Spatio-Temporal Resource Meshes for Serverless Computing

Hailiang Zhao, Shuiguang Deng, Senior Member, IEEE, Jianwei Yin, Schahram Dustdar, Fellow, IEEE, and Albert Y. Zomaya, Fellow, IEEE

Abstract—Serverless computing is leading the way to a simplified and general purpose programming model for the cloud. A key enabler behind serverless computing is efficient load balancing, which routes function invocations to appropriate backend VM nodes. However, current load balancing algorithms implemented in serverless platforms are static and simple to operate without advantageous performance guarantee. Although policies such as Pod or JFIQ yield asymptotically optimal average response time, the information they depend on are usually unavailable. In addition, dispatching serverless functions with strict deadlines to heterogenous VM nodes with limited input data processing speeds online is difficult because the resource contention is intensified with frequently creating and evicting of function instances. To design an online load balancing algorithm without assumptions on the arrival patterns of serverless functions and the processing capacity of VM nodes while maximizing the social welfare, i.e., the sum of utility of functions and the revenue of the serverless platform, we construct several pseudo-social welfare functions and marginal cost functions, where the latter is to estimate the marginal cost for provisioning resources to every newly arrived function based on present resource surplus. The proposed algorithm, named OnSocMax, works by following the solutions of several convex optimization problems. It is proved to be $\alpha$-competitive for some $\alpha$ at least 2. We also validate the theoretical superiority of OnSocMax with simulations and the results show that it significantly outperforms two handcrafted benchmarking policies on the social welfare.

Index Terms—Serverless computing, social welfare maximization, load balancing, online algorithm.

1 INTRODUCTION

Today’s cloud computing is a paradigm of computing as a utility [1]. To reduce the tedious process to configure VM-based machines while improving the Quality of Experience (QoE) of end users, a new computation paradigm, named serverless computing, emerged and has enormous potential to be the dominant way of building elastic cloud services [2], [3]. Serverless allows end users to write and deploy their code on a flexible pay-as-you-go basis without the hassle of worrying about the underlying infrastructure. It erases the heavy burden to use the cloud by handling the system administration operations virtually, such as the installation of operating systems, libraries, runtime dependencies, and binaries. The prominent technical enablers of serverless are on-demand resource provisioning and seamless auto-scaling, with the help of efficient load balancing algorithms [2], [3] and multi-tenant resource isolation in different hierarchy [4]. Load balancing algorithms are carefully designed to dispatch the uploaded code and fetched data (abstracted as functions) to selected backend resources, with a target of optimizing the QoE or system efficiency [5], [6], [7], [8]. Meanwhile, auto-scaling techniques are developed to monitor the functions and automatically adjust resource provisioning to maintain steady, predictable performance at the lowest possible cost.

Optimal load balancing is a classic problem and has been extensively studied in distributed Web service systems [7], [8], [9], [10], [11]. For practical serverless scenarios, a fundamental challenge is that backend resource provisioning decisions should be made online without the knowledge of future function invocation requests. Besides, serverless functions are stateless during their execution and may experience the infrastructure freeze-thaw life cycle, which could lead to frequently creating and evicting of function instances on different VM nodes [12], [13], [14]. Therefore, to ensure robustness, load balancing policies implemented in current Kubernetes native serverless platforms, such as RoundRobin, SessionAffinity, Fairness, LeastConnections, amongst others, are static and simple to operate, but they offer no advantageous performance guarantee. Meanwhile, a series of academic studies on general online load balancing problems, which promise long-term performance guarantees, assume that function invocation requests (or jobs, etc) arrive according to Poisson process and service rates of VM nodes are exponentially distributed [9], [13], [16], [17], [18]. Under stochastic ordering assumption, policies such as Join-the-Shortest-Queue (JSQ) [19], Join-the-Idle-Queue (JIQ) [9], Power-of-$d$-Choices (Pod) [17], and Join-the-Fastest-of-the-Shortest-Queues (JFIQ) [15] are raised and analyzed based on the Continuous-Time Markov Chains (CTMCs) and Lyapunov Stability theories. However, their performance guarantees (mostly on the average response time) are established on sufficient assumptions, which are tough to satisfy in pro-
duction systems. If we take service locality\footnote{2} strict deadlines of functions, limited input data processing capacity of VM nodes, and the other unique characteristics of serverless platforms mentioned above into account, the performance guarantees are even harder to achieve. Another problem that cannot be ignored is that these algorithms operate in a First-Come-First-Serve manner with size-limited buffered queues and function preemption is not allowed. The other series of works concentrate on energy-efficient geographical load balancing\cite{5,6,7,20,21}. Their intention is either reducing non-renewable energy consumption or increasing energy efficiency of distributed VM nodes with inexpensive approaches for enabling large-scale on-demand response. However, in addition to the unpredictable function arrivals, their performance further suffers from resource contention if function instances of different importance and urgency are co-located on heterogenous servers\cite{13,20}.

Online load balancing of serverless functions with strict deadlines and limited input data processing speed on heterogenous resource units have been considered mostly from the perspective of either end user utility maximization, or the revenue of the serverless platform\cite{2,12,13}. 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 utility of functions and the revenue of the serverless platform are maximized simultaneously. To fill the gap, this paper studies the online load balancing problem for serverless functions with strict deadlines and limited input data processing speed by maximizing the sum of utilities of both functions 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.\footnote{2} 2.1. With the service locality constraint, each arrived function with fractional input data can only be dispatched to its available resource units. We design an algorithm, OnSocMax, to decide how the function invocation is scheduled and its input data is partitioned under the above mentioned constraints in serverless platforms. OnSocMax works by solving several well-designed pseudo-social welfare maximization problems online, and has no assumptions on the arrival pattern of serverless functions and service rates of VM nodes. We provide rigorous analysis to show that OnSocMax is $\alpha$-competitive for some $\alpha \geq 2$. Our main contributions are summarized as follows:

- We first establish the resource mesh model for serverless and study the online social welfare maximization problem for serverless functions with strict deadlines and limited input data processing speed. Our model approximates the real-world serverless systems and has assumptions only on the analytic properties of utilities.
- We propose an online algorithm OnSocMax which yields a competitive ratio at least 2 for general utility settings. Particularly, it has a polynomial complexity when the utility of serverless functions are linear and share the same coefficient. The theoretical superiority of OnSocMax is validated with simulations and the results show that it significantly outperforms baselines.

The rest of this paper is organized as follows. We formally introduce the system model and formulate the problem in Sec.\footnote{2} 2. We present the design details of the online algorithm OnSocMax with sufficient theoretical analysis in Sec.\footnote{3} 3. We present the numerical results in Sec.\footnote{4} 4 and discuss related work in Sec.\footnote{5} 5. Finally, we conclude this paper in Sec.\footnote{6} 6.

