Online Social Welfare Maximization with Spatio-Temporal Resource Mesh for Serverless

Hailiang Zhao*, Shuiguang Deng*†, Zhengzhe Xiang†, and Jianwei Yin*

*Zhejiang University, †Zhejiang University City College

Emails: *{hliangzhao, dengsg, zjuyjw}@zju.edu.cn, †xiangzz@zucc.edu.cn

Abstract—Serverless computing is leading the way to a simplified and general purpose programming model for the cloud. A key enabler behind serverless is efficient load balancing, which routes continuous workloads to appropriate backend resources. However, current load balancing algorithms implemented in Kubernetes native serverless platforms are simple heuristics without performance guarantee. Although policies such as Pod or JFIQ yield asymptotically optimal mean response time, the information they depend on are usually unavailable. In addition, dispatching jobs with strict deadlines, fractional workloads, and maximum parallelism bound to limited resources online is difficult because the resource allocation decisions for jobs are intertwined. To design an online load balancing algorithm without assumptions on distributions while maximizing the social welfare, we construct several pseudo-social welfare functions and cost functions, where the latter is to estimate the marginal cost for provisioning services to every newly arrived job based on present resource surplus. The proposed algorithm, named OnSocMax, works by following the solutions of several convex pseudo-social welfare maximization problems. It is proved to be \( \alpha \)-competitive for some \( \alpha \) at least 2. We also validate OnSocMax with simulations and the results show that it distinctly outperforms several handcrafted benchmarks.

I. INTRODUCTION

Today’s cloud computing is a paradigm of computing as a utility [1]. To reduce the tedious process to configure the VM-based machines while improving the QoE of end users, a computation paradigm, named serverless, emerged and has enormous potential to be the dominant way of building elastic cloud services [2]. [3]. It erases the heavy burden to use the cloud by handling the system administration operations virtually, including the installation of operating systems, libraries, and runtime dependencies. The prominent technical enablers underlying serverless is on-demand resource provisioning and seamless auto-scaling, based on multi-tenant isolation in different granularities [4] and efficient load balancing algorithms [2]. [3]. Load balancer is a component to expose the APIs to the workloads through mapping the HTTP requests to carefully selected backend resources. Theoretically, it decides how the workloads are dispatched to distributed computing instances and how to allocate restricted resources to them, with a target of optimizing the QoS of end users [5]–[8].

A fundamental challenge to load balancer is that, in real-life scenarios, workload dispatching and resource allocation decisions should be made online without knowledge about job arrivals. However, load balancing strategies implemented in current industrial serverless platforms, such as RoundRobin and SessionAffinity [5] are simple to operate but offer no performance guarantee. Meanwhile, a series of academic studies on online load balancing, which promise long-term performance guarantees, assume that jobs arrive according to Poisson process and service rates of computing instances are exponentially distributed [9]–[13]. Under stochastic ordering assumption, policies such as Join-the-Shortest-Queue (JSQ) [14], Join-the-Idle-Queue (JIQ) [11], Power-of-d-Choices (Pod) [12], and Join-the-Fastest-of-the-Shortest-Queues (JFIQ) [9] are raised and analyzed based on Continuous-Time Markov Chains (CTMC) and Lyapunov Stability theories. However, their performance guarantees (mostly on mean response time) are established on sufficient assumptions, which are tough to satisfy in production systems. Further, if we take heterogeneous service rates, service locality [5] strict deadlines and many other realistic constraints into account, the performance guarantees are even harder to achieve. Another problem that cannot be ignored is that, these policies operate in a FCFS manner and job preemption is not allowed. The other series of researches concentrate on energy-efficient geographical load balancing [5]–[7], [15]. [16]. The intention is either reducing non-renewable energy consumption or increasing energy efficiency of distributed computing instances with inexpensive approaches for enabling large-scale on-demand response. However, due to the unpredictable job arrivals and deadlines, it is difficult for them to choose between more valuable jobs with larger deadlines and less valuable yet more emergent jobs [15].

Online load balancing of deadline-sensitive jobs have been considered mostly from the perspective of either jobs’ utility maximization (by measuring the delay or QoS), or the revenue of the serverless platform [5]. To the best of our knowledge, for serverless, none of works have been found which study the problem from the perspective of social welfare maximization, where the utilities of jobs and the revenue of the serverless platform are maximized simultaneously. For load balancing of fractional workloads with maximum parallelism bounds, we find only one previous work [15]. We compare the contributions of this work with ours in Sec. V. Another recent work on social welfare is [17], which studies a generic load balancing problem based on the game-theoretical techniques. However, this work does not consider strict job deadlines and

1https://kubernetes.io/docs/concepts/services-networking/service/
2Service locality means the required functions cannot be executed on the chosen resource unit because the runtime or dependencies are not satisfied.
the characteristics of serverless. Moreover, the social welfare formulated in this paper is the geometric mean of jobs’ costs, which is completely different from our work. To fill the gap, in this paper, we study the online load balancing problem for deadline-sensitive jobs by maximizing the sum of utilities of both jobs and the platform. Our model is established based on the so-called resource mesh of spatio and temporal resource units, which will be detailed in Sec. II-A. Each arrived job with fractional workloads can only be dispatched to its available resource units. We design an algorithm, OnSocMax, to decide how the input workloads of arrived jobs are partitioned and dispatched under realistic constraints in serverless. To name a few, anti-affinity service locality and unpredictable system failures of computing instances, etc. Specifically, OnSocMax works by solving several well-designed pseudo-social welfare maximization problems online, and has no assumptions on the arrival pattern and service rates. We provide rigorous analysis to show that OnSocMax is $\alpha$-competitive for some $\alpha \geq 2$. Our contributions are summarized as follows.

- We first establish the resource mesh model for serverless and study the online social welfare maximization problem for deadline-constraint jobs with fractional workloads and parallelism bounds. Our model farthest approximate the real-world serverless systems and has assumptions only on the utility functions.
- We propose an online algorithm OnSocMax which yields a competitive ratio at least 2 for general utility settings. Particularly, it has a linear complexity when the utility of jobs are linear and share the same coefficient.

The rest of this paper is organized as follows. We introduce the system model and formulate the problem in Sec. II. We then present design details of the online algorithm OnSocMax with sufficient theoretical analysis in Sec. III. We present the numerical results in Sec. IV and discuss related work in Sec. V. This paper is concluded in Sec. VI.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a serverless computing platform such as AWS Fargate and Google Cloud Functions. The serverless platform is composed of multiple computing instances distributed across different data-centers and CDN sites geographically, which are managed by the containerized orchestration framework Kubernetes. We build our model on serverless because it removes the need to explicitly provision and manage computing instances, as well as the underlying resources are suited to be meshed. Let us use $\mathcal{K}$ to denote the set of computing instances and index each of them by $k$.

