Online Control of Cloud and Edge Resources Using Inaccurate Predictions

Lei Jiao[1], Antonia Tulino[2,3], Jaime Llorca[2], Yue Jin[2], Alessandra Sala[2], Jun Li[1]
[1]University of Oregon [2]Nokia Bell Labs [3]University of Naples Federico II

***Abstract*—We study cloud resource control in the global-local distributed cloud infrastructure. We firstly model and formulate the problem while capturing the multiple challenges such as the inter-dependency between resources and the uncertainty in the inputs. We then propose a novel online algorithm which, via the regularization technique, decouples the original problem into a series of subproblems for individual time slots and solves both the subproblems and the original problem over every prediction time window to jointly make resource allocation decisions. Compared against the offline optimum with accurate inputs, our approach maintains a provable parameterized worst-case performance gap with only inaccurate inputs under certain conditions. Finally, we conduct evaluations with large-scale, real-world data traces and show that our solution outperforms existing methods and works efficiently with near-optimal cost in practice.**

I. Introduction

The cloud infrastructure begins to incorporate distributed local clouds at the network edge [10], in addition to global centralized clouds at the network core. Allocating resources in this complex infrastructure to accommodate the time-varying service workload faces essential challenges. First of all, resources often need to be allocated jointly across geographic locations. User demands or requests, originating from diverse regions, reach the edge clouds for latency or privacy related processing, and then proceed to the core clouds for the key business logic related processing. This entails allocating resources at each cloud en route, taking into account request distribution, resource prices, and cloud capacities. Moreover, resources are often inter-dependent since the allocation of one resource relies on the allocation of others [13], [15]. For example, Virtual Machines (VMs) are provisioned to run services; servers are provisioned to host VMs; on-site electricity generators, when necessary, need to be provisioned to power servers; all such resources need to be allocated in an coordinated manner for the service workload. Finally, resource allocation is not a one-time transaction, and often needs to be reconfigured repeatedly on the fly with limited or inaccurate knowledge of future inputs [9]. Especially, regardless of workload variations, changing resource allocations frequently over time influences service quality, reliability, and performance, and can further result in service credibility damage and revenue loss [16].

Existing research falls insufficient in this scenario. Dynamic resource allocation with reconfigurations has only been considered for "flat" distributed clouds with no global-local structure [12]. Most predictive online algorithms can use accurate predictions, but cannot leverage inaccurate predictions while maintaining the performance guarantees [11], [13], [15]; those that explore inaccurate predictions have never captured the resource inter-dependency complexities [6].

In this paper, we study this cloud resource control problem while capturing and addressing the aforementioned challenges. We model and formulate this problem. Our allocation cost models, based on affine functions, capture the usage cost of resources with arbitrarily varying prices; our reconfiguration cost models adopt non-linear functions to account for the cost of transitioning between successive resource allocations [12]. We model the inaccurate predictions of the dynamic inputs, such as resource prices and service workloads, as falling in a range [9], [18] relative to the dynamic real values, without assuming how they may vary within that range.

After revisiting the standard Fixed Horizon Control (FHC) and Model Predictive Control (MPC) algorithms and the state-of-the-art Averaging Fixed Horizon Control (AFHC) algorithm [6], [12], we leverage the regularization technique [5] to design a novel Regularization-Assisted Control (RAC) algorithm. RAC decouples the original offline problem into a series of regularized subproblems solvable online at individual time slots, and solves such subproblems and the original problem altogether over the same prediction time window to jointly determine the resource allocation. We can formally prove that RAC, which only takes the inaccurate inputs, provides parameterized finite-constant worst-case guarantees for our resource control problem under certain conditions, compared to the offline optimum which takes all accurate inputs.