## 2 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 VM nodes distributed across different data-centers and CDN sites geographically, which are managed by the containerized orchestration framework Kubernetes. Let us use $K$ to denote the set of VM nodes and index each of them by $k$. Key notations used in this paper is summarized in Table\footnote{1} 1.

| Notation | Description |
| --- | --- |
| $\mathcal{K}$ | The set of VM nodes |
| $k \in \mathcal{K}$ | The $k$-th VM node in $\mathcal{K}$ |
| $\mathcal{N}$ | The set of serverless functions |
| $n \in \mathcal{N}$ | The $n$-th serverless function in $\mathcal{N}$ |
| $a_n, \forall n \in \mathcal{N}$ | The arrival time of function $n$ |
| $d_n, \forall n \in \mathcal{N}$ | The strict deadline of function $n$ |
| $r_n, \forall n \in \mathcal{N}$ | The input workload size of function $n$ |
| $\mathcal{K}_n$ | The set of VM nodes available to function $n$ |
| $\mathcal{T}$ | The set of time slots |
| $\tau$ | The length of each time slot in $\mathcal{T}$ |
| $\mathcal{R} = \mathcal{K} \times \mathcal{T}$ | The resource mesh |
| $r \in \mathcal{R}$ | The $r$-th resource unit in $\mathcal{R}$ |
| $C_r, \forall r \in \mathcal{R}$ | The maximum input that can be processed by $r$ |
| $\mathcal{R}_n, \forall n \in \mathcal{N}$ | The set of resource units available to $n$ |
| $x_{nr}$ | The size of workloads of $n$ dispatched to $r$ |
| $\mathcal{R}_{nr}$ | The maximum processing capacity of $r$ for $n$ |
| $f_n(\cdot)$ | The utility of function $n$ |
| $g_n(\cdot)$ | The revenue of invoking function $n$ on $r$ |
| $\phi_r(\cdot)$ | The estimation of the marginal cost for $r$ |
| $\mathcal{W}_n(x_n)$ | The pseudo-social welfare of decision $x_n$ |
| $\omega(n)$ | The resource usage of $r$ before function $n$ arrives |

### 2.1 Spatio-Temporal Resource Mesh

Each VM node is capable of processing a set of heterogenous serverless functions arriving in sequence with different service rates. Let us denote the set of functions as $\mathcal{N}$ and index each of them by $n$. Each function $n$ has the input data of size $q_n$ (in MB). Take video transcoding with ExCamera\cite{22} as an example, the inputs are raw video frames. ExCamera firstly partitions the input into frame pieces with negligible cost. Then, it parallelizes the “slow” pieces of the encoding, and performs the “fast” pieces serially\cite{23}. $\forall n \in \mathcal{N}$, we use $a_n$ and $d_n$ to represent its arrival time and strict 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 instances.

Our target is to maximize the social welfare, i.e., the sum of all functions’ utility 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 maximum workload processing capacity of \( r \) for function \( n \). It results to the constraint \( 0 \leq x_{nr} \leq \chi_{nr} \). Note that \( C_r \) is designed to indicate the overall processing capacity while \( \chi_{nr} \) is the workload processing capacity to the specific function \( n \) of the \( r \)-th resource unit.

We take a zero-startup utility \( f_n : [0, \chi_{nr}] \rightarrow \mathbb{R} \), where \( \chi_{nr} \equiv \{ \chi_{nr} \}_{r \in \mathcal{R}_n} \) as the measurement of user satisfaction for function \( n \). As a widely accepted assumption in previous works [6, 10, 20, 24, 25], 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 \( \{f_n\}_{n \in \mathcal{N}} \). Note that we allow functions to have different utilities. For each function \( n \in \mathcal{N} \), its utility is defined as the sum of separate sub-utilities achieved through each available resource unit:

\[
f_n(x_n) \triangleq \sum_{r \in \mathcal{R}_n} f_n(x_{nr}), \forall n \in \mathcal{N}, \tag{2}\]

where \( x_n \equiv \{x_{nr}\}_{r \in \mathcal{R}_n} \). For a given function \( n \), \( f_n \) can also be different on different \( r \in \mathcal{R}_n \). To sum up, a function can be described with the quadruple \( \{a_n, \mathcal{R}_n, \chi_{nr}, f_n\} \).

Serverless comes with a pay-for-value billing model [23]. Therefore, its revenue is linearly proportional to the actual resource consumption. Formally, we define the revenue of invoking function \( n \) on resource unit \( r \) as

\[
g_{nr}(x_{nr}) \triangleq \frac{x_{nr}}{C_r}, \forall n \in \mathcal{N}, r \in \mathcal{R}_n, \tag{3}\]

where \( \frac{\chi_{nr}}{C_r} \) is used to indicate the fractional resource consumed by \( x_{nr} \), and \( \beta_{nr} \) is the price of provisioning total resource of \( r \) to \( n \). For instance, \( \beta_{nr} \) equals to $0.015 when the raw video’s resolution is less than 720p with Amazon Elastic Transcoder [27]. Our model is consistent with all the mainstream platform providers’ pricing strategies [27, 28, 29, 30].

### 2.3 Online Social Welfare Maximization

Based on the above content, we formulate the social welfare maximization problem as follows:

\[
\mathcal{P}_1 : \max_{\{x_n\}_{n \in \mathcal{N}}} \sum_{n \in \mathcal{N}} f_n(x_n) + \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}_n} g_{nr}(x_{nr}) \\
\text{s.t.} \quad \sum_{r \in \mathcal{R}_n} x_{nr} \leq a_n, \forall n \in \mathcal{N}, \tag{4}
\]

\[
x_{nr} = 0, \forall n \in \mathcal{N}, r \in \mathcal{R}\setminus\mathcal{R}_n, \tag{5}
\]

\[
\sum_{n \in \mathcal{N}} x_{nr} \leq C_r, \forall r \in \mathcal{R}, \tag{6}
\]

\[
0 \leq x_{nr} \leq \chi_{nr}, \forall n \in \mathcal{N}, r \in \mathcal{R}_n. \tag{7}
\]

As an offline optimization problem, although \( \mathcal{P}_1 \) is difficult to solve, it is built based on complete knowledge. However, in online settings, the platform should not have the information of the \( n \)-th quadruple \( \{a_n, \mathcal{R}_n, \chi_{nr}, f_n\} \) until the invocation request of function \( n \) arrives. To design an

3. https://aws.amazon.com/ec2/spot/pricing/

4. The discrete version of problem \( \mathcal{P}_1 \) is actually a multi-dimensional 0-1 knapsack problem, which is proved to be NP-complete [31].
efficient online algorithm with the worst-case performance guarantee, we introduce the following notations

\[
\begin{align*}
  t &\triangleq \min_{n \in N} \min_{r \in R_n} \left( \frac{\partial f_n}{\partial x_{nr}} + \frac{\partial g_n}{\partial x_{nr}} \right) \\
v &\triangleq \max_{n \in N} \max_{r \in R_n} \left( \frac{\partial f_n}{\partial x_{nr}} + \frac{\partial g_n}{\partial x_{nr}} \right).
\end{align*}
\]

(8)

The ratio between these two constants, i.e., \( \frac{v}{t} \), demonstrates the fluctuation of the marginal social welfare, which will be introduced later. In previous theoretical papers, this ratio is viewed as a known variable and it helps construct the resource provision decisions \([24], [32], [33], [34], [35], [36]\). For example, in \([34]\), the ratio is set as 36 in default.

