x^t > \tilde{C}_x^t, \\ \mathbf{S_2}(C_x^t) - \mathbf{S_2^C}(\tilde{C}_x^t), & else, \end{cases}$$

and further satisfies

$$\delta(\boldsymbol{\Delta}) \leq \frac{U_g}{\min_j c_j}(\min\{C, \sum_{j\in\mathcal{J}}\sum_{m\in\mathcal{M}} c_j X_{m,j}^{max}\} - \min_{m\in\mathcal{M},j\in\mathcal{J}} \Delta_m c_j).$$

Proof. Due to space constraints, we just present a sketch of its proof. We analyze the optimal gap in two cases: the assigned resources satisfy $C_x^t > \tilde{C}_x^t$ or $C_x^t \leq \tilde{C}_x^t$. In the former case, the improvement of online service performance dominates the joint optimization considering the switching constraints. Due to the nature of $[\cdot]^+$, reducing the value of decision $\boldsymbol{x}^t$ would not violate the feasibility of switching constraints. It further indicates $\mathbf{S_2}(\tilde{C}_x^t)$ can be constructed in the sub-problem $\mathbf{S_2}(\cdot)$ of (11). Then, the gap between $\mathbf{S_1}(C_y^t)$ and $\mathbf{S_1}(\tilde{C}_y^t)$ satisfies:

$$0 \leq \mathbf{S_1}(C_y^t) - \mathbf{S_1}(\tilde{C}_y^t) \leq \mathbf{S_2}(\tilde{C}_x^t) - \mathbf{S_2}(C_x^t).$$

Otherwise, it would contradict the optimality of π^t. In the later case, based on the mentioned optimal property of sub-problem, we have $\mathbf{S_1}(C_y^t) - \mathbf{S_1}(\tilde{C}_y^t) \leq 0$. Combining two cases, the optimal gap can be bounded with only respect to the objective values of online services. Sequentially, we use the Assumption 1 of utility functions to further quantify the optimal gap. In the former case, we construct a feasible solution $\hat{\boldsymbol{z}}^t$ for online services, which approximates the optimal one $\tilde{\boldsymbol{x}}^t$ as much as possible. Thus, the resource change between $\tilde{\boldsymbol{x}}^t$ and $\hat{\boldsymbol{z}}^t$ can be denoted as $\Delta_C^t = \sum_m \sum_j c_j[\tilde{x}_{m,j}^t - \hat{z}_{m,j}^t]^+$, and the proxy needs to assign these resource Δ_C^t to worse configurations. Since the assigned resources hold $C_x^t > \tilde{C}_x^t$, the feasibility of constructed decisions can be guaranteed within the switching constraints. In this way, the optimal gap can be bounded as $\mathbf{S_2}(\tilde{C}_x^t) - \mathbf{S_2^C}(\tilde{C}_x^t) \leq U_g(\frac{\Delta_C^t}{\min_j c_j}) \leq \frac{U_g}{\min_j c_j}([\tilde{C}_x^t - \min_{m\in\mathcal{M},j\in\mathcal{J}} \Delta_m c_j]^+)$. The first inequality holds due to the non-decreasing and concavity of utility functions; the second inequality holds since the allocated resource must exceed the $\boldsymbol{c}^\top \boldsymbol{x}_m^{t-1} + \Delta_m \min_j c_j \geq \Delta_m \min_j c_j$ if $\Delta_C^t \neq 0$. The later case can be easily proved as $\mathbf{S_2}(C_x^t) - \mathbf{S_2^C}(\tilde{C}_x^t) = \mathbf{S_2}(C_x^t) - \mathbf{S_2^C}(C_x^t) + \mathbf{S_2^C}(C_x^t) - \mathbf{S_2^C}(\tilde{C}_x^t) \leq \frac{U_g}{\min_j c_j}([\tilde{C}_x^t - \min_{m\in\mathcal{M},j\in\mathcal{J}} \Delta_m c_j]^+)$. Thus, the Lemma 2 is proved. □

The lemma above states that the weighted performance gap of the offline services can be bounded to the one of online services, and the bounded optimality loss is highly related to the bound of gradients of utility functions $F^t(\cdot)$ and performance sensitivity $\boldsymbol{\Delta}$ over online services in set $\mathcal{M}$. On this basis, we further establish the following result.