We conduct extensive evaluations using real-world locations of the 19 Equinix US data centers as global core clouds [2] and the most populous 101 US cities as local edge clouds [1]. With the real-world 400-hour Google data center workload [3], Amazon EC2 spot VM prices and the wholesale electricity prices [14], we run FHC, MPC, AFHC, and RAC to minimize the operational cost of resource usage plus the revenue loss incurred by reconfigurations, using inaccurate predictions of the inputs. We find the following: (1) RAC outperforms all its three counterparts and achieves near-optimal cost; (2) RAC positions itself as much superior to others when predictions are only available for the near future, as is often the case in reality; (3) RAC is robust, least influenced by the amount of prediction noise while providing consistent cost savings; (4) RAC incurs moderate additional average execution time compared to FHC, and is faster than MPC and AFHC whose average execution time grows almost linearly with the prediction window size.

Due to the page limit, we omit all the proofs of the lemmas and the theorems in this paper.

978-1-5386-2542-2/18/$31.00 @2018 IEEE

II. Model Formulation

A. Models

Cloud and SLA: We model a system composed of a set of core clouds, denoted by $\mathcal{J}$, and a set of edge clouds, denoted by $\mathcal{I}$, at distributed geographic locations. All clouds are connected via carrier networks or the Internet. To meet the Service Level Agreement (SLA), we assume for requests coming from the edge cloud $i \in \mathcal{I}$, only a subset of the core clouds, denoted by $\mathcal{J}_i \subseteq \mathcal{J}$, can be used for further processing, which can be determined by network delay, security risk, and so on. Accordingly, let $\mathcal{I}_j \subseteq \mathcal{I}$ denote the subset of the edge clouds for which the core cloud $j \in \mathcal{J}$ can satisfy the SLA.

Workload: We time-slot the system as $\mathcal{T} = \{1, 2, ..., T\}$. We use λ_{it} to denote the workload or the number of user requests received at the edge cloud i in the time slot $t \in \mathcal{T}$. The workload at each edge cloud is incurred by all the users in the corresponding geographic region.

Resource, Price, and Revenue Loss: We model two types of resources, while our models can be indeed generalized to more types of resources with complex dependencies. The two types of resources can be VMs and servers, or servers and on-site electricity generators, where the first type of resource directly handles or "covers" the workload, and the second type of resource powers or "covers" the first type of resource. Consider VMs and servers. Let a_{it}, $\forall i \in \mathcal{I}$, $\forall t \in \mathcal{T}$ be the resource price, or the unit operational cost per VM per time slot, at cloud i in time slot t and let b_i, $\forall i \in \mathcal{I}$ be the reconfiguration price, or the unit reconfiguration cost, e.g., the revenue loss per reconfiguration incurred by service degradation while booting one more VM, at cloud i. Let e_{it}, $\forall i \in \mathcal{I}, \forall t \in \mathcal{T}$ be the unit operational cost for servers at cloud i in time slot t and let f_i, $\forall i \in \mathcal{I}$ be the unit reconfiguration cost for servers at cloud i. Similarly, we use c_{jt} and d_j to denote the unit operational and the unit reconfiguration cost for VMs at cloud j in time slot t, and use g_{jt} and h_j to denote the unit operational and the unit reconfiguration cost for servers at cloud j in time slot t. The unit operational cost captures software maintenance cost and hardware powering/cooling cost; the unit reconfiguration cost reflects revenue loss due to service degradation, SLA violation, or facility wear and tear.

Resource Cap: By X_{ij} and Z_{ij}, we represent the caps for VMs and servers, i.e., the maximum numbers of the available VMs and servers that can be allocated at edge cloud i to handle the workload going towards the core cloud $j \in \mathcal{J}_i$, respectively; by Y_{ij} and W_{ij}, we represent the caps for VMs and servers that can be allocated at core cloud j to handle the workload originating from the edge cloud $i \in \mathcal{I}_j$, respectively. Here we model the *cap*s for each resource at each cloud, where the sum of such caps at a cloud equals the capacity of that resource at that cloud. For example, Amazon EC2 often has multiple "availability zones" co-located at a common location for fairness and reliability purposes, and our approach enables us to capture the capacity of each zone as its cap and model multiple co-located availability zones as one single cloud.