It is worth noting that the online decision \( x_n \) made for function \( n \) when it arrives implies the idea of resource reservation. Support that at time \( t \), function \( n \) arrives. Formally, for function \( n \), by solving \( P_1 \) online, the resource provision decisions

\[
\{x_{nr, t'} | r k' \in R_n, t' > t \}
\]

are the resource reservation results for executing function \( n \).

3 ALGORITHM DESIGN

The key challenge to solve \( P_1 \) in online settings is that the dispatching of each function’s workloads to each resource unit are coupled because of \([6]\). Nevertheless, if we could construct several feasible dual variables corresponding to \( \{x_n\}_{n \in N} \) in \( P_1 \), 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 provision problems, i.e., estimate the cost for provisioning resources to each newly arrived serverless function as a function of resource surplus \([24], [25], [33], [34], [37]\). In the following sections, firstly, we show how the pseudo-social welfare functions are constructed. Then, based on these constructed functions, we introduce our algorithm OnSocMax. It works by solving several pseudo-social welfare maximization problems polynomially online. To guarantee that OnSocMax is \( \alpha \)-competitive, we analyze the requirements that the marginal cost functions should satisfy. In addition, we give the bound of the gap between the competitive ratio achieved by OnSocMax and the optimal competitive ratio of a simplified case under a particular condition. In the end, we discuss how to extend OnSocMax to serverless functions with non-partitionable input workloads and the implementation concerns.

3.1 Pseudo-Social Welfare Function

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

\[
\begin{align*}
  f_n(x_n) &+ \sum_{r \in R_n} \int_{\omega_{nr}^{(i)}(x_n)}^{\omega_{nr}^{(i)+1}} \phi_r(u) du + g_n(x_{nr}),
\end{align*}
\]

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 function arrives, the higher cost it has to pay \([24]\). The first component is the pseudo-utility of executing function \( n \), 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

\[
\begin{align*}
  f_n(x_n) + \sum_{r \in R_n} g_n(x_{nr}) - \int_{\omega_{nr}^{(i)}}^{\omega_{nr}^{(i)+1}} \phi_r(u) du,
\end{align*}
\]

the second component can be regarded as the net profit of the platform for provisioning resources to function \( n \). In this case, the later a function arrives, the harder the resource surplus to meet its requirements before deadline, 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 \( x^*_n \) 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

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

and \( \mu \triangleq [\mu_n]_{n \in N} \) and \( \lambda \triangleq [\lambda_r]_{r \in R} \) are the dual variables corresponding to \((3)\) and \((6)\), respectively.

Proof. The result is immediate with Lagrangian. \( \square \)

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

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

(10)

Based on this, we can interpret \( p \) as the marginal cost for processing unit workload \([24]\). 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 \( \phi_r(C_r) \geq \nu \), if (i) \( x^*_n \triangleq x^*_n \in R_n \) and \( \mu^*_n \) are respectively the optimal primal solution and the optimal dual solution to \((3)\) of the following problem \( P_3 \):

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

s.t. \((4), (5), (7).\)

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

\[
\begin{align*}
  \omega_r^{(n+1)} &\approx \omega_r^{(n)} + x^*_n \\
  \omega_r^{(1)} &\approx 0,
\end{align*}
\]

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

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 \( \phi_r(C_r) \geq \nu \), there must exist a resource usage
level $\omega_r \leq C_r$ such that $\phi_r(\omega_r) = \nu$. Note that the function $f_n + \sum_{r \in \mathcal{R}_n} g_{nr}$ is non-decreasing and its derivative on $r$ is not more than $\omega_r$. Therefore, when the input of $\phi_r$ is $\omega_r(\nu) + x_{nr}$, suppose $\omega_r + x_{nr} \leq \omega_r$. Consequently, the derivative of the integral function
\[
\Phi_r(x_{nr}) = \int_{\omega_r(\nu)}^{\omega_r(\nu) + x_{nr}} \phi_r(u) du
\]
is continuous, non-decreasing, and convex when $x_{nr} \leq \omega_r - \omega_r(\nu)$. The convexity is because $\Phi_r$, 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 $x_{nr}, \mu_{nr}, \gamma_{nr},$ and $\zeta_{nr}$ to denote the optimal primal and dual solutions of $P_3$ ($\mu_{nr}^* \rightarrow \Phi_r$) while $\gamma_{nr}$ and $\zeta_{nr}$ to the right part and left part of (7), respectively. The KKT conditions of $P_3$ are listed below.
\[
\begin{align*}
&f'_{nr}(x_{nr}^*) + \frac{\omega_r}{C_r} - \phi_r(\omega_r(\nu+1)) - \mu_{nr}^* = f_n(x_{nr}^*) + g_{nr}(x_{nr}^*) \\
&\gamma_{nr}^*(x_{nr}^* - \bar{x}_{nr}) = 0 \\
&\zeta_{nr}^* \cdot x_{nr}^* = 0 \\
&\mu^*_{nr} \left( \sum_{r \in \mathcal{R}_n} x_{nr}^* - g_{nr} \right) = 0.
\end{align*}
\]
With KKT conditions (13), we show that the optimal solution $x_{nr}^*$ of $P_3$ simultaneously optimizes the conjugate $\xi_{nr}(p)$ given $p = \phi_r(\omega_r(\nu+1)) + \mu_{nr}^*$, i.e.,
\[
\begin{align*}
\xi_{nr}\left(\phi_r(\omega_r(\nu+1)) + \mu_{nr}^*\right) &= f_n(x_{nr}^*) + g_{nr}(x_{nr}^*) \\
&= \left(\phi_r(\omega_r(\nu+1)) + \mu_{nr}^*\right) x_{nr}^*.
\end{align*}
\]

3.2 OnSocMax Design

OnSocMax is built on solving $P_3$ for each newly arrived function $n$ in sequence. The procedure is captured in Algorithm 1. We place a hat on top of variables that denote the variables involved in OnSocMax.