A. Spatio-Temporal Resource Mesh

The platform is capable of processing a set of heterogeneous jobs arriving in sequence with different service rates. Let us denote the set of jobs as $\mathcal{N}$ and index each job by $n$. Each job $n$ has the input workloads of size $q_n$. Take video transcoding with ExCamera [18] as an example, the inputs are raw video frames. ExCamera firstly partition the input into frame pieces with negligible cost. Then, it parallelizes the “slow” pieces of the encoding, and performs the “fast” pieces serially [19]. For discrete jobs not suitable for parallelism, our model is still applicable with appropriate rounding policies, for example, the Fenchel duality [20] taken in [15]. More details are presented in Sec. III-C. $\forall n \in \mathcal{N}$, we use $a_n$ and $d_n$ to represent its arrival time and deadline to be finished. To maximize the utilities of the serverless function users and the revenue of the platform from a long-term vision, we consider the time horizon from $\min_{n \in \mathcal{N}} a_n$ to $\max_{n \in \mathcal{N}} d_n$ and evenly divide the horizon into slots of length $\tau$. Let us use $\mathcal{T}$ to denote the set of time slots and index each of them with $t$. The time slot length $\tau$ can be set as one fourth of the minimum instance reserved time, for example, 15 minutes for AWS spot instance$^4$.

To manipulate the computing resources in $\mathcal{K}$ from both dimension of time and space, we introduce a spatio-temporal resource division model called resource mesh. We use $\mathcal{R} \triangleq \mathcal{K} \times \mathcal{T}$ to denote the set of resource units and index each of them by $r$. Each resource unit $r$ can process at most $C_r$ workloads of all jobs. This value could be obtained through a variety of approaches from static code analysis to profiling.

3In Kubernetes, affinity and anti-affinity constrain which instances a pod is eligible to be scheduled on, based on labels on the instance.

4https://aws.amazon.com/ec2/spot/pricing/
previous runs based on hardware heterogeneity [19]. For each job \( n \in \mathcal{N} \), we use
\[
\mathcal{R}_n \triangleq \{ r_{klt} \in \mathcal{R} \mid \left| \frac{d_n}{r} \right| \leq t \leq \left| \frac{d_n}{r} \right|, k \in \mathcal{K} \}
\tag{1}
\]
to denote its available resource units. Fig. [1] gives an example.

### B. Utility and Revenue Functions

For each job \( n \in \mathcal{N} \), we need to decide how to dispatch its workloads to its available resource units under parallelism limit and deadline constraint, with the purpose of maximizing the social welfare, which is the sum of all jobs’ utilities and the revenue of the serverless platform. Formally, we use \( x_{nr} \) to denote the size of workloads dispatched to \( r \in \mathcal{R}_n \) and \( \chi_{nr} \) to denote the parallelism bound when it is processed on \( r \). It results to the constraint \( 0 \leq x_{nr} \leq \chi_{nr} \). We design this bound too high degree of parallelism leads to non-negotiable communication overhead and even unforeseen errors [15]. Besides, this formulation takes the anti-affinity, service locality and unpredictable system failure of computing instances into consideration. When those conditions happen to resource unit \( r \) during processing the \( n \)-th job, we can simply set \( \chi_{nr} \), as zero online.

We take a zero-startup utility function \( f_n : [0, \chi_{nr}] \rightarrow \mathbb{R} \), where \( \chi_{nr} \triangleq \{ \chi_{nr} \}_{r \in \mathcal{R}_n} \), as the measurement of user satisfaction for job \( n \). As a widely accepted assumption in previous works [6], [15], [21]–[23], we require \( \{ f_n \}_{n \in \mathcal{N}} \) to be non-decreasing, concave, and continuously differentiable on each dimension \( r \). Proportional fairness and \( \alpha \)-fairness are good options for \( \{ f_n \}_{n \in \mathcal{N}} \) [24]. Note that we allow jobs in \( \mathcal{N} \) to have different utility functions. For each job \( n \in \mathcal{N} \), its utility is a sum of separate sub-utilities achieved through each available resource unit:
\[
f_n(x_n) \triangleq \sum_{r \in \mathcal{R}_n} f_{nr}(x_{nr}) \quad \forall n \in \mathcal{N} \tag{2}
\]

where \( x_n \triangleq \{ x_{nr} \}_{r \in \mathcal{R}_n} \). For a given job \( n \), \( f_{nr} \) can also be different on different resource unit \( r \in \mathcal{R}_n \). To sum up, a job can be described with the quadruple \( \{ q_n, \mathcal{R}_n, \chi_{nr}, f_{nr} \} \).

Serverless comes with a pay-for-value billing model [19]. Thus, its revenue is linearly proportional to the actual resource consumption. Formally, we define the revenue for provisioning resource \( r \in \mathcal{R} \) as
\[
g_{nr}(x_{nr}) \triangleq \beta_{nr} \cdot \frac{x_{nr}}{C_r} \quad \forall n \in \mathcal{N}, r \in \mathcal{R}_n \tag{3}
\]

where \( \frac{x_{nr}}{C_r} \) is the fractional resource consumed by \( x_{nr} \), and \( \beta_{nr} \) is a ratio indicating the unit price per resource unit for job \( n \). For instance, \( \beta_{nr} \) equals to $0.015 when the raw video’s resolution is less than 720p with Amazon Elastic Transcoder [25]. Actually, our model is consistent with all the mainstream platform providers’ pricing strategies [25]–[28].

### C. Online Social Welfare Maximization

Based on the above content, we formulate the social welfare maximization problem as follows:
\[
P_1 : \max_{\{ x_n \}_{n \in \mathcal{N}}} \sum_{n \in \mathcal{N}} f_{n}(x_{n}) + \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}_n} g_{nr}(x_{nr})
\tag{4}
\]
s.t.
\[
\sum_{r \in \mathcal{R}_n} x_{nr} \leq q_n, \forall n \in \mathcal{N},
\tag{5}
\]
\[
x_{nr} = 0, \forall n \in \mathcal{N}, r \in \mathcal{R} \setminus \mathcal{R}_n,
\tag{6}
\]
\[
\sum_{n \in \mathcal{N}} x_{nr} \leq C_r, \forall r \in \mathcal{R},
\tag{7}
\]
\[
0 \leq x_{nr} \leq \chi_{nr}, \forall n \in \mathcal{N}, r \in \mathcal{R}_n.
\tag{8}
\]

As an offline optimization problem, although \( P_1 \) is difficult to solve [4] it is built based on complete knowledge. In online settings, the platform should not have the information of the \( n \)-th quadruple \( \{ q_n, \mathcal{R}_n, \chi_{nr}, f_{nr} \} \) until job \( n \) arrives. To design an efficient online algorithm with the worst-case performance guarantee, we introduce the following notations
\[
\ell \triangleq \min_{n \in \mathcal{N}} \min_{r \in \mathcal{R}_n} \{ \frac{\partial f_{nr}}{\partial x_{nr}} + \frac{\partial g_{nr}}{\chi_{nr}} \}
\tag{9}
\]
\[
u \triangleq \max_{n \in \mathcal{N}} \max_{r \in \mathcal{R}_n} \{ \frac{\partial f_{nr}}{\partial x_{nr}} + \frac{\partial g_{nr}}{\chi_{nr}} \}
\tag{10}
\]

and assume that the platform know them at the very beginning. Those two constants demonstrates the fluctuation of the marginal social welfare. This assumption is widely accepted in the online resource allocation problems [22], [30]–[34]. For example, in [32], the fluctuation ratio \( \frac{\ell}{\nu} \) is set as 36 in default.

### III. Algorithm Design with Theoretical Analysis

The key challenge to solve \( P_1 \) in online settings lies in that the dispatching of each job’s workloads to each resource unit are coupled because of (9). Nevertheless, if we could construct several feasible dual variables corresponding to \( P_1 \), and take these dual variables as the cost for using each resource unit, a near optimal solution could be obtained. To do this, we design several pseudo-social welfare functions with estimated marginal costs. In this design, we utilize an important principle for solving online resource allocation problems, i.e., estimate the cost for provisioning services to each newly arrived job as a function of resource surplus [22], [23], [31], [32], [35].