Inaccurate Prediction: We assume a bounded prediction model [9], [18]. We use $\mathcal{N}_t \triangleq \{\lambda_{it}, \forall i\} \cup \{a_{it}, e_{it}, \forall i\} \cup \{c_{jt}, g_{jt}, \forall j\}$ to denote the actual accurate time-varying inputs that appear in time slot t, and $\hat{\mathcal{N}}_t \triangleq \{\hat{\lambda}_{it}, \forall i\} \cup \{\hat{a}_{it}, \hat{e}_{it}, \forall i\} \cup \{\hat{c}_{jt}, \hat{g}_{jt}, \forall j\}$ to denote the corresponding predicted inputs for time slot t. We focus on a specific case of $\hat{\mathcal{N}}_t$, where $\delta_l \mathcal{N}_t \leq \hat{\mathcal{N}}_t \leq \delta_u \mathcal{N}_t, \forall t$, $0 < \delta_l \leq 1$, and $\delta_u \geq 1$. That is, every element of $\hat{\mathcal{N}}_t$ is lower-bounded and upper-bounded by the corresponding element of $\mathcal{N}_t$ times δ_l and δ_u, respectively. We assume δ_l and δ_u are known in prior. We also use $\hat{\mathcal{N}}_t \backslash \{\hat{\lambda}_{it}, \forall i\}$ to denote the predicted inputs excluding the workload, and use $\mathcal{N}_n \triangleq \{b_i, f_i, \forall i\} \cup \{d_j, h_j, \forall j\}$ to denote the nonvariant inputs.

Allocation and Reconfiguration: Resource allocations are our variables. We use x_{ijt} to represent the number of VMs allocated at edge cloud i to handle the requests that are sent to core cloud j in time slot t, and y_{ijt} to represent the number of VMs allocated at core cloud j to handle these same requests that are sent from edge cloud i in time slot t. For servers, we use z_{ijt} to represent the resources allocated at edge cloud i to "cover" x_{ijt} in t, and w_{ijt} to represent the resources allocated at core cloud j to "cover" y_{ijt} in t. The reconfiguration is computed as $(m-n)^+ \triangleq \max\{m-n, 0\}$, where n is the amount of resources allocated in one time slot and m is the amount of resources allocated in the next time slot.

B. Problem Formulation

We let $\mathbf{P}$, with the objective P, denote our resource control problem that has the accurate knowledge of the inputs. We let $\hat{\mathbf{P}}$, with the objective $\hat{P}$, denote our resource control problem that only has inaccurate predictions of the inputs. We only present the formulation of $\hat{\mathbf{P}}$; replacing $\{\hat{a}_{it}, \hat{e}_{it}, \hat{c}_{jt}, \hat{g}_{jt}, \hat{\lambda}_{it}\}$ by $\{a_{it}, e_{it}, c_{jt}, g_{jt}, \lambda_{it}\}$ would produce the formulation of $\mathbf{P}$. $\hat{\mathbf{P}}$ and $\mathbf{P}$ are related in the sense that the predicted inputs are bounded by constants times the actual inputs.