Although $P_3$ is a convex program, we cannot obtain its analytic solution with (13) directly. To solve it iteratively, we transform it into the following problem $P'_3$:
\[
P'_3 : \min_{x_{nr} \in \mathcal{R}_n} \left[ \Phi_r(x_{nr}) - f_n(x_{nr}) - g_{nr}(x_{nr}) \right]
\]
s.t.
\[
\begin{align*}
&\sum_{r \in \mathcal{R}_n} x_{nr} - g_{nr} + s = 0, \quad (17) \\
x_{nr} - \bar{x}_{nr} + l_r = 0, \forall r \in \mathcal{R}_n, \quad (18) \\
q_r - x_{nr} = 0, \forall r \in \mathcal{R}_n, \quad (19) \\
s, l_r, q_r \geq 0, \forall r \in \mathcal{R}_n. \quad (20)
\end{align*}
\]

Algorithm 1: OnSocMax

Input: $\{C_r\}_{r \in \mathcal{R}}$ and $\{g_{nr}\}_{n \in \mathcal{N}, r \in \mathcal{R}}$

Output: Online solution to $P_3$ and final utilizations for the resource mesh

1. $\forall r \in \mathcal{R} : \hat{\omega}_r(1) \leftarrow 0$
2. while a new function $n$ arrives do
3. \hspace{1em} Receive the quadruple $\{g_n, \bar{x}_n, \hat{x}_n, f_n\}$
4. \hspace{1em} Get the (near) optimal solution $\hat{x}_n$ of $P_3$
5. \hspace{1em} for $r \in \mathcal{R}_n$ do in parallel
6. \hspace{2em} $\hat{\omega}_r(\nu+1) \leftarrow \hat{\omega}_r(\nu) + \hat{x}_{nr} / \hat{x}_{nr}$ // Update utilization
7. \hspace{1em} end for
8. \hspace{1em} $n \leftarrow n + 1$
9. end while
10. return $\{\hat{x}_n\}_{n \in \mathcal{N}}$ and $\{\hat{\omega}_r(\nu+1)\}_{r \in \mathcal{R}}$
where \( l \triangleq [l_r]_{r \in \mathbb{R}_+}, q \triangleq [q_r]_{r \in \mathbb{R}_+} \) and \( s \) are introduced slack variables. In addition, we define the dual variables respectively to (17), (18), and (19) as \( \mu, y \triangleq [y_r]_{r \in \mathbb{R}_+} \), and \( z \triangleq [z_r]_{r \in \mathbb{R}_+} \).

The augmented Lagrangian of \( P'_3 \) is

\[
L_\sigma(x_n, l, q, \mu, y, z) = \sum_{r \in \mathbb{R}_+} \left[ \Phi_r(x_{nr}) - f_{nr}(x_{nr}) - g_{nr}(x_{nr}) \right] + \mu \left( \sum_{r \in \mathbb{R}_+} x_{nr} - q_n + s + \sum_{r \in \mathbb{R}_+} y_r (x_{nr} - x_{nr} + l) \right) + \sum_{r \in \mathbb{R}_+} z_r (q_r - x_{nr}) + \frac{\sigma}{2} \rho(x_n, s, l, q),
\]

where the penalty function \( \rho(\cdot) \) is defined as

\[
\rho(x_n, s, l, q) \triangleq \sum_{r \in \mathbb{R}_+} \left( q_r - x_{nr} \right)^2 + \left( \sum_{r \in \mathbb{R}_+} x_{nr} - q_n + s \right)^2 + \sum_{r \in \mathbb{R}_+} \left( x_{nr} - x_{nr} + l \right)^2.
\]

and \( \sigma > 0 \) is the penalty coefficient. If we consider the augmented Lagrangian \( L_\sigma(x_n, s, l, q, \mu, y, z) \) as the function of the slack variables \([s, l, q]\), to minimize it, we can get their optimal values:

\[
\begin{align*}
    s^* &= \max \left\{ -\frac{\mu}{\sigma} + q_n - \sum_{r \in \mathbb{R}_+} x_{nr}, 0 \right\}, \\
l^* &= \max \left\{ -\frac{\mu}{\sigma} + \sum_{r \in \mathbb{R}_+} x_{nr} - q_n, 0 \right\}, \forall r \in \mathbb{R}_+, \quad (23)
\end{align*}
\]

Taking (23) into (21), we have

\[
L_\sigma(x_n, \mu, y, z) = \sum_{r \in \mathbb{R}_+} \left[ \Phi_r(x_{nr}) - f_{nr}(x_{nr}) - g_{nr}(x_{nr}) \right] + \frac{\sigma}{2} \left[ \max \left\{ \frac{\mu}{\sigma} + \sum_{r \in \mathbb{R}_+} x_{nr} - q_n, 0 \right\}^2 - \frac{\mu^2}{\sigma^2} \right] + \frac{\sigma}{2} \sum_{r \in \mathbb{R}_+} \left[ \max \left\{ \frac{y_r}{\sigma} + x_{nr} - x_{nr}, 0 \right\}^2 - \frac{y_r^2}{\sigma^2} \right] + \frac{\sigma}{2} \sum_{r \in \mathbb{R}_+} \left[ \max \left\{ \frac{z_r}{\sigma} - x_{nr}, 0 \right\}^2 - \frac{z_r^2}{\sigma^2} \right]. \quad (24)
\]

Besides, we define the constraint violation degree \( v(\cdot) \) of a solution \( x_n \) given \((\mu, y, z)\) by

\[
v(x_n \mid \mu, y, z) \triangleq \max \left\{ \sum_{r \in \mathbb{R}_+} x_{nr} - q_n, -\frac{\mu}{\sigma_k} \right\} + \sum_{r \in \mathbb{R}_+} \max \left\{ x_{nr} - x_{nr}, -\frac{y_r}{\sigma_n} \right\} + \sum_{r \in \mathbb{R}_+} \max \left\{ -x_{nr}, -\frac{z_r}{\sigma_k} \right\}. \quad (25)
\]

Based on (24), we can solve \( P'_3 \) approximately with the augmented Lagrangian method. The procedure is summarized in Algorithm 2 which is used to substitute step 4 of OnSocMax.

### 3.3 Competitive Analysis

OnSocMax is at most polynomial because \( P_3 \) is convex and can be solved efficiently in polynomial time with augmented Lagrangian method. Obviously, \( \{x_n\}_{n \in \mathcal{N}} \) is feasible to \( P_1 \). To quantify how “good” OnSocMax is, we adopt the standard competitive analysis framework [5].