In the following sections, firstly, we show how the pseudo-social welfare functions are designed. Then, based on these functions, we introduce our algorithm \textsc{OnSocMax} by solving several pseudo-social welfare maximization problems online. To guarantee \textsc{OnSocMax} is \( \alpha \)-competitive, we demonstrate what requirements the cost functions should satisfy. In the end, we give the bound of the gap between the competitive ratio achieved by \textsc{OnSocMax} and the optimal competitive ratio of a simplified case under certain conditions.

\footnote{The discrete version of \( P_1 \) is actually a multi-dimensional 0-1 knapsack problem, which is proved to be NP-complete [29].}
A. Pseudo-Social Welfare Function

For each arrived job \( n \), we define the pseudo-social welfare function, denoted by \( \mathcal{W}_n(x_n) \), as

\[
(f_n(x_n) - \sum_{r \in R_n} \int_{\omega_n(r)}^{\omega_n(r)+x_{nr}} \phi_r(u)du) + \sum_{r \in R_n} g_{nr}(x_{nr}),
\]

where \( \phi_r \) is a non-decreasing estimation of the marginal cost for the resource unit \( r \in R \) processing unit workload when the resource surplus \( u \in [0, C_r] \). We also define \( \phi_r(u) = +\infty \) when \( u > C_r \). The non-decreasing property profoundly reflects an underlying economic phenomenon, i.e., a thing is valued in proportion to its rarity. The later a job arrives, the higher cost it has to pay \( [22] \). The first component is the pseudo-utility of job \( n \), which is the utility of it minus the cost to pay. The second component is the revenue of the platform. If we organize \( \mathcal{W}_n(x_n) \) as

\[
f_n(x_n) + \sum_{r \in R_n} \left( g_{nr}(x_{nr}) - \int_{\omega_n(r)}^{\omega_n(r)+x_{nr}} \phi_r(u)du \right),
\]

the second component can be regarded as the net profit of the platform for serving job \( n \). In this case, the later a job arrives, the harder the resource surplus to meet its requirements, which results to higher cost. The following content applies to both of these two interpretations.

To bridge connections between the optimal dual variables of \( P_1 \) and the optimal solution \( x^*_{nr} \) that maximizes \( \mathcal{W}_n \), we firstly introduce the dual problem of \( P_1 \) as follows.

Proposition 1. The dual problem of \( P_1 \) is:

\[
P_2 : \min_{\mu, \lambda} \sum_{n \in N} \sum_{r \in R} \xi_{nr}(\mu_n + \lambda_r) + \sum_{n \in N} \mu_n g_n + \sum_{r \in R} \lambda_r C_r
\]

\[
s.t. \ [5], [7], \mu \geq 0, \lambda \geq 0,
\]

where

\[
\xi_{nr}(p) \triangleq \max_{x_{nr} \in [0, x_{nr}]} \left( f_n(x_{nr}) + \left( g_{nr}(x_{nr}) - p \cdot x_{nr} \right) \right),
\]

and \( \mu \triangleq \{\mu_n\}_{n \in N} \) and \( \lambda \subseteq \{\lambda_r\}_{r \in R} \) are the dual variables corresponding to \( [4] \) and \( [6] \), respectively.

Proof. The result is immediate with Lagrangian. We just omit the details here.

\( \xi_{nr}(p) \) is known as the convex conjugate of the fractional social welfare \( f_n + g_{nr} \). Taking a closer look at the conjugate \( f_n + g_{nr} \) and the pseudo social welfare \( \mathcal{W}_n(x_n) \), if we could find appropriate \( p^* \) and \( x^*_{nr} \), we can bridge their connection through

\[
\mathcal{W}_n(x^*_{nr}) \approx \sum_{r \in R_n} \xi_{nr}(p^*).
\]

Based on this, we can interpret \( p \) as the marginal cost for processing unit workload \( [22] \). We bridge the subtle connection between \( \xi_{nr} \) and \( \mathcal{W}_n \) in the following proposition, which is crucial for the design of OnSoCMax.

Proposition 2. \( \forall n \in N_r \in R \), when \( \phi_r(C_r) \geq \omega_r \), if \( x_n = \{x_{nr}\}_{r \in R_n} \) and \( Mathbf{\mu^*_n} \) are respectively the optimal primal and dual solutions to \( [4] \) of the following problem \( P_3 \):

\[
P_3 : \max \mathcal{W}_n(x_n)
\]

\[
s.t. \ [4], [5], [7],
\]

and (ii) the resource usage level \( \omega_r \) is updated by

\[
\begin{align*}
\omega_{n+1} &= \omega_n + x^*_{nr} \\
\omega_1 &= 0,
\end{align*}
\]

then, \( x^*_{nr} \) is also the optimal solution that maximizes \( \xi_{nr}(p) \) given \( \phi_r(\omega_{n+1}) + \mathbf{\mu^*_n} \).

Proof. By the definition of the non-decreasing marginal cost function \( \phi_r(\cdot) \), we can find that it is discontinuous at \( C_r \). Thus, when \( \phi_r(C_r) \geq \omega_r \), there must exist a resource usage level \( \omega_r \leq C_r \) such that \( \phi_r(\omega_r) = \omega_r \). Note that the function \( f_n + \sum_{r \in R} g_{nr} \) is non-decreasing and its derivative on \( r \) is not more than \( \omega_r \). Therefore, when the input of \( \phi_r \) is \( \omega_n + x_{nr} \), suppose \( \omega_n + x_{nr} \leq \omega_r \). Consequently, the derivative of the integral function

\[
\mathcal{\Phi}_r(x_{nr}) \triangleq \int_{\omega_n(r)}^{\omega_n(r)+x_{nr}} \phi_r(u)du
\]

is continuous, non-decreasing, and convex when \( x_{nr} \leq \omega_r - \omega_n \). The convexity is because \( \mathcal{\Phi}_r(\cdot) \), i.e., \( \phi_r \), is non-decreasing. Thus, \( P_3 \) is a convex optimization program and its optimal solution can be obtained through KKT conditions. Let us use \( x^*_{nr}, \mu^*_n, \gamma^*_n, \) and \( \zeta^*_n \) to denote the optimal primal and dual solutions of \( P_3 \) \( (\mu^*_n \to [4]) \) while \( \gamma^*_n \) and \( \zeta^*_n \) to the right part and left part of \( [7] \), respectively. The KKT conditions of \( P_3 \) are listed below:

\[
\begin{align*}
f_n^*(x_{nr}^*) + \frac{\beta_{nr}}{C_r} > \phi_r(\omega_{n+1}^*) + \mu^*_n \triangleq \gamma^*_n - \zeta^*_n \\
\gamma^*_n(x_{nr}^* - \bar{x}_{nr}) &= 0 \\
\zeta^*_n \cdot x_{nr}^* &= 0 \\
\mu^*_n \left( \sum_{r \in R_n} x_{nr}^* - g_n \right) &= 0.
\end{align*}
\]

With KKT conditions \( [13] \), we show that the optimal solution \( x^*_{nr} \) of \( P_3 \) simultaneously optimizes the conjugate \( \xi_{nr}(p) \) given \( p = \phi_r(\omega_{n+1}^*) + \mu^*_n \).