Theorem 1. *The average timely online performance derived by our reactive resource provisioning algorithm satisfies*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t\in\mathcal{T}} \mathbb{E}[-F^t(\boldsymbol{x}^t)] \leq -F^* + \delta(\boldsymbol{\Delta}) + \mu H_1, \qquad (14)$$

where the constant $H_1 = \frac{1}{2}\sum_n \max\{B_n, U_y - B_n\}^2$ *, and* F^* *is the optimal value of* $\mathbf{P_0}$ *under any feasible solution.*

The proof of Theorem 1 is given in the technical report [27]. Based on the definition (8), the long-term constraints of offline services in set $\mathcal{N}$ can be guaranteed, due to $\lim_{T\to\infty} \frac{1}{T}\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}} \mathbb{E}\{Q_n^t\} = \mathcal{O}(\frac{1}{\mu}) < \infty$. Hence, Theorem 1 implies that, by decreasing parameter $\mu > 0$, the proxy can derive a near-optimal resource provisioning policy, at the expense of the increased accumulation of offline services. In addition, $\delta(\boldsymbol{\Delta})$ can become negligible when the switching constraints are loose.

V. Predictive resource provisioning policy

Predictions about the future play a key role in online optimization, which are widely considered in many applications. In practice, the proxy can leverage popular machine learning techniques to forecast future workloads for enabling more proactive resource management with informed decision making. Particularly, in order to derive high accurate performance, the size of the prediction window would not be very large, only few time slots. Thus, in this section, we advocate a predictive resource provisioning algorithm by incorporating these predictive results into our proposed reactive algorithm. Formally, we define a frame-based structure $\mathcal{W}_k$ to substitute the above slot-based structure, where the k-th ($k \in \{0, 1, \dots\}$) frame is defined as the prediction interval that contains slots

$\tau = kW, kW+1, \ldots, kW+W-1$, and its window size is denoted by an integer $W > 0$. At the beginning of the k-th frame, the proxy accurately predicts a series of the random states $\boldsymbol{\sigma}^{kW}, \ldots, \boldsymbol{\sigma}^{kW+W-1}$ over the W slots, and then outputs the provisioning decisions $\boldsymbol{x}^\tau$ and $\boldsymbol{y}^\tau$, $\tau \in \mathcal{W}_k$, for the entire k-th frame. Thus, similar to (11) used in our Algorithm 1, the predictive online optimization problem $\mathbf{P_{pre}}$ should be re-defined as follows:

$$\Gamma^*(W) := \min_{\boldsymbol{x}^\tau \in \mathcal{X}^\tau, \boldsymbol{y}^\tau \in \mathcal{Y}_0} \sum_{\tau \in \mathcal{W}_k} -F^\tau(\boldsymbol{x}^\tau) + \sum_{n \in \mathcal{N}} \lambda_n^\tau (B_n - g_n^t(\boldsymbol{y}_n^\tau))$$
$$\text{s.t.} \quad \lambda_n^\tau \text{ updated according to (12)}, \forall \tau.$$

However, different from the problem (11), a series of Lagrange multipliers of offline services in this joint optimization can not be regarded as given constants, but dependent variables in terms of decision $\boldsymbol{y}^t$. This change significantly complicates the problem $\mathbf{P_{pre}}$, since it may not be a convex optimization problem. To overcome this challenge, we adopt the first Lagrange multiplier λ_n^{kW} to approximate the future ones λ_n^τ over the prediction window, denoted as problem $\mathbf{P_A^{pre}}$:

$$\Gamma_A^*(W) := \min_{\boldsymbol{x}^\tau \in \mathcal{X}^\tau, \boldsymbol{y}^\tau \in \mathcal{Y}_0} \sum_{\tau \in \mathcal{W}_k} -F^\tau(\boldsymbol{x}^\tau) + \sum_{n \in \mathcal{N}} \lambda_n^{kW} (B_n - g_n^\tau(\boldsymbol{y}_n^\tau)). \tag{15}$$