**Definition 1.** (38) For any arrival instance \( A \) of all functions \( n \in \mathcal{N}, \) the competitive ratio for an online algorithm is defined as

\[
\alpha \triangleq \max_{A} \frac{\Theta_{P_1}(A)}{\Theta_{on}(A)}.
\]
where $\Theta_{P_1}(A)$ is the maximum objective value of $P_1$, $\Theta_{on}(A)$ is the objective function value of $P_1$ obtained by this online algorithm.

The competitive ratio $\alpha$ 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 $\{\hat{\phi}_r\}_{r \in \mathcal{R}}$ should satisfy to guarantee that OnSocMax is $\alpha$-competitive for some $\alpha$.

**Theorem 1.** 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

\[
\hat{\phi}_r(\omega) = \begin{cases} 
\tilde{\phi}_r(\omega) & \omega \in [0, \tilde{\omega}_r) \\
\varepsilon & \omega \in [\tilde{\omega}_r, C_r] \\
\omega & \omega \in (C_r, +\infty),
\end{cases}
\]

(27)

where $\tilde{\omega}_r$ is a resource utilization threshold, and $\hat{\phi}_r$ is a non-decreasing function that satisfies

\[
\begin{align*}
\tilde{\phi}_r(\omega)C_r & \leq \alpha \int_0^\omega \hat{\phi}_r(u)du - \omega \cdot \varepsilon, \quad \omega \in [\tilde{\omega}_r, C] \\
\tilde{\phi}_r(C_r) & \geq \varepsilon.
\end{align*}
\]

(28)

**Proof.** To prove this result, we refer to the technique named instance-dependent online primal-dual approach, proposed in [34]. The key idea is to construct a dual solution to $P_2$ based on the solution $\{\tilde{x}_n\}_{n \in \mathcal{N}}$ produced by OnSocMax. Then, it uses this dual objective to build the upper bound of the optimum of $P_1$ based on weak duality. When building the upper bound, this technique studies the worst-case instances under different scenarios.

Let us use $B \triangleq \{A_1, A_2, \ldots\}$ to denote the set of arrival instances of all functions, and use $\Theta_{P_1}(A)$ to denote a feasible objective value of the dual problem $P_2$ for any arrival instance $A$. Hereinafter, we just replace $\omega_r^{(N)}$ by $\tilde{\omega}_r^{N}$ for simplification. We divide $B$ into three disjoint sets:

\[
\begin{align*}
B_1 & \triangleq \{A \mid 0 \leq \tilde{\omega}_r^{N} < \tilde{\omega}_r, \forall r \in \mathcal{R}\} \\
B_2 & \triangleq \{A \mid \tilde{\omega}_r \leq \tilde{\omega}_r^{N} \leq C_r, \forall r \in \mathcal{R}\} \\
B_3 & \triangleq B \setminus (B_1 \cup B_2).
\end{align*}
\]

(29)

$B_1$ and $B_2$ contain the instances whose final utilizations of all resource units in the mesh are below and above the threshold $\tilde{\omega}_r$, respectively. Our goal is to prove that, under the conditions (27) and (28), $\forall A \in B_1, B_2, B_3$ respectively, the following relations hold:

\[
\alpha \cdot \Theta_{on}(A) \geq \Theta_{P_1}(A) \geq \Theta_{P_1}^*(A).
\]

(30)

In the following analysis, we just drop the parentheses and $A$ for simplification.

**Case I:** $\forall A \in B_1$, from (27) we can find that the marginal costs experienced by all functions are the same, i.e., $\varepsilon$. In this case, each function $n$ is processed with maximum permitted rate $\tilde{\lambda}_{nr}$ on $r \in \mathcal{R}_n$. Thus, $\Theta_{P_1}/\Theta_{on} = 1 \leq \alpha$.

**Case II:** $\forall A \in B_2$, we construct a feasible dual solution to $P_2$ as

\[
\begin{align*}
\mu_n & = \mu_r^{(N)}, \quad \forall n \in \mathcal{N} \\
\lambda_r & = \tilde{\phi}_r(\tilde{\omega}_r^{N}), \quad \forall r \in \mathcal{R},
\end{align*}
\]

(31)

where $\mu_r^{(N)}$ is the optimal dual solution to $P_3$ 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,

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

(32)

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

\[
\Theta_{P_1} \leq \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \xi_{nr}(\mu_r^{(N)} + \hat{\phi}_r(\tilde{\omega}_r^{N})) + \sum_{n \in \mathcal{N}} \mu_r^{(N)} \varrho_n + \sum_{r \in \mathcal{R}} \hat{\phi}_r(\tilde{\omega}_r^{N})C_r \quad \triangleright \text{the right-side is } \Theta_{P_2}
\]

\[
\leq \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \xi_{nr}(\mu^{(n+1)}_r + \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})) + \sum_{n \in \mathcal{N}} \mu^{(n+1)}_r \varrho_n + \sum_{r \in \mathcal{R}} \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})C_r \quad \triangleright \ref{32}
\]

\[
= \sum_{r \in \mathcal{R}} \left[ \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})C_r - \sum_{n \in \mathcal{N}} \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})\hat{x}_{nr} \right]
\]

\[
+ \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \leq \Theta_{mp} \triangleq (\ref{14}).
\]

(33)

The last equality holds because $\hat{x}_{nr}$ simultaneously maximizes $P_3$ and the conjugate $\xi_{nr}(\mu_r^{(N)} + \hat{\phi}_r(\tilde{\omega}_r^{N}))$ (result of Proposition 2). Since $\{\hat{\phi}_r\}_{r \in \mathcal{R}}$ are non-decreasing, $\forall n \in \mathcal{N}, r \in \mathcal{R},$

\[
\hat{\phi}_r(\tilde{\omega}_r^{(n+1)})\hat{x}_{nr} \geq \int_{\tilde{\omega}_r^{(n)}}^{\tilde{\omega}_r^{(n+1)}} \hat{\phi}_r(u)du.
\]

(34)

(33) is illustrated in Fig. 3 Further, we have

\[
\sum_{n \in \mathcal{N}} \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})\hat{x}_{nr} \geq \int_{\tilde{\omega}_r^{(n+1)}}^{\tilde{\omega}_r^{(n)}} \hat{\phi}_r(u)du,
\]

(35)

where $\tilde{\omega}_r^{(1)} = 0$ because of (11). Besides, from Fig. 2 we can find that $\xi_{nr}(\mu_r^{(N)} + \hat{\phi}_r(\tilde{\omega}_r^{(n+1)})) \geq 0$ holds for all the serverless functions. Thus, based on (34), we have

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

Based on the above results (34) and (35), we have

\[
\Theta_{mp} \leq \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \left[ \hat{\phi}_r(\tilde{\omega}_r^{N})C_r - \int_0^{\tilde{\omega}_r^{N}} \hat{\phi}_r(u)du \right]
\]

\[
+ \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \quad \triangleright \ref{34}
\]

\[
\leq \sum_{r \in \mathcal{R}} \left( (\alpha - 1) \int_0^{\tilde{\omega}_r^{(N)}} \hat{\phi}_r(u)du \right) \triangleright \ref{28} & \text{ drop } \lambda_r \cdot \tilde{\omega}_r^{(N)}
\]

\[
+ \sum_{n \in \mathcal{N}, r \in \mathcal{R}} \left( f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr}) \right) \alpha. \quad \triangleright \ref{35}
\]
The final expression is exactly \( \alpha \cdot \Theta _ { \Omega _ { n} } \). Thus, \( \Theta _ { p _ { r } } / \Theta _ { \Omega _ { n} } \leq \alpha \).