- **Case 1**: When \( f_n^*(x_{nr}) + \frac{\beta_{nr}}{C_r} > \phi_r(\omega_{n+1}^*) + \mu^*_n \), \( \mathcal{W}_n \) is an increasing function on \( r \) under \( [4] \). Thus, we have \( x^*_{nr} = \bar{x}_{nr} \), which leads to

\[
f_n^*(\bar{x}_{nr}) + \frac{\beta_{nr}}{C_r} > \phi_r(\omega_{n+1}^*) + \mu^*_n.
\]

\( [14] \) indicates that \( f_n^*(x_{nr}) + \left( g_{nr}(x_{nr}) - p \cdot x_{nr} \right) \) is monotone increasing in feasible region \([0, \bar{x}_{nr}]\) by setting \( p = \phi_r(\omega_{n+1}^*) + \mu^*_n \). Therefore, \( x^*_{nr} = \bar{x}_{nr} \) is argmax \( \mathcal{W}_n(x_{nr}) \) \( \forall \omega_{n+1} \in [0, \bar{x}_{nr}] \), which means the same \( x_{nr} \) maximizes both \( P_3 \) and the conjugate simultaneously given \( p = \phi_r(\omega_{n+1}^*) + \mu^*_n \). Thus, \( [16] \) holds.
• **Case II:** When \( f_{n ultimately, we have \( x^*_{n} = 0 \), which leads to
\[
f'_{n'}(0) + \frac{\beta_{n'}}{C_{n'}} < \Phi_{r}(\omega_{r}^{(n+1)}) + \mu_{n'} \tag{15}
\]
and \( \omega_{r}^{(n+1)} = \omega_{r}^{(n)} + 0 = \omega_{r}^{(n)} \). Analogously, \([15]\) means that \( f_{n'}(x_{n'}) + (g_{n'}(x_{n'}) - p' \cdot x_{n'}) \) is monotone decreasing in feasible region \([0, x_{n'}] \) given \( p = \Phi_{r}(\omega_{r}^{(n+1)})) + \mu_{n'} \). Therefore, \( x^*_{n'} = 0 \) is an argmax over \( x_{n'} \) \( \{ f_{n'}(x_{n'}) + (g_{n'}(x_{n'}) - p' \cdot x_{n'}) \} \), which also leads to (16).

**Case III:** When \( f_{n'}(x_{n'}) + \frac{\beta_{n'}}{C_{n'}} = \Phi_{r}(\omega_{r}^{(n+1)}) + \mu_{n'} \), \( x^*_{n'} \) is an maximum of \( f_{n'}(x_{n'}) + (g_{n'}(x_{n'}) - p' \cdot x_{n'}) \) given \( p = \Phi_{r}(\omega_{r}^{(n+1)})) + \mu_{n'} \) since \( \gamma_{n'} = \zeta_{n'}^{*} = 0 \). Thus, \( x^*_{n'} \) is in \( \argmax \{ f_{n'}(x_{n'}) + (g_{n'}(x_{n'}) - p' \cdot x_{n'}) \} \), which means (16) holds.

All the three conditions are visualized in Fig. 2.

This proposition leads to the follow equation
\[
\xi_{n'}(\Phi_{r}(\omega_{r}^{(n+1)}) + \mu_{n'}) = f_{n'}(x^*_{n'}) + g_{n'}(x^*_{n'}) - (\Phi_{r}(\omega_{r}^{(n+1)}) + \mu_{n'}) \cdot x_{n'}.
\tag{16}
\]

**Algorithm 1:** **OnSocMax** for the serverless platform

**Input:** \( \{ C_{n} \}_{n \in \mathcal{R}} \) and \( \{ g_{n} \}_{n \in \mathcal{N}, r \in \mathcal{R}} \)

**Output:** Online solution to \( P_{1} \) and final utilizations for the resource mesh. 

1. \( \forall r \in \mathcal{R} : \hat{\omega}_{r}^{(1)} \leftarrow 0 \)
2. while a new job \( n \) arrives do
3. Receive the quadruple \( \{ q_{n}, R_{n}, C_{n}, f_{n} \} \)
4. /* Solve the convex program \( P_{3} \) optimally */
5. for \( r \in \mathcal{R} \) do
6. Get \( x_{n'} \) with KKT conditions \([15]\) by
7. \( \hat{x}_{n'} \leftarrow \left\{ \begin{array}{ll}
\left( \begin{array}{c}
\xi_{n'} \\
0
\end{array} \right) & (a) \\
\left( f_{n'}(x_{n'}) + (g_{n'}(x_{n'}) - p' \cdot x_{n'}) \right) & (b) \\
\hat{\phi}_{r}(\omega_{r}) - \mu_{n'} - \frac{\beta_{n'}}{C_{n'}} & (c)
\end{array} \right. 
\right. 
\)
8. \( n \leftarrow n + 1 \)
9. end for
10. end while
11. return \( \{ x_{n} \}_{n \in \mathcal{N}} \) and \( \{ \hat{\omega}_{r}(\xi_{n'}(1)+1) \}_{r \in \mathcal{R}} \)

**Definition 1.** For any arrival instance \( A \) of all jobs \( n \in \mathcal{N} \), the competitive ratio for an online algorithm is defined as
\[
\alpha \triangleq \max_{\forall A} \frac{\Theta_{P}(A)}{\Theta_{on}(A)},
\tag{17}
\]
where \( \Theta_{P}(A) \) is the maximum objective value of \( P_{1} \), \( \Theta_{on}(A) \) is the objective function value of \( P_{1} \) obtained by this online algorithm.

Competitive ratio quantifies the worst-case ratio between the optimum and the objective obtained by the online algorithm. The smaller \( \alpha \) is, the better the online algorithm. An online algorithm is called \( \alpha \)-competitive if its ratio is upper bounded by \( \alpha \).

Now we give the requirements the marginal cost functions \( \{ \Phi_{r} \}_{r \in \mathcal{R}} \) should satisfy to guarantee that OnSocMax is \( \alpha \)-competitive for some \( \alpha \).

**Theorem 1.** (Extended from Theorem 4 of [22]) **OnSocMax** is \( \alpha \)-competitive for some \( \alpha \geq 1 \) if \( \forall r \in \mathcal{R} \), the marginal cost function \( \hat{\phi}_{r} \) is in the form of
\[
\hat{\phi}_{r}(\omega) = \left\{ \begin{array}{ll}
\hat{\varphi}_{r}(\omega) & \omega \in [\hat{\omega}_{r}, C_{r}] \\
+\infty & \omega \in (C_{r}, +\infty)
\end{array} \right. 
\tag{18}
\]
where \( \hat{\omega}_{r} \) is a resource utilization threshold, and \( \hat{\varphi}_{r} \) is a non-decreasing function that satisfies
\[
\left\{ \begin{array}{ll}
\hat{\varphi}_{r}(\omega) & \leq 0 \int_{0}^{\omega} \hat{\varphi}_{r}(u) du - \lambda \cdot \omega, \quad \omega \in [\hat{\omega}_{r}, C_{r}] \\
\hat{\varphi}_{r}([\hat{\omega}_{r}, +\infty)) & = \lambda, \hat{\varphi}_{r}(C_{r}) > \nu
\end{array} \right. 
\tag{19}
\]

**Proof.** To prove this result, we refer to the technique named instance-dependent online primal-dual approach, proposed in [32]. The key idea is to construct a dual solution to \( P_{3} \) based on the solution \( \{ \tilde{x}_{n} \}_{n \in \mathcal{N}} \) produced by OnSocMax. Then, it
use this dual objective to build the upper bound of the optimum of $\mathcal{P}_3$ based on weak duality. When building the upper bound, this technique studies the worst-case instances under different scenarios.