The approximated problem $\mathbf{P_A^{pre}}$ is a convex problem since the max operator $[x_{m,j}^\tau - x_{m,j}^{\tau-1}]^+$ can be replaced by an auxiliary variable $z_{m,j}^\tau$, which satisfies linear inequalities, i.e., $z_{m,j}^\tau \geq 0$ and $x_{m,j}^\tau - x_{m,j}^{\tau-1} - z_{m,j}^\tau \leq 0$. In this regard, the problem $\mathbf{P_A^{pre}}$ can also be efficiently solved. Additional benefit of this approximation is the complexity reduction of future prediction for offline services with a limited window-size. Based on the modified stochastic subgradient algorithm, the detailed procedure of predictive one is presented in Algorithm 2.

Algorithm 2: Predictive Resource Provisioning Policy

Initialization: a provisioning policy $\boldsymbol{x}^0 = \boldsymbol{y}^0 = \mathbf{0}$, step-size μ, a prediction window size W, and Lagrange multipliers $\boldsymbol{\lambda}^0 = \mathbf{0}$
for *each time frame* $t = kW, k = 0, 1, \ldots$ **do**
  Predict successive system states $\boldsymbol{\sigma}^\tau$ over the given window and acquire the utility functions
  Solve the joint optimization (15) to obtain a series of decisions $\boldsymbol{x}^\tau$, $\boldsymbol{y}^\tau$, and perform them
  Update the Lagrange multipliers based on (12)
end

Particularly, we focus on the perfect prediction model and defer involved cases with noisy prediction in a future study. The extensions to noisy predictions with theoretical performance guarantee are difficult and have confirmed the insights initially proven in models with perfect predictions [28].

A. Performance Analysis

Similar to (13), we first construct a relaxation of approximated problem $\mathbf{P_A^{pre}}$, which is denoted as problem $\mathbf{P_{RA}^{pre}}$:

$$\tilde{\Gamma}_R^*(W) := \min_{\tilde{\boldsymbol{x}}^\tau \in \mathcal{X}_-, \tilde{\boldsymbol{y}}^\tau \in \mathcal{Y}_0} \sum_{\tau \in \mathcal{W}_k} -F^\tau(\tilde{\boldsymbol{x}}^\tau) + \sum_{n \in \mathcal{N}} \lambda_n^{kW} (B_n - g_n^\tau(\tilde{\boldsymbol{y}}_n^\tau)). \tag{16}$$

Compared with (15), switching constraints (1) are absent from (16) in the whole k-th frame; hence, it clearly holds that $\tilde{\Gamma}^*(W) \leq \Gamma^*(W)$. Besides, regardless of problem (15) and (16), the approximations hold the upper bound as shown in following lemma.

Lemma 3. *For each k-th frame in both (15) and (16), any possible provisioning decisions $\boldsymbol{x}^\tau$, and $\boldsymbol{y}^\tau$ are taken, we have*

$$\sum_{\tau \in \mathcal{W}_k} \sum_{n \in \mathcal{N}} \lambda_n^\tau \big(B_n - g_n^\tau(\boldsymbol{y}_n^\tau)\big) - \lambda_n^{kW} \big(B_n - g_n^\tau(\boldsymbol{y}_n^\tau)\big) \leq D,$$

where $D = \frac{W(W-1)\mu N}{2}\big((B_n - U_y)^2 + B_n U_y\big)$ is a constant, and the optimal gap between the problem (15) and (16) satisfies $\Gamma_A^*(W) \leq \tilde{\Gamma}_R^*(W) + \delta(\boldsymbol{\Delta}, W)$, where $\delta(\boldsymbol{\Delta}, W) = W\delta(\boldsymbol{\Delta}) - I(k, W) \leq W\delta(\boldsymbol{\Delta})$, and $I(k, W)$ denotes the performance improvement of a joint consideration over the whole frame compared to the single slot optimization.