**Case III:** \( \forall A \in B _ { i } \), we define two disjoint sets to split the resource mesh \( R \):

\[
\begin{align*}
R _ { 1 } & := \{ r \in R \mid 0 \leq \hat{\omega } ^ { r } _ { p } < \hat{\omega } _ { r }, \forall r \in R \} \\
R _ { 2 } & := \{ r \in R \mid \hat{\omega } _ { r } \leq \hat{\omega } ^ { r } _ { p } \leq C _ { r }, \forall r \in R \}.
\end{align*}
\] (36)

For resource unit \( r \) in different sets, the corresponding dual variables are constructed in different ways. We extend \( P _ { 1 } \) to \( P _ { 2 } ^ { \prime } \) by adding the following constraint:

\[
\sum _ { n \in N } \sum _ { r \in R _ { 1 } } x _ { n r } \leq \sum _ { r \in R } \hat{\omega } ^ { r } _ { p }.
\] (37)

Apparently, \( P _ { 1 } ^ { \prime } \) is the same as \( P _ { 1 } \) for OnSocMax since (37) is not violated by \( \{ \hat{\omega } _ { n } \} _ { n \in N } \). The dual problem \( P _ { 2 } ^ { \prime } \) to \( P _ { 1 } ^ { \prime } \) is

\[
P _ { 2 } ^ { \prime } : \min _ { \mu, \lambda } \sum _ { n \in N } \left( \sum _ { r \in R _ { 1 } } \xi _ { n r } (\mu _ { n } + \lambda _ { r } + \delta ) + \sum _ { r \in R _ { 2 } } \xi _ { n r } (\mu _ { n } + \lambda _ { r } ) \right) + \sum _ { n \in N } \mu _ { n } \theta _ { n } + \lambda _ { r } C _ { r } + \delta \sum _ { r \in R } \hat{\omega } ^ { r } _ { p }.
\]

\[
\text{s.t. } 5, 7, \mu \geq 0, \lambda \geq 0, \delta \geq 0,
\]

where \( \delta \) is the dual variable corresponding to the newly added constraint (37). Then, we construct the dual solution to \( P _ { 2 } ^ { \prime } \) as

\[
\delta = \lambda
\]

\[
\mu _ { n } = \mu ^ { \prime } _ { n } \quad \forall n \in N.
\] (39)

Based on (39), we can follow a similar approach as show in **Case II** to obtain that \( \Theta _ { p _ { r } } / \Theta _ { \Omega _ { n} } \leq \alpha \). A slight difference is that, in **Case III**, when applying (28) to \( \Theta _ { \Omega _ { n} } \), 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 (39), we have the detailed design of \( \{ \hat{\omega } _ { r } \} _ { r \in R } \), which is irrelevant with the utilities \( \{ f _ { n r } \} _ { n \in N } \) and \( \{ g _ { n r } \} _ { n \in N, r \in R } \), as follows.

**Theorem 2.** \( \forall r \in R _ { n } \), if the marginal cost function \( \hat{\omega } _ { r } \) introduced in the pseudo-social welfare function is designed as

\[
\hat{\omega } _ { r } (\omega ) = \begin{cases} 
\frac{t}{\exp (\alpha - t) - \exp (-\alpha)} e ^ { \frac{\alpha}{\omega _ { r } } } + \frac{1}{\alpha} & \omega \in [\omega _ { r }, C _ { r }] \\
\infty & \omega \in (C _ { r }, +\infty),
\end{cases}
\]

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

\[
\hat{\omega } - 1 = \frac{1}{\alpha - 1} + \ln \frac{\hat{\omega } + 1}{\hat{\omega } - 1} \quad (40)
\]

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

**Proof.** We firstly introduce the Gronwall’s inequality (39) as follows. \( \forall x \in [\omega, \infty] \), if \( f (x) \leq a (x) + b (x) \int _ { x } ^ { \infty } f (u) du \), then

\[
f (x) \leq a (x) + b (x) \int _ { x } ^ { \infty } a (u) \left( \int _ { u } ^ { \infty } b (w) dw \right) du.
\] (41)

where \( f (x) \) is continuous, \( a (x) \) and \( b (x) \) are integrable and \( \forall x \in [\omega, \infty] \), \( b (x) \geq 0 \). The result remains valid if all the ‘\( \leq \)’ are replaced by ‘\( = \)’. Applying (41) to (28) leads to

\[
t \leq \hat{\omega } _ { r } (C _ { r } ) \leq \frac{t}{\alpha} + \left[ \frac{\hat{\omega } _ { r } (\hat{\omega } - 1)}{C _ { r } } - \frac{t}{\alpha} \right] e ^ { \frac{1}{\alpha} (\hat{\omega } - \hat{\omega } _ { r } ) }.
\] (42)

Thus, the minimum \( \hat{\omega } \) is achieved when all inequalities in (28) and (12) are binding, which leads to the design of \( \{ \hat{\omega } _ { r } \} _ { r \in R } \) and the competitive ratio achieved by (40).

In the following, we prove the results of (ii). When \( R = 1, P _ { 1 } \) degenerates to the general one-way trading (GOT) problem (40). The optimal competitive ratio is proved to be \( 1 + \ln \left( \frac{v}{t} \right) \leq 24, 53, 54, 56, 40 \). With \( \hat{\omega } \geq \frac{\alpha}{\alpha + 1} + 1 \), let us take \( y \geq 1 \) as a substitute for \( \hat{\omega } - 1 \). Then

\[
\hat{\omega } - 1 = \frac{1}{\alpha - 1} + \ln \frac{\hat{\omega } + 1}{\hat{\omega } - 1} \quad (40)
\]

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

By the design of \( \hat{\omega } _ { r } (\cdot ) \), we observe that \( \alpha \geq 2 \) holds because \( \frac{\alpha}{\alpha + 1} \geq 1 \). When \( \{ f _ { n r } \} _ { n \in N, r \in R _ { n } } \) are linear and share the same coefficient, \( \hat{\omega } = 2 \).