Let us use $\mathcal{B} \triangleq \{A_1, A_2, \ldots\}$ to denote the set of arrival instances of all jobs, and use $\Theta_{\mathcal{P}_2}(\mathcal{A})$ to denote a feasible objective value of the dual problem $\mathcal{P}_2$ for any arrival instance $\mathcal{A}$. Hereinafter, we just replace $\omega_r^{(n)}$ by $\hat{\omega}_r^N$ for simplification. We divide $\mathcal{B}$ into three disjoint sets:

$$
\begin{align*}
\mathcal{B}_1 & \triangleq \{ A \mid 0 \leq \hat{\omega}_r^N < \hat{\omega}_r, \forall r \in \mathcal{R} \} \\
\mathcal{B}_2 & \triangleq \{ A \mid \hat{\omega}_r \leq \hat{\omega}_r^N \leq C_r, \forall r \in \mathcal{R} \} \\
\mathcal{B}_3 & \triangleq \mathcal{B} \setminus (\mathcal{B}_1 \cup \mathcal{B}_2). 
\end{align*}
$$

(20)

$\mathcal{B}_1$ and $\mathcal{B}_2$ contain the instances whose final utilizations of all resource units in the mesh are below and above the threshold $\hat{\omega}_r$, respectively. Our goal is to prove that, under the conditions (18) and (19), $\forall \mathcal{A} \in \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ respectively, the following relations hold:

$$
\alpha \cdot \Theta_{on}(\mathcal{A}) \geq \Theta_{\mathcal{P}_2}(\mathcal{A}) \geq \Theta^*_p(\mathcal{A}).
$$

(21)

In the following analysis, we just drop the parentheses and $\mathcal{A}$ for simplification.

**Case I:** $\forall \mathcal{A} \in \mathcal{B}_1$, from (18) we can find that the marginal costs experienced by all jobs are the same, i.e., $\iota$. In this case, each job $n$ is processed with maximum permitted rate $\hat{\tau}_n$ on $r \in \mathcal{R}_n$. Thus, $\Theta_{\mathcal{P}_2}(\mathcal{A}) = \Theta_{on}(\mathcal{A}) = \Theta^*_p(\mathcal{A}).$

**Case II:** $\forall \mathcal{A} \in \mathcal{B}_2$, we construct a feasible dual solution to $\mathcal{P}_2$ as

$$
\begin{align*}
\mu^*_n &= \hat{\mu}_n, \\
\lambda_r &= \hat{\lambda}_r.
\end{align*}
$$

(22)

where $\mu^*_n$ is the optimal dual solution to $\mathcal{P}_3$ introduced by (13). Let $p \geq p' \geq 0$ and denote the optimal solution that maximizes the conjugate $\xi_{nr}(p)$ by $\hat{x}_{nr}$ given $p$. Then,

$$
\begin{align*}
\xi_{nr}(p) &= f_{nr}(\hat{x}_{nr}) + (g_{nr}(\hat{x}_{nr}) - p \cdot \hat{x}_{nr}) \\
&\leq f_{nr}(\hat{x}_{nr}) + (g_{nr}(\hat{x}_{nr}) - p' \cdot \hat{x}_{nr}) \\
&\leq \max_{\hat{x}_{nr}} \left[ f_{nr}(\hat{x}_{nr}) + (g_{nr}(\hat{x}_{nr}) - p' \cdot \hat{x}_{nr}) \right] \\
&= \xi_{nr}(p').
\end{align*}
$$

(23)

which indicates that the conjugate $\xi_{nr}(p)$ is non-increasing with $p$. The above derivation uses the fact that $f_{nr} + g_{nr}$ is non-decreasing. Based on weak duality and the non-increasing property of the conjugate, we have

$$
\Theta_{\mathcal{P}_2} \leq \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \xi_{nr} \left( \mu^*_n + \hat{\phi}_r(\hat{\omega}_r^N) \right) + \sum_{n \in \mathcal{N}} \mu^*_n \hat{\theta}_n + \sum_{r \in \mathcal{R}} \hat{\phi}_r(\hat{\omega}_r^N) \hat{C}_r \\
+ \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \xi_{nr} \left( \mu^*_n + \hat{\phi}_r(\hat{\omega}_r^{(n+1)}) \right) + \sum_{n \in \mathcal{N}} \mu^*_n \hat{\theta}_n + \sum_{r \in \mathcal{R}} \hat{\phi}_r(\hat{\omega}_r^N) \hat{C}_r \\
\leq \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \xi_{nr} \left( \mu^*_n + \hat{\phi}_r(\hat{\omega}_r^{(n+1)}) \right) + \sum_{n \in \mathcal{N}} \mu^*_n \hat{\theta}_n + \sum_{r \in \mathcal{R}} \hat{\phi}_r(\hat{\omega}_r^N) \hat{C}_r \\
&\leq \sum_{r \in \mathcal{R}} \left( \hat{\phi}_r(\hat{\omega}_r^N) \hat{C}_r - \sum_{n \in \mathcal{N}} \hat{\phi}_r(\hat{\omega}_r^{(n+1)}) \hat{x}_{nr} \right) \\
&\leq \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \triangleq \Theta_{\text{tmp}}.
$$

(24)

The above derivation uses the fact that $\hat{\omega}_r^{(n+1)}$ is non-increasing with $n$. Besides, from Fig. 2 we can find that $\xi_{nr}(\mu^*_n + \hat{\phi}_r(\hat{\omega}_r^{(n+1)})) \geq 0$ holds for all jobs. Thus, based on (25), we have

$$
\sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \leq \int_0^{\hat{\omega}_r^N} \hat{\phi}_r(u) du.
$$

(26)

Based on the above relations (25) and (26), we have

$$
\Theta_{\text{tmp}} \leq \sum_{r \in \mathcal{R}} \left( \hat{\phi}_r(\hat{\omega}_r^N) \hat{C}_r - \int_0^{\hat{\omega}_r^N} \hat{\phi}_r(u) du \right) \\
+ \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \triangleq \Theta_{\text{tmp}}.
$$

(25)

$$
< \sum_{r \in \mathcal{R}} (\alpha - 1) \int_0^{\hat{\omega}_r^N} \hat{\phi}_r(u) du \quad \triangleright \quad \text{drop } (\alpha - 1) \cdot \hat{\omega}_r^N
$$

\& (19)

$$
+ \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \alpha. \quad \triangleright \quad \text{drop } (\alpha - 1) \cdot \hat{\omega}_r^N
$$

(26)

The final expression is exactly $\alpha \cdot \Theta_{on}$. Thus, $\Theta_{\mathcal{P}_2}/\Theta_{on} < \alpha$.