Proof. Based on its definition (12), the Lagrange multiplier λ_n^τ satisfies the following inequalities:

$$\lambda_n^{kW} - \mu(\tau - kW)(U_y - B_n) \leq \lambda_n^\tau \leq \lambda_n^{kW} + \mu(\tau - kW)B_n, \forall n.$$

Thus, the Lagrange penalty during the prediction frame satisfies as follows:

$$\begin{aligned}
&\sum_{\tau \in \mathcal{W}_k} \sum_{n \in \mathcal{N}} \lambda_n^\tau (B_n - g_n^\tau(\boldsymbol{y}_n^\tau) \leq \sum_{\tau \in \mathcal{W}_k} \sum_{n \in \mathcal{N}} \lambda_n^{kW} (B_n - g_n^\tau(\boldsymbol{y}_n^\tau)) \\
&+ \mu(\tau - kW)(B_n^2 + (U_y - B_n)U_y) \\
&\leq \sum_{\tau \in \mathcal{W}_k} \sum_{n \in \mathcal{N}} \lambda_n^{kW} (B_n - g_n^\tau(\boldsymbol{y}_n^\tau) + D.
\end{aligned}$$

Then we construct a feasible solution $(\dot{\boldsymbol{x}}^\tau, \dot{\boldsymbol{y}}^\tau)$ over the k-th frame, which independently optimize the slot-problem:

$$\dot{\Gamma}^*(\tau) := \min_{\dot{\boldsymbol{x}}^\tau \in \mathcal{X}^\tau, \dot{\boldsymbol{y}}^\tau \in \mathcal{Y}_0} -F^\tau(\dot{\boldsymbol{x}}^\tau) + \sum_{n \in \mathcal{N}} \lambda_n^{kW} (B_n - g_n^\tau(\dot{\boldsymbol{y}}_n^\tau)).$$

Obviously, the value of $\Gamma_A^*(W) \leq \sum_{\tau \in \mathcal{W}_k} \dot{\Gamma}^*(\tau)$, since the decisions are made based on a joint consideration over the whole frame in optimization $\mathbf{P_A^{pre}}$. Such a performance improvement by prediction can be denoted as $I(k, W) = \sum_{\tau \in \mathcal{W}_k} \dot{\Gamma}^*(\tau) - \Gamma_A^*(W) \geq 0$. Besides, the joint optimization (16) can be solved by greedy minimizing slot-problem over the frame, since both $\tilde{\boldsymbol{x}}^\tau$, $\tilde{\boldsymbol{y}}^\tau$ decisions are time-invariant without considering the switching constraints. Then the Lemma 3 can be easily proved. □

Note that the value of performance improvement $I(k, W)$ highly depends on the constraint $\boldsymbol{\Delta}$ and the value of Lagrange multipliers of offline service, which is reflected by the step-size μ. Based on the above lemma, we can derive the performance guarantee result as follows:

Theorem 2. *The average timely online performance derived by our predictive resource provisioning algorithm satisfies*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[-F^t(\boldsymbol{x}^t)] \leq -F^* + \frac{\delta(\boldsymbol{\Delta}, W)}{W} + \mu H_2, \tag{17}$$

where $H_2 = H_1 + \frac{(W-1)N}{2}\big((B_n - U_y)^2 + B_n U_y\big)$ is a constant.

The proof of Theorem 2 is given in the technical report [27]. Performance evaluation in Section VI collaborates the substantial performance boosting by predictive resource provisioning even with some estimation errors (e.g. due to imperfect predictions).

VI. Performance Evaluation

A. Experimental Setup

In the co-located edge cluster, heterogeneous computing processors (e.g. CPU and GPU) can be virtualized into corresponding unit resources in terms of their averaged computing capacities [29]. In our simulation, taking Amazon EC2's C5 instance as an example [30], we consider three different configurations of containers c5.large, c5.xlarge, and c5.2xlarge with 2, 4, and 8 vCPUs, respectively ($J = 3$). A total of 100 vCPUs ($C = 100$) are allocated to five online services ($M = 5$) and four different offline services ($N = 4$) on demand. Besides, we adopt CVXPY [26] optimizer to solve the convex optimization problem of all online algorithms.