### 3.4 Extending to Non-Partitionable Workloads

OnSocMax can also be applied to serverless functions whose workloads are not permitted to be partitioned. Specifically, in this case, (7) is replaced by

\[
x _ { n r } \in \{ 0, \infty \}, \forall n \in N, r \in R _ { n },
\] (43)

and \( \omega _ { n r } = \sum _ { n \in N } x _ { n r } \). To solve the new problem in online settings, we can approximate the marginal cost defined in (12) with \( \hat{\omega } _ { r } (\omega _ { r } ^ { n r } + \hat{\mu } _ { n r } ) x _ { n r } \). With this substitution, the case III in Fig. 2 is merged into Case I or Case II, and OnSocMax achieves the same competitive ratio as shown in **Theorem 2**. This approach is exactly the implementation of (10).
3.5 Implementation Concerns

In our model, serverless functions are invoked 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 VM nodes that have been used for running functions beforehand and are maintained to serve future invocations. Besides, considering that the workloads of one function 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 data packets between cloud functions. Based on these techniques, OnSocMax can be easily implemented as an optional policy for the load balancer and it is triggered everytime a new function invocation request arrives.

![Image](55x396 to 293x536)

Fig. 4. The architecture of OnSocMax.

Fig. 4 gives the overview diagram of OnSocMax implementation. OnSocMax can be integrated with OpenWhisk, which exposes an Nginx-based REST interface for creating and invoking functions. When triggering the invocation of a newly arrived function, the load balancer selects the best VM nodes for current execution and resource reservation with the running results of OnSocMax and sends the scheduling decision to the Kafka-based distributed message queue. With that, the invoker inside the cluster executes the function by starting a corresponding Docker container. During the functions’ execution, a rendezvous server is in continuous running for packets relay. At last, the running record will be written to a database and return back to end users.

4 Experimental Validation

In this section, we conduct several simulations to validate the theoretical superiority of OnSocMax. The experiments are not meant to be exhaustive, rather they are used to illustrate the potential of OnSocMax. A system implementation of OnSocMax will be left as our future work based on the architecture shown in Fig. 4.

4.1 Experimental Setup

Functions and VM nodes. We consider a cluster with 10 VM nodes in the time horizon of 24 time slots. The processing capacity of VM nodes are generated from an i.i.d. Gaussian $\mathcal{N}(\mu = 20, \sigma = 2)$. By setting $\tau$ as 60 minutes, the time horizon represents one day. We set the number of serverless functions as 20. The number of function arrivals in each time slot follows a Poisson distribution with a mean of 2.03 function requests, which is independent of other time slots in this day. The deadline of each function is calculated based on the arrive time and the maximum service duration of it, where the latter is generated by an Exponential distribution with a mean of 4 time slots (2 hours). Each function has an input workload whose size is generated from a Normal distribution $\mathcal{N}(\mu = 18, \sigma = 3)$. The maximum data processing speed of each function is generated from a Normal distribution $\mathcal{N}(\mu = 7, \sigma = 1)$.

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

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

where the coefficient $a$ is generated from a uniform distribution in $[0, \mu]$. Similarly, the pricing parameter $\beta_{nr}$ in $g_{nr}(\cdot)$ is generated from the uniform distribution in $[0.1, 0.5]$.

Hyper-Parameters of OnSocMax. There are many algorithmic hyper-parameters involve setting in Algorithm 2. For example, the initial penalty coefficient $\sigma_0$, the initial constraint violation coefficient $\sigma_0$, the initial precision coefficient $\varepsilon_0$, etc. Their default settings are listed in Table 2 in the last line of the algorithm.

![Table 2](55x396 to 293x536)

| Parameter | Value | Parameter | Value |
|-----------|-------|-----------|-------|
| $\sigma_0$ | 0.97  | $p$       | 1.002 |
| $\theta_1$ | 0.99 | $\theta_2$ | 0.999 |
| $\eta$ | 0.1 | $\varepsilon$ | 10 |
| learning rate | 2 | decay | 0.95 |

Algorithms Compared. We compare OnSocMax with two handcrafted online algorithms.

- **Max-First.** In Max-First, each VM node always serves the function with the highest myopic social welfare, i.e., the sum of the utility of the chosen function and the revenue for serving it in each time slot is maximal. Max-First is myopic because it always maximizes the partial social welfare that it sees.
- **Equal-Share.** In Equal-Share, each VM node serves every newly arrived function with equal opportunity within its capacity limit.

4.2 Simulation Results

In the following, firstly, we verify the performance of OnSocMax against the handcrafted policies on the social welfare.

6. Note that the Poisson distribution and the exponential distribution are used for data generation, which is not necessary to OnSocMax.
achieved. Then, we analyze the robustness of OnSocMax under different parameter settings.

**Theoretical Superiority.** Fig. 5 and Fig. 6 show the social welfare achieved under different service duration and the maximum processing speed settings. We can observe that all the algorithms achieve higher social welfare when these two variables increase. The reason is that, when the capacity of VM nodes are sufficient, increasing the service duration and the maximum processing speed of functions can increase the opportunities of being fully served. In spite of this, OnSocMax always performs the best among these online algorithms. Besides, OnSocMax performs the best for linear utilities. This is because the logarithmic and polynomial utilities have diminishing returns, which could increase the fluctuation ratio $\psi$. This will cause more functions be served with their marginal costs fall into the second segment of $\phi_r(\omega)$, which further leads to the decrease of social welfare.

5 **RELATED WORK**

Optimal load balancing is fully investigated under classic settings, where multiple identical servers with exponentially distributed service rates process continuous arrived jobs. With CTMC and Lyapunov Stability theories, load balancing policies such as JSQ [19], JIQ [9], Pod [17], and JFIQ [15] are proposed and analyzed on the average response time and cross-server communication overhead. In a most recent work [15], Weng et al. proposed the JFSQ and JFIQ policies under the constraints of heterogenous service rates and service locality. They prove 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 series of works study the energy-efficient and multi-resource sharing load balancing from a different theoretical basis [5], [6], [7], [11], [20], [21], [43], [44]. Generally, the objective is to improve the energy efficiency with on-demand resource allocation. Thereinto, online load balancing of deadline-sensitive functions (or jobs, etc) have been studied in [6], [20], [44]. In a similar work [20], 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 [45]. 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.
6 Conclusion

In this paper, we study the online optimal load balancing problem for serverless with a target of maximizing the social welfare. The serverless functions we considered have strict deadlines and limited data processing speeds. To take both the spatio and temporal resource of the serverless platform into consideration, we establish a model of resource mesh. Each function invocation request can only be dispatched to the VM nodes available to it based on the service locality constraints. With 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 $\alpha$ at least 2. Online load balancing for serverless functions with complex workflows and communication patterns will be studied in future.