**Case III:** $\forall \mathcal{A} \in \mathcal{B}_3$, we define two disjoint sets to split the resource mesh $\mathcal{R}$:

$$
\begin{align*}
\mathcal{R}_1 & \triangleq \{ r \in \mathcal{R} \mid 0 \leq \hat{\omega}_r^N < \hat{\omega}_r, \forall r \in \mathcal{R} \} \\
\mathcal{R}_2 & \triangleq \{ r \in \mathcal{R} \mid \hat{\omega}_r \leq \hat{\omega}_r^N \leq C_r, \forall r \in \mathcal{R} \}.
\end{align*}
$$

(27)
For resource unit \( r \) in different sets, the corresponding dual variables are constructed in different ways. We extend \( P_1 \) to \( P'_1 \) by adding the following constraint:

\[
\sum_{n \in N} \sum_{r \in R_1} x_{nr} \leq \sum_{r \in R} \hat{\phi}_r^N. \tag{28}
\]

Apparently, \( P'_1 \) is the same as \( P_1 \) for OnSocMax since (28) is not violated by \( \{ \hat{x} \}_{n \in N} \). The dual problem \( P'_2 \) to \( P'_1 \) is

\[
P'_2: \min_{\mu, \lambda} \sum_{n \in N} \left[ \sum_{r \in R_1} \xi_{nr}(\mu_n + \lambda_r + \delta) + \sum_{r \in R_2} \xi_{nr}(\mu_n + \lambda_r) \right] + \sum_{n \in N} \mu_n \theta_n + \sum_{r \in R} \lambda_r C_r + \delta \sum_{r \in R} \hat{\phi}_r^N
\]

\[s.t. \quad (5), (7), \mu_n \geq 0, \lambda_r \geq 0, \delta \geq 0, \]

where \( \delta \) is the dual variable corresponding to the newly added constraint (28). Then, we construct the dual solution to \( P'_2 \) as

\[
\hat{\lambda}_r = \begin{cases} 
0 & \forall r \in R_1 \\
\hat{\phi}_r(\hat{\omega}^N_r) & \forall r \in R_2
\end{cases}
\]

\[
\delta = \frac{l}{\alpha}, \quad \mu_n = \mu^*_n \quad \forall n \in N. \tag{30}
\]

Based on (30), we can follow a similar approach as shown in Case II to obtain that \( \Theta^*_{\text{soc}} / \Theta_{\text{on}} \leq \alpha \). A slight difference is that, in Case III, when applying (19) to \( \Theta'_{\text{tmp}} \), the result is tightly bounded.

Theorem 1 extends the Two-point Boundary Value ODEs for designing the marginal cost functions from standard 0-1 knapsack problem to multi-dimensional fractional problems. Based on Theorem 1 and Gronwall’s Inequality [37], we have the detailed design of \( \{ \hat{\phi}_r \}_{r \in R} \), which is irrelevant with the utilities \( \{ f_n \}_{n \in N} \) and \( \{ \theta_{nr} \}_{n \in N, r \in R} \), as follows.

Theorem 2. \( \forall r \in R_n \), if the marginal cost function \( \hat{\phi}_r \) used in step 6-condition (c) of OnSocMax is designed as

\[
\hat{\phi}_r(\omega) = \begin{cases} 
\frac{t}{\exp(\xi) - \exp(\frac{\alpha \xi}{\alpha - 1})} \frac{\omega}{\alpha - 1} & \omega \in [0, \hat{\omega}_r] \\
\frac{t}{\exp(\xi) - \exp(\frac{\alpha \xi}{\alpha - 1})} e^{\frac{\alpha \xi}{\alpha - 1}} \frac{\omega}{\alpha - 1} & \omega \in (\hat{\omega}_r, C_r] \\
\frac{t}{\exp(\xi) - \exp(\frac{\alpha \xi}{\alpha - 1})} \frac{\omega}{\alpha - 1} & \omega \in (C_r, +\infty)
\end{cases}
\]

where \( \hat{\omega}_r = \frac{C_r}{\alpha - 1} \), then (i) OnSocMax is \( \hat{\alpha} \)-competitive, where \( \hat{\alpha} \) is the solution of

\[
\hat{\alpha} - 1 = \frac{1}{\hat{\alpha} - 1} + \ln \left( \frac{\hat{\alpha} - 1}{\hat{\alpha} - 1} \right) \tag{31}
\]

and (ii) when \( \hat{\alpha} \geq \frac{1}{2} + 1 \), the gap between \( \hat{\alpha} \) and the optimal competitive ratio when \( |R| = 1 \) is at least \( \frac{2}{\sqrt{\pi} - 1} \ln \frac{\sqrt{\pi} - 1}{\sqrt{\pi} - 1} \approx 0.1368 \).

Proof. We firstly introduce the Gronwall’s inequality [37] as follows. \( \forall x \in [\xi, \theta] \), if \( f(x) \leq a(x) + b(x) \int_x^\theta f(u)du \), then

\[
f(x) \leq a(x) + b(x) \int_x^\theta a(u) \left( \int_u^\theta b(w)dw \right)du, \tag{32}
\]

where \( f(x) \) is continuous, \( a(x) \) and \( b(x) \) are integrable and \( \forall x \in [\xi, \theta], b(x) \geq 0 \). The result remains valid if all the \( \leq \) are replaced by \( = \). Applying (32) to (19) leads to

\[
\begin{align*}
t \leq \hat{\phi}_r(\hat{C}_r) & \leq \frac{t}{\alpha} + \left( \frac{\hat{\phi}_r(\hat{C}_r - 1)}{C_r} - \frac{t}{\alpha} \right) e^\left( \frac{\alpha \hat{C}_r - 1}{\alpha - 1} \right). \tag{33}
\end{align*}
\]

Thus, the minimum \( \hat{\alpha} \) is achieved when all inequalities in (19) and (33) are binding, which leads to the design of \( \{ \hat{\phi}_r \}_{r \in R} \) and the competitive ratio achieved by (31).

In the following, we prove the results of (ii). When \( R = 1 \), \( P_1 \) degenerates to the general one-way trading (GOT) problem [38]. The optimal competitive ratio is proved to be \( 1 + \ln(\frac{\omega}{\lambda}) \) [22], (31), (32), (34), (38). With \( \hat{\alpha} \geq 1, \frac{\lambda}{\omega} \geq 1 \), let us take \( y \geq 1 \) as a substitute for \( \alpha - 1 \). Then

\[
\hat{\alpha} - 1 - \ln \left( \frac{\omega}{\alpha} \right) = y - \ln \left( \frac{\omega}{\alpha} \right) \quad \Rightarrow \text{with (31)}
\]

\[
= \ln \left( y + 1 - \frac{\lambda}{\omega} \right) + \frac{1}{y} - \ln y \leq \text{gap}(y).
\]

Applying \( \ln(x) \leq x - 1, \forall x \geq 1 \) to the logarithm in \( \text{gap}(y) \), we have \( \text{gap}(y) \leq y + \frac{1}{y} - \ln y - \frac{1}{\alpha} \). By analyzing the upper bound of \( \text{gap}(y) \), we can easily find that when \( y^* = \sqrt{\frac{\alpha + 1}{2}} \), its upper bound is at least \( \sqrt{\frac{1}{2}} - \ln y^* \), which directly leads to the result in (ii).

By the design of \( \hat{\phi}_r(\cdot) \), we observe that \( \hat{\alpha} \geq 2 \) holds because \( \frac{\omega}{\lambda} \geq 1 \). Unsurprisingly, OnSocMax has a linear complexity of \( O(|N| \cdot |R|) \) when \( \{ f_{nr} \}_{n \in N, r \in R} \) are linear and share the same coefficient. In this case, \( \hat{\alpha} = 2 \).