$$
\begin{aligned}
\min \quad & \hat{P} = \textstyle\sum_t \hat{P}_t = \sum_t \sum_i \sum_{j \in \mathcal{J}_i} \left(\hat{a}_{it} x_{ijt} + \hat{c}_{jt} y_{ijt}\right) \\
& + \textstyle\sum_t \sum_i b_i \Big(\sum_{j \in \mathcal{J}_i} x_{ijt} - \sum_{j \in \mathcal{J}_i} x_{ijt-1} \Big)^+ \\
& + \textstyle\sum_t \sum_j d_j \Big(\sum_{i \in \mathcal{I}_j} y_{ijt} - \sum_{i \in \mathcal{I}_j} y_{ijt-1} \Big)^+ \\
& + \textstyle\sum_t \sum_i \sum_{j \in \mathcal{J}_i} \left(\hat{e}_{it} z_{ijt} + \hat{g}_{jt} w_{ijt}\right) \\
& + \textstyle\sum_t \sum_i f_i \Big(\sum_{j \in \mathcal{J}_i} z_{ijt} - \sum_{j \in \mathcal{J}_i} z_{ijt-1} \Big)^+ \\
& + \textstyle\sum_t \sum_j h_j \Big(\sum_{j \in \mathcal{I}_j} w_{ijt} - \sum_{j \in \mathcal{I}_j} w_{ijt-1} \Big)^+ \\
\text{s.t.} \quad & s_{ijt} \leq x_{ijt} \leq X_{ij}, \quad \forall j \in \mathcal{J}_i, \forall i, \forall t, && \text{(1a)} \\
& s_{ijt} \leq y_{ijt} \leq Y_{ij}, \quad \forall j \in \mathcal{J}_i, \forall i, \forall t, && \text{(1b)} \\
& \textstyle\sum_{j \in \mathcal{J}_i} s_{ijt} \geq \hat{\lambda}_{it}, \quad \forall i, \forall t, && \text{(1c)} \\
& s_{ijt} \geq 0, \quad \forall j \in \mathcal{J}_i, \forall i, \forall t, && \text{(1d)} \\
& x_{ijt} \leq z_{ijt} \leq Z_{ij}, \quad \forall j \in \mathcal{J}_i, \forall i, \forall t, && \text{(1e)} \\
& y_{ijt} \leq w_{ijt} \leq W_{ij}, \quad \forall j \in \mathcal{J}_i, \forall i, \forall t. && \text{(1f)}
\end{aligned}
$$

The objective captures the total operational cost plus the revenue loss incurred by reconfigurations for all edge clouds and core clouds over time. The first inequalities (i.e., the first "$\leq$") in Constraints (1a) and (1b), together with Constraints (1c) and (1d), ensure the "workload cover". We introduce the auxiliary variable s_{ijt} to help capture this notion. The first inequalities in Constraints (1e) and (1f) ensure the "resource cover". The second inequalities (i.e., the second "$\leq$") in

Constraints (1a), (1b), (1e) and (1f) ensure that the allocated resources never exceed the corresponding cap. Note we assume $x_{ijt}=y_{ijt}=z_{ijt}=w_{ijt}=0$, $\forall j \in \mathcal{J}_i$, $\forall i$, $\forall t \leq 0$.

III. Algorithm and Analysis

A. Existing Control Algorithms

We introduce some additional notations. For accurate inputs, we have the objective $P = \sum_{t=1}^{T} P_t$, so for each t we use $\mathbf{P}_{t..t+w}$ to denote the problem defined over the time horizon $\{t, t+1, ..., t+w\}$ with the objective $P_{t..t+w} = \sum_{\tau=t}^{t+w} P_\tau$. Analogously, for predicted inputs, we have $\hat{\mathbf{P}}_{t..t+w}$ with the objective $\hat{P}_{t..t+w}$. We also use $\{\hat{\mathcal{X}}_t, \hat{\mathcal{X}}_{t+1}, ..., \hat{\mathcal{X}}_{t+w}\}$, where $\hat{\mathcal{X}}_t \triangleq \{\hat{x}_{ijt}, \hat{y}_{ijt}, \hat{z}_{ijt}, \hat{w}_{ijt}, \hat{s}_{ijt}, \forall j \in \mathcal{J}_i, \forall i\}$, to denote the optimal solution to $\hat{\mathbf{P}}_{t..t+w}$.