Acknowledgments

This work was partially supported by the National Science Foundation of China (NSFC) under Grants U20A20173 and 62125206. Schahram Dustdar’s work is supported by the Zhejiang University Deqing Institute of Advanced technology and Industirilization (ZDATI).

References

[1] M. S. Aslanpour, A. N. Toosi, C. Cicconetti, B. Javadi, P. Sbarski, 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. Sirimaa, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelone, B. Javadi, L. M. Vaquero, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojic, 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. 3, Nov. 2018.

[3] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, 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, “Geographical 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] 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. 68, no. 11, p. 1056–1071, Nov. 2011.

[10] 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.

[11] G. Aumala, E. Boza, L. Ortiz-Avila, 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.

[12] W. Lloyd, M. Yu, B. Zhang, O. David, and G. Leavesley, “Improving data migration mitigation to serverless computing platforms: Latency mitigation with keep-alive workloads,” in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 195–200.

[13] H. Yu, A. A. Iriappane, H. Wang, and W. J. Lloyd, “Faasrank: Learning to schedule functions in serverless platforms,” in 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2021, pp. 31–40.

[14] H. Yu, H. Wang, J. Li, and S.-J. Park, “Harvesting idle resources in serverless computing via reinforcement learning,” arXiv preprint arXiv:2108.12717, 2021.

[15] W. Weng, X. Zhou, and R. Srikant, “Optimal load balancing in bipartite graphs,” arXiv preprint arXiv:2008.08380, 2020.

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

[17] 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.

[18] D. Ruten and D. Mukherjee, “Load balancing under strict constraints: Inefficiency, optimality, and k-competitiveness,” Applied Probability, pp. 406–413, 1978.

[19] 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.

[20] 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.

[21] S. Fouladi, R. S. Wahby, B. Shacklett, K. V. Balasubramanian, W. Zeng, R. Blahera, A. Sivaraman, G. Porter, and K. Winston, “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.

[22] E. Jonas, J. Schleier-Smith, V. Sriekanti, 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.

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

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

[25] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, “An axiomatic theory of fairness in network resource allocation,” in 2010 Proceedings IEEE INFOCOM, 2010, pp. 1–9.

[26] Amazon Web Services, Inc., “Amazon elastic transcoder pricing,” https://aws.amazon.com/elastictranscoder/pricing/, 2022.

[27] Google Cloud, “Pricing details for the transcoder api,” https://cloud.google.com/transcoder/pricing, 2022.

[28] Alibaba Cloud, “Apsaravideo for media processing,” https://www.alibabacloud.com/product/mts/pricing, 2022.

[29] Tencent Cloud, “Video processing pricing,” https://cloud.tencent.com/product/mts/pricing, 2022.

[30] J. Puchinger, 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.

[31] 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.

[32] Y. Zhou, A. Zomaya, 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.
of IEEE.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

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

[40] 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.

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

[42] “Openwhisk: Open source serverless cloud platform,” https://openwhisk.apache.org/.

[43] S. Shpait, 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.

[44] 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.

[45] V. Bilb, 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.

Hailiang Zhao received the B.S. degree in 2019 from the school of computer science and technology, Wuhan University of Technology, Wuhan, China. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include cloud & edge computing, distributed systems and optimization algorithms. He has published several papers in flagship conferences and journals including IEEE ICWS 2019, IEEE TPSD, IEEE TMC, etc. He has been a recipient of the Best Student Paper Award of IEEE ICWS 2019. He is a reviewer for IEEE TSC and Internet of Things Journal.

Shuiguang Deng is currently a full professor at the College of Computer Science and Technology in Zhejiang University, China, where he received a BS and PhD degree both in Computer Science in 2002 and 2007, respectively. He previously worked at the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include Edge Computing, Service Computing, Cloud Computing, and Business Process Management. He serves for the journal IEEE Transactions on Services Computing, Knowledge and Information Systems, Computing, and IET Cyber-Physical Systems: Theory & Applications as an Associate Editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TSCVC. He is a fellow of IET and a senior member of IEEE.

Jianwei Yin received the Ph.D. degree in computer science from Zhejiang University (ZJU) in 2001. He was a Visiting Scholar with the Georgia Institute of Technology. He is currently a Full Professor with the College of Computer Science, ZJU. Up to now, he has published more than 100 papers in top international journals and conferences. His current research interests include service computing and business process management. He is an Associate Editor of the IEEE Transactions on Services Computing.

Schahram Dustdar is a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group at the TU Wien. He is founding co-Editor-in-Chief of ACM Transactions on Internet of Things (ACM TIoT) as well as Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Computing Surveys, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. Dustdar is recipient of multiple awards: TCI Distinguished Service Award (2021), IEEE TCSVC Outstanding Leadership Award (2018), IEEE TCSC Award for Excellence in Scalable Computing (2019), ACM Distinguished Scientist (2009), ACM Distinguished Speaker (2021), IBM Faculty Award (2012). He is an elected member of the Academia Europaea: The Academy of Europe, where he is chairman of the Informatics Section, as well as an IEEE Fellow (2016), an Asia-Pacific Artificial Intelligence Association (AAIA) President (2021) and Fellow (2021). He is an EAI Fellow (2021) and an I2CICC Fellow (2021). He is a Member of the 2022 IEEE Computer Society Fellow Evaluating Committee (2022).

Albert Y. Zomaya is the Peter Nicol Russell Chair Professor of Computer Science and Director of the Centre for Distributed and High Performance Computing at the University of Sydney. To date, he has published > 600 scientific papers and articles and is (co-)author/editor of > 30 books. A sought-after speaker, he has delivered > 250 keynote addresses, invited seminars, and media briefings. His research interests span several areas in parallel and distributed computing and complex systems. He is currently the Editor in Chief of the ACM Computing Surveys and served in the past as Editor in Chief of the IEEE Transactions on Computers (2010-2014) and the IEEE Transactions on Sustainable Computing (2016-2020).

Professor Zomaya is a decorated scholar with numerous accolades including Fellowship of the IEEE, the American Association for the Advancement of Science, and the Institution of Engineering and Technology (UK). Also, he is an Elected Fellow of the Royal Society of New South Wales and an Elected Foreign Member of Academia Europaea. He is the recipient of the 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian Science, the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), IEEE Computer Society Technical Achievement Award (2014), ACM Mawii Reginald A. Fessenden Award (2017), the New South Wales Premier’s Prize of Excellence in Engineering and Information and Communications Technology (2018), and the Research Innovation Award, IEEE Technical Committee on Cloud Computing (2021).