C. Extending to Non-fractional Workloads

OnSocMax can be applied to jobs whose workloads are not permitted to be fractioned. Specifically, in this case, (7) is replaced by

\[
x_{nr} \in \{0, x_{nr}\}, \forall n \in N, r \in R_n, \tag{34}
\]

where \( \overline{x}_{nr} = \theta_{nr} \). To solve the new problem in online settings, we can approximate the marginal cost defined in (12) with \( \hat{\phi}_r(\omega^{(n)} + \beta_{nr}) \overline{x}_{nr} \). With this substitution, the conditions (c) in step 6 of OnSocMax is merged into condition (a) or (b), and OnSocMax achieves the same competitive ratio as shown in Theorem 2. Detailed proof for this result is omitted because of space constraints. This approach is exactly the implementation of [10].
D. Implementation Concerns

In our model, jobs are processed by invoking cloud functions across different resource units, which relies on the multi-tenant hardware sharing technique such as VM-like isolation. The approach adopted by AWS Lambda is maintaining an active pool of computing instances that have been used to run functions beforehand and are maintained to serve future invocations. Besides, considering that the workloads of one job are dispatched to different resource unit across a time window, an efficient communication mechanism is required. Take the video transcoding with ExCamera as an example, it uses a long-running VM-based rendezvous server, facilitated with a coordinator, to relay packets between cloud functions. Based on these techniques, OnSocMax can be easily plugged in the API gateway and triggered at the beginning of each time slot.

IV. EXPERIMENTAL VALIDATION

A. Experimental Setup

Jobs and Computing Instances. We consider a cluster with 10 computing instances in the time horizon of 24 time slots. The processing capacity of computing instances are generated from an i.i.d. Gaussian $N(\mu = 20, \sigma = 2)$. By setting $\tau$ as 60 minutes, the time horizon represents one day. We set the total number of transcoding jobs as 20. The number of job arrivals in each time slot follows a Poisson distribution with a mean of 2.03 jobs, which is independent of other time slots in this day. The deadline of each job is calculated based on the arrive time and the maximum service duration of it, where the latter is generated from an exponential distribution with a mean of 4 time slots (2 hours). Each job has a workload whose size is generated from a Normal distribution $N(\mu = 18, \sigma = 3)$. The parallelism bound of each job is generated from a Normal distribution $N(\mu = 7, \sigma = 1)$.

Utilities and Pricing Parameters. The utility of job $n$ is set as a zero-startup, non-decreasing concave function. We study $f_{nr}$ in three cases: linear function, logarithmic function, and polynomial function. Specifically, for each $n \in N$, $r \in R_n$,

$$f_{nr}(x) = \begin{cases} 
ax \quad \text{linear} \\
\log(x+1) \quad \text{log} \\
\sqrt{x} \quad \text{poly},
\end{cases}$$

where the coefficient $a$ is generated from a uniform distribution in $[1, 3]$. Similarly, the pricing parameter $\beta_{nr}$ in $g_{nr}(\cdot)$ is generated from the uniform distribution in $[0.1, 0.5]$.

Algorithms Compared. We compare OnSocMax with two handcrafted online algorithms. The first algorithm is called Max-First, where each computing instance always serves the job with highest myopic social welfare, i.e., the sum of the utility of the job and the revenue for serving it in each time slot (subject to the processing capacity of instances and parallelism bound of jobs). The second is called Equal-Share, where each computing instance serves each job with equal opportunity within capacity limits.

B. Simulation Results

Effectiveness. Fig. 4 shows the social welfare achieved of three algorithms under different service duration and parallelism bound settings. We can observe that, overall, all the algorithms achieve higher social welfare when the service duration and parallelism bound increase. The reason is that, when the capacity of computing instances are sufficient, increasing the service duration and the parallelism bound of jobs can increase the opportunities that jobs been fully served. In spite of this, OnSocMax always performs the best among these online algorithms.

OnSocMax performs the best for linear utility functions. This is because the logarithmic and polynomial functions have diminishing returns, which could increase the fluctuation ratio $\frac{\omega}{\eta}$. This will cause more jobs served with their marginal costs fall into the second segment of $\hat{\phi}(\omega)$, which further leads to the decrease of social welfare.

Robustness. We verify the robustness of OnSocMax under different settings of service capacity of computing instances and service demand of jobs. These two variables actually tune the congestion level, i.e., the coverage rate of service demands, from different angles. We can find that OnSocMax is robust to the changes of the congestion level from Fig. 5.

V. RELATED WORKS

Online load balancing policies are fully investigated under classic settings, where multiple identical servers with exponentially distributed service rates process continuous arrived jobs. Based on CTMC and Lyapunov Stability theories, load balancing policies such as JSQ, JIQ, Pod, and JFIQ are proposed and analyzed on the mean response time and cross-server communication overhead. In a most recent work, Weng et al. proposed the JFSQ and JFIQ policies under the constraints of heterogenous service rates and service locality. They show that, under a well-connected bipartite graph condition, these two policies can achieve the minimum mean response time in both the many-server regime and the sub Halfin-Whitt regime.

Another line of works study the energy-efficient and multi-resource sharing load balancing from a different theoretical basis. Generally, the objective is to improve the energy efficiency with on-demand resource allocation. Thereinto, online load balancing of deadline-sensitive jobs have been studied in [6], [15], [41]. In a similar work, the authors design online algorithms for both fractional and non-fractional workload model under concave utility settings. The optimality of designed algorithms hold when all the jobs have the same deadline and share a single type of resource. Compared with it, our work is more general with the concept of resource mesh and the technique of marginal cost estimation, which makes it more suitable for serverless computing.

Online load balancing under the objective of minimizing the Nash Social Welfare is revisited in a recent paper [17]. This work provides tight bounds on the price of anarchy (PoA) of pure Nash equilibria and on the competitive ratio of the general greedy algorithm under very general latency functions.
In this paper, we study the online load balancing problem for serverless with a target of maximizing the social welfare. The job we considered is continuous arriving, deadline-sensitive, serverless with a target of maximizing the social welfare. The algorithm is proved to be \( \alpha \)-competitive for an \( \alpha \) at least 2. Online load balancing for jobs with complex workflows and communication patterns will be studied in future.

VI. Conclusion

In this paper, we study the online load balancing problem for serverless with a target of maximizing the social welfare. The job we considered is continuous arriving, deadline-sensitive, and has maximum parallelism bound. To take both the spatio and temporal resource into consideration, we establish a model of resource mesh. Every job can only be dispatched to the resource units available to it. Based on the marginal cost estimation technique, we design an online algorithm OnSocMax by following the solutions of several convex pseudo-social welfare maximization problems. The algorithm is proved to be \( \alpha \)-competitive for an \( \alpha \) at least 2. Online load balancing for jobs with complex workflows and communication patterns will be studied in future.

References

[1] M. S. Aslanpour, A. N. Toosi, C. Cicconetti, B. Javadi, P. Sharski, D. Taibi, M. Assuncao, S. S. Gill, R. Gaire, and S. Dustdar, “Serverless edge computing: Vision and challenges,” in 2021 Australasian Computer Science Week Multiconference, ser. ACSW ’21, 2021.

[2] R. Buyya, S. N. Siramala, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojevic, C. Varela, R. Bahsoon, M. D. D. Assuncao, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, and H. Shen, “A manifesto for future generation cloud computing: Research directions for the next decade,” ACM Comput. Surv., vol. 51, no. 5, Nov. 2018.