TABLE I
Simulation Parameters of Online Services

Model	Unit processed amount $a_{m,j}^t$		
	c5.large	c5.xlarge	c5.2xlarge
EfficientNet-b0	[3276, 3984]	[10933, 12155]	[23140, 24466]
EfficientNet-b1	[2898, 3118]	[8469, 9106]	[17490, 18420]
EfficientNet-b2	[2528, 2700]	[7651, 8296]	[16710, 17166]
EfficientNet-b3	[1290, 1742]	[5841, 6408]	[12354, 12966]
EfficientNet-b4	[1054, 1414]	[4585, 5265]	[9832, 10852]

Online Services. We consider online services as image classification based recognition tasks on ImageNet, which requires different DNN inference models to achieve their goal (e.g. accuracy). Further, we use the number of dynamic passengers at London's underground stations and select the passenger amount of the five largest stations (i.e., Waterloo, King's Cross, Victoria, Pancras, and Paddington) to represent the workload arrival of each online service, respectively [31]. Besides, we add a constant $\beta = 10$ to the enlarged ($\times 100,000$) normalized values to prevent the denominator from being 0 in 5. Note that we assume that each container configuration j is associated with its resource, i.e., configurations of containers c5.large, c5.xlarge, and c5.2xlarge, respectively. Here, A_m^t represents the processed workload arrivals, and $a_{m,j}^t$ represents the unit processed amount of online services under different configurations. Particularly, we adopt a popular DNN model, EffcientNet in [32], to measure these parameters. By choosing moderate batch-size of input, the unit processed amount $a_{m,j}^t$ is calculated based on the recorded inference speed and length of slot when the assigned resource in container is fully utilized.

Offline Services. We consider four different offline applications, and B_n represents the average throughput threshold of each offline application, i.e., the file size (MB) processed in a unit time slot (15 minutes). Specifically, we associate with the one-week quarterly traces of the data centers of Google, Facebook, HP, and Microsoft to each offline application [16], respectively. We take the mean value of each trace in 672 time slots and enlarge it by a factor of 2,000 as $B_n = (1444, 1350, 1168, 446)$. As shown in Table II, the parameter $p_{n,j}$ represents the throughput of a container with configuration j in a single time slot for application n, which is calculated by the given processing densities of several typical applications in [33]. Besides, we assume b_n^t is the Gaussian noise (the mean and variance is 0 and 100, respectively) introduced in each time slot.

TABLE II
Simulation Parameters of Offline Services

Configuration	Unit throughput $p_{n,j}$			
	Service 1	Service 2	Service 3	Service 4
c5.large	454	410	308	272
c5.xlarge	908	820	616	544
c5.2xlarge	1816	1640	1232	1088

B. Benchmarks

To verify the efficacy of the proposed algorithms, we compare them to three benchmarks.

StaticOpt: The statistics of system dynamics are available in advance, e.g. the arrival rates of online services A_m^t and the Gaussian noise b_n^t for offline services. Then the static (fixed) optimal resource provisioning policy can be directly derived by existing convex optimization techniques [26].

Dynamic resource partition (DRP): Based on the above static optimal algorithm, edge resources can be partitioned for processing online and offline services, respectively. Inspired by work [6], a random perturbation parameter, called resource-stealing, is introduced to enable dynamical adjustment of resource partitions, which follows a uniform distribution (i.e., [-10,10]).

MPC-based algorithm [34]: As a competitive benchmark to our predictive provisioning algorithm, MPC-based solution in each slot solves the optimization problem in (15) by looking ahead over a prediction window, and performs only the first slot decision rather than the entire decisions of the whole prediction window to enable adaptive adjustments. Thus, its computational complexity (i.e., amount of prediction and provisioning decision-making) is W times more than ours.