[3] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Mothsamy, R. Rabah, A. Slominski, and P. Suter, Serverless Computing: Current Trends and Open Problems. Singapore: Springer Singapore, 2017, pp. 1–20.

[4] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift, “Peeking behind the curtains of serverless platforms,” in 2018 USENIX Annual Technical Conference (USENIX ATC 18), Jul. 2018, pp. 133–146.

[5] M. A. Adnan, R. Sugihara, and R. K. Gupta, “Energy efficient geographical load balancing via dynamic deferral of workload,” in 2012 IEEE Fifth International Conference on Cloud Computing, 2012, pp. 188–195.

[6] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, “Greening geographical load balancing,” IEEE/ACM Transactions on Networking, vol. 23, no. 2, pp. 657–671, 2015.

[7] J. Luo, L. Rao, and X. Liu, “Temporal load balancing with service delay guarantees for data center energy cost optimization,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 775–784, 2014.

[8] S. Moharir, S. Sanghavi, and S. Shakkottai, “Online load balancing under graph constraints,” in Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, ser. SIGMETRICS ’13, 2013, p. 363–364.

[9] W. Weng, X. Zhou, and R. Sitaraman, “Optimal load balancing in bipartite graphs,” arXiv preprint arXiv:2008.08830, 2020.

[10] E. Anton, U. Ayesta, M. Jonckheere, and I. M. Verloop, “Improving the performance of heterogeneous data centers through redundancy,” Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 3, Nov. 2020.

[11] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, “Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services,” Perform. Eval., vol. 66, no. 11, p. 1056–1071, Nov. 2011.

[12] D. Mukherjee, S. C. Borst, J. S. Van Leeuwaarden, and P. A. Whiting, “Universality of power-of-d load balancing in many-server systems,” Stochastic Systems, vol. 8, no. 4, pp. 265–292, 2018.

[13] D. Rutten and D. Mukherjee, “Load balancing under strict compatibility constraints,” 2020.

[14] R. R. Weber, “On the optimal assignment of customers to parallel servers,” Journal of Applied Probability, pp. 406–413, 1978.

[15] Z. Zheng and N. B. Shroff, “Online multi-resource allocation for deadline-sensitive jobs with partial values in the cloud,” in IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[16] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, “Fog computing for energy-aware load balancing and scheduling in smart factory,” IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4548–4556, 2018.

[17] V. Bilò, G. Monaco, L. Moscardelli, and C. Vinci, “Nash social welfare in selfish and online load balancing,” in Web and Internet Economics, X. Chen, N. Gravin, M. Hoefer, and R. Mehta, Eds. Cham: Springer International Publishing, 2020, pp. 323–337.

[18] S. Fouladi, R. S. Wahby, B. Schacklett, K. V. Balasubramaniam, W. Zeng, R. Bhalauro, A. Sivaraman, G. Porter, and K. Weinstein, “Encoding, fast and slow: Low-latency video processing using thousands of tiny threads,” in 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Mar. 2017, pp. 363–376.

[19] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar et al., “Cloud programming simplified: A Berkeley view on serverless computing,” arXiv preprint arXiv:1902.03383, 2019.

[20] D. P. Bertsekas, “Nonlinear programming,” Journal of the Operational Research Society, vol. 48, no. 3, pp. 334–334, 1997.

[21] J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, “Load balancing in data center networks: A survey,” IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2324–2352, 2018.

[22] X. Tan, B. Sun, A. Leon-Garcia, Y. Wu, and D. H. Tsang, “Mechanism design for online resource allocation: A unified approach,” Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 2, Jun. 2020.

[23] H. Zhao, S. Deng, Z. Liu, Z. Xiang, J. Yin, S. Dustdar, and A. Zomaya, “Dopso: Decentralized, privacy-preserving, and low-complexity online slicing for multi-tenant networks,” IEEE Transactions on Mobile Computing, pp. 1–1, 2021.

[24] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, “An axiomatic theory for energy-aware load balancing and scheduling in smart factory,” IEEE Transactions on Industrial Informatics, vol. 4, no. 2, Jun. 2020.

[25] X. Chen, N. Gravin, M. Hoefer, and R. Mehta, Eds. Cham: Springer International Publishing, 2020, pp. 323–337.

[26] E. Anton, U. Ayesta, M. Jonckheere, and I. M. Verloop, “Improving the performance of heterogeneous data centers through redundancy,” Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 3, Nov. 2020.

[27] Alibaba Cloud, “Apsaravideo for media processing,” https://www.aliyun.com/products/video/20191223/157229.html.

[28] Google Cloud, “Pricing details for the transcoder api,” https://cloud.google.com/transcoder/pricing, 2021.

[29] Amazon Web Services, Inc, “Amazon elastic transcoder pricing,” https://aws.amazon.com/elastictranscoder/pricing/, 2021.

[30] Alibaba Cloud, “Apsaravideo for media processing,” https://www.ali巴巴.com/product/mts/pricing, 2021.
[28] Tencent Cloud, “Video processing pricing,” [https://cloud.tencent.com/product/mps/pricing] 2021.

[29] J. Pachinger, G. R. Raidl, and U. Pferschy, “The multidimensional knapsack problem: Structure and algorithms,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250–265, 2010.

[30] Z. Zhang, Z. Li, and C. Wu, “Optimal posted prices for online cloud resource allocation,” *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, Jun. 2017.

[31] Y. Zhou, D. Chakrabarty, and R. Lukose, “Budget constrained bidding in keyword auctions and online knapsack problems,” in *Internet and Network Economics*, C. Papadimitriou and S. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 566–576.

[32] B. Sun, A. Zeynali, T. Li, M. Hajiesmaili, A. Wierman, and D. H. 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, Nov. 2020.

[33] Z. Zheng and N. Shroff, “Online welfare maximization for electric vehicle charging with electricity cost,” in *Proceedings of the 5th International Conference on Future Energy Systems*, 2014, p. 253–263.

[34] X. Tan, A. Leon-Garcia, Y. Wu, and D. H. K. Tsang, “Online combinatorial auctions for resource allocation with supply costs and capacity limits,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 655–668, 2020.

[35] L. Yang, M. H. Hajiesmaili, and W. S. Wong, “Online linear programming with uncertain constraints : (invited paper),” in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, 2019, pp. 1–6.

[36] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 2005.

[37] D. S. Mitrinovic, J. Pecaric, and A. M. Fink, *Inequalities involving functions and their integrals and derivatives*. Springer Science & Business Media, 2012, vol. 53.

[38] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, “Optimal search and one-way trading online algorithms,” *Algorithmica*, vol. 30, no. 1, pp. 101–139, 2001.

[39] T. A. Wagner, “Acquisition and maintenance of compute capacity,” Jun. 23 2020, uS Patent 10,691,498.

[40] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, “Computational load balancing on the edge in absence of cloud and fog,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1499–1512, 2019.

[41] B. Lucier, I. Menache, J. S. Naor, and J. Yaniv, “Efficient online scheduling for deadline-sensitive jobs: Extended abstract,” in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, 2013, p. 305–314.

[42] G. Aumala, E. Boza, L. Ortiz-Avilés, G. Totoy, and C. Abad, “Beyond load balancing: Package-aware scheduling for serverless platforms,” in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2019, pp. 282–291.