C. Experimental Results

Obviously, the performance of our proposed algorithms depend on parameters μ and W and $\mathbf{\Delta}$. We will further discuss these impact with different values of parameters. Unless otherwise specified, we set the parameters as $\Delta = 3, W = 6$, and $T = 672$ in the following simulation.

Effect of the step-size μ: Fig. 3 plots the averaged percentage of timely completion rate for online services under different algorithms with different step-sizes μ. As the value of step-size μ decreases, the averaged completion rates of proposed online algorithms gradually converge to a near-optimal performance, which confirms theoretical analysis of online service performance as characterized by Theorem 1 and 2. This is due to the fact that a smaller μ emphasizes the importance of online services and thus defers delay-tolerant workloads aggressively, which is reflected by intensifying the congestion of buffer queue for offline services, as shown in Fig. 4. Specially for predictive algorithms, the double-edged sword effect of step-size μ is more evident. On the one hand, the benefit of prediction (i.e., the performance improvement

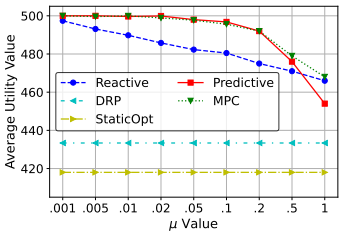

Fig. 3. Effect of step-size μ on the average online completion rate.

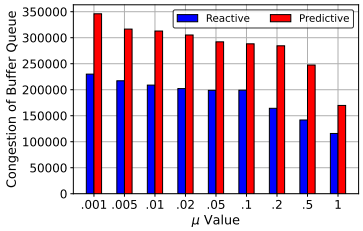

Fig. 4. Effect of step-size μ on the congestion level of buffer queue.

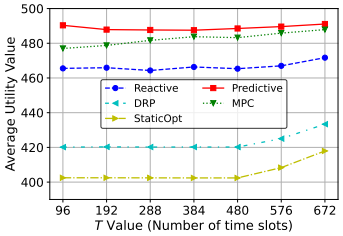

Fig. 5. Dynamic average online performance with time slot increasing.

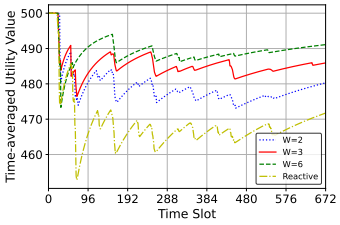

Fig. 6. Time-averaged online performance with different prediction window size W.

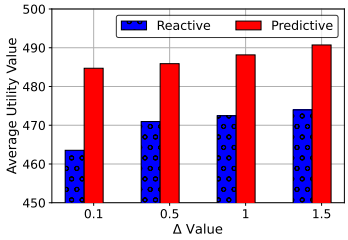

Fig. 7. Effect of $\mathbf{\Delta}$ on the time-averaged online service performance.

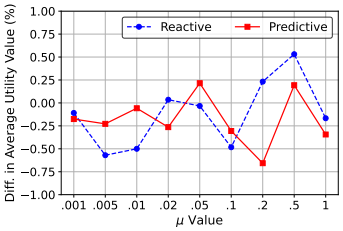

Fig. 8. Robustness test results.

$I(k, W)$) plays a dominant position with small μ. However, service performance considerably deteriorates with large step-size μ, due to large approximation error of future λ_n^τ (e.g. the constant H_2 in Theorem 2) in the predictive window. Thus, compared with our predictive provisioning algorithm, the MPC-based one achieves better online performance with large μ, since approximately Lagrange multipliers are more accurate by more frequent updates (i.e., W times). We fix μ to be 0.25 and vary T from 96 time slots (one day) to 672 time slots (one week), which is a sufficient range for exploring the utilization of proposed algorithms. As shown in Fig. 5, the proposed predictive algorithm improves the completion rate by around 5%, 13.5%, and 17.5%, compared to the reactive, DRP and StaticOpt ones, respectively.

Effect of the prediction window size W: We fix μ to be 0.25 and plot the time-averaged online service performance for different prediction window size W under the predictive algorithm in Fig. 6. Note that the reactive algorithm is a special case of the predictive algorithm when $W = 1$. Clearly, the predictive algorithm outperforms the reactive algorithm for three different window sizes. At the same time, with the increase of W, the performance of the predictive algorithm itself is also improved to a certain extent, since more future information assist the proxy to determine more reasonable decisions for their hybrid temporal couplings, especially for online services. However, as the length of the prediction window increases, the difficulty of accurate prediction increases correspondingly.

Effect of the control parameter $\mathbf{\Delta}$: We tune the control parameter $\mathbf{\Delta}$ in constraint (1) and compare the performance of reactive algorithm and predictive algorithm as shown in Fig. 7. We observe that as the control parameter increases (i.e., the online service becomes less sensitive to resource allocation latency), the solution space in each slot of the reactive algorithm and predictive algorithm also becomes larger, and thus the online service performance improves. It means that the sensitivity of online service to latency will affect its performance, so it is necessary to introduce stable constraint in our model. At the same time, the overall change of the predictive algorithm is small, which to a certain extent indicates that the future information in the prediction window is conducive to weakening this limitation, because the algorithm can break the performance bottleneck of online services by pre-configuring resources.

Robustness analysis: Intuitively, the prediction window becomes longer and the difficulty of accurate prediction gradually increases. To analyze the robustness of the proposed algorithms, we explore the influence of estimation errors on the performance of reactive and predictive algorithm. For each online service, we add a random estimated error (50%, uniformly distributed) to the amount of workload it contains. We run experiments for different μ values and compare the results to those obtained using the original workload data sets. In Fig. 8, we use the results on data sets *without* estimation errors as the baseline and show the differences in average utility value as a percentage due to injected estimation errors. Fig. 8 indicates that for all μ values we experimented with, the difference (due to errors) in average utility value is between -0.75 and 0.75 percent, which can be almost neglected.

VII. Conclusion

In this paper, we study an edge-centric resource provisioning optimization. To tackle hybrid temporal couplings arisen by heterogeneous constraints of co-located services and different time-scale performances, we first propose a reactive provisioning approach to accommodate system dynamics, which leverages the Lagrange relaxation for devising a constraint-aware stochastic subgradient algorithm. Moreover, we also advocate a predictive provisioning approach by integrating machine learning techniques to boost online service performances. Via both rigorous theoretical analysis and extensive trace-driven simulations, we verify the efficacy of the proposed algorithms for online and offline services co-location at the edge.

REFERENCES

[1] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: a first look at public edge platforms," in *IMC '21: ACM Internet Measurement Conference, Virtual Event, USA, November 2-4, 2021*. ACM, 2021, pp. 37–53.

[2] J. Ren, G. Yu, and G. Ding, "Accelerating DNN training in wireless federated edge learning systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 219–232, 2021.

[3] A. Ali-Eldin, B. Wang, and P. J. Shenoy, "The hidden cost of the edge: a performance comparison of edge and cloud latencies," in *SC '21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021*. ACM, 2021, pp. 23:1–23:12.

[4] W. Chen, J. Rao, and X. Zhou, "Preemptive, low latency datacenter scheduling via lightweight virtualization," in *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017*. USENIX Association, 2017, pp. 251–263.

[5] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. M. Swift, "Peeking behind the curtains of serverless platforms," in *2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*. USENIX Association, 2018, pp. 133–146.

[6] P. Delgado, F. Dinu, A. Kermarrec, and W. Zwaenepoel, "Hawk: Hybrid datacenter scheduling," in *2015 USENIX Annual Technical Conference, USENIX ATC '15, July 8-10, Santa Clara, CA, USA*. USENIX Association, 2015, pp. 499–510.

[7] J. Li, H. Xu, Y. Zhu, Z. Liu, C. Guo, and C. Wang, "Aryl: An elastic cluster scheduler for deep learning," *CoRR*, vol. abs/2202.07896, 2022.

[8] Y. Yao, H. Gao, J. Wang, B. Sheng, and N. Mi, "New scheduling algorithms for improving performance and resource utilization in hadoop YARN clusters," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1158–1171, 2021.

[9] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*. USENIX Association, 2020, pp. 205–218.

[10] N. Akhtar, A. Raza, V. Ishakian, and I. Matta, "COSE: configuring serverless functions using statistical learning," in *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*. IEEE, 2020, pp. 129–138.

[11] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, 2022.

[12] W. You, L. Jiao, S. Bhattacharya, and Y. Zhang, "Dynamic distributed edge resource provisioning via online learning across timescales," in *17th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2020, Virtual Event, Italy, June 22-25, 2020*. IEEE, 2020, pp. 1–9.

[13] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mob. Comput.*, vol. 20, no. 3, pp. 939–951, 2021.

[14] Y. Jin, L. Jiao, Z. Qian, S. Zhang, N. Chen, S. Lu, and X. Wang, "Provisioning edge inference as a service via online learning," in *17th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2020, Virtual Event, Italy, June 22-25, 2020*. IEEE, 2020, pp. 1–9.

[15] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, 2019.

[16] Z. Zhou, S. Yu, W. Chen, and X. Chen, "Ce-iot: Cost-effective cloud-edge resource provisioning for heterogeneous iot applications," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8600–8614, 2020.

[17] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, 2021.

[18] S. Shillaker and P. R. Pietzuch, "Faasm: Lightweight isolation for efficient stateful serverless computing," in *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*, A. Gavrilovska and E. Zadok, Eds. USENIX Association, 2020, pp. 419–433.

[19] M. Kwon, D. Gouk, C. Lee, B. Kim, J. Hwang, and M. Jung, "Dc-store: Eliminating noisy neighbor containers using deterministic I/O performance and resource isolation," in *18th USENIX Conference on File and Storage Technologies, FAST 2020, Santa Clara, CA, USA, February 24-27, 2020*, S. H. Noh and B. Welch, Eds. USENIX Association, 2020, pp. 183–191.

[20] B. Sun, A. Zeynali, T. Li, M. H. Hajiesmaili, A. Wierman, and D. H. K. Tsang, "Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, pp. 51:1–51:32, 2020.

[21] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, and M. J. Neely, "Power cost reduction in distributed data centers: A two-time-scale approach for delay tolerant workloads," *IEEE Trans. Parallel Distributed Syst.*, vol. 25, no. 1, pp. 200–211, 2014.

[22] T. Chen, X. Wang, and G. B. Giannakis, "Cooling-aware energy and workload management in data centers via stochastic optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 402–415, 2015.

[23] H. Yu, M. H. Cheung, L. Huang, and J. Huang, "Predictive delay-aware network selection in data offloading," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 1376–1381.

[24] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, ser. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers, 2010.

[25] M. Shi, X. Lin, and S. Fahmy, "Competitive online convex optimization with switching costs and ramp constraints," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 876–889, 2021.

[26] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[27] T. Report, "Edge-centric resource provisioning for online and offline services co-location," https://1drv.ms/b/s!AqHdoWGUzboyvWUJPh6zM8GuIIsk?e=UyIW86, 2022.

[28] Y. Lin, G. Goel, and A. Wierman, "Online optimization with predictions and non-convex losses," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 1, pp. 1–32, 2020.

[29] C. Zhang, M. Yu, W. Wang, and F. Yan, "Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving," in *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*, D. Malkhi and D. Tsafrir, Eds. USENIX Association, 2019, pp. 1049–1062.

[30] A. EC2, "Configuration information," https://https://aws.amazon.com/ec2/instance-types/, 2022, [Online; accessed 23-February-2022].

[31] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2018, pp. 1–9.

[32] L. Melas-Kyriazi, "Efficientnet-pytorch," https://github.com/lukemelas/EfficientNet-PyTorch, 2022, [Online; accessed 7-March-2022].

[33] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[34] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *2012 International Green Computing Conference, IGCC 2012, San Jose, CA, USA, June 4-8, 2012*. IEEE Computer Society, 2012, pp. 1–10.