We revisit the existing control algorithms, including Fixed Horizon Control (FHC), Model Predictive Control (MPC), and Averaging Fixed Horizon Control (AFHC). FHC solves the problem in every prediction window and applies all the solutions; MPC re-solves the problem in the prediction window at every time slot and only applies the solution of the current time slot; AFHC takes the average of multiple FHC algorithms for each time slot. We formally describe these algorithms as below. Note the starting time slot for FHC and MPC is time slot 1, and the starting time slot for AFHC is time slot $-w+1$.

- FHC: At t, where $t=1, (w+1)+1, 2(w+1)+1$,...,we solve $\hat{\mathbf{P}}_{t..t+w}$ and apply the solution $\{\hat{\mathcal{X}}_t, \hat{\mathcal{X}}_{t+1}, ..., \hat{\mathcal{X}}_{t+w}\}$ to the time slots $\{t, t+1, ..., t+w\}$.
- MPC: At t, where $t = 1, 2, 3, ...$, we solve $\hat{\mathbf{P}}_{t...t+w}$ and acquire the solution $\{\hat{\mathcal{X}}_t, \hat{\mathcal{X}}_{t+1}, ..., \hat{\mathcal{X}}_{t+w}\}$, but only apply $\hat{\mathcal{X}}_t$ to t without applying others to the rest time slots.
- AFHC: We run $w+1$ FHC algorithms, starting at $t = -w+1, -w+2, ..., 1$, respectively, and at each $t \geq 1$, we take the average of all $\hat{\mathcal{X}}_t$s from the $w+1$ algorithms, and apply it to t.

B. Regularization-Assisted Control

We present our design of a novel algorithm that we name Regularization-Assisted Control (RAC). RAC solves the "regularized" one-shot problems sequentially in each prediction window, keeping the solution of the last time slot of every prediction window as the "anchor" and then solves the slice of the original problem (i.e., the one before regularization) over each prediction window except the anchor time slot. We describe RAC as two steps, with $w \geq 0$ as the size of the prediction window in terms of the number of time slots.

- RAC Step #1: At t, where $t=1, (w+1)+1, 2(w+1)+1$,...,we take the predicted inputs $\{\hat{\mathcal{N}}_t, \hat{\mathcal{N}}_{t+1}, ..., \hat{\mathcal{N}}_{t+w}\}$ to construct a series of regularized problems $\{\widetilde{\mathbf{P}}_t, \widetilde{\mathbf{P}}_{t+1}, ..., \widetilde{\mathbf{P}}_{t+w}\}$, and solve them sequentially so that we acquire the solutions $\{\widetilde{\mathcal{X}}_t, \widetilde{\mathcal{X}}_{t+1}, ..., \widetilde{\mathcal{X}}_{t+w}\}$.
- RAC Step #2: At t, where $t=1, (w+1)+1, 2(w+1)+1$,..., we solve $\hat{\mathbf{P}}_{t..t+w}$, with the objective $\hat{P}_{t..t+w} = \sum_{\tau=t}^{t+w} \hat{P}_\tau$, under the condition that the solution for the time slot $t+w$ is fixed as $\widetilde{\mathcal{X}}_{t+w}$. We then acquire the solutions $\{\hat{\mathcal{X}}_t, \hat{\mathcal{X}}_{t+1}, ..., \hat{\mathcal{X}}_{t+w-1}, \widetilde{\mathcal{X}}_{t+w}\}$ and apply them to the time slots $\{t, t+1, ..., t+w\}$, respectively.

Algorithm Comparison: FHC, MPC, and AFHC solve the original problem only; RAC engages both the regularized problem and the original problem over the same prediction window. For T time slots, FHC solves $\left\lceil \frac{T}{w+1} \right\rceil$ problems, MPC solves T problems, and AFHC solves $\sum_{n=0}^{w} \left\lceil \frac{T+n}{w+1} \right\rceil$ problems, all with each problem across $w+1$ time slots; RAC solves T regularized problems with each problem at a single time slot, plus $\left\lceil \frac{T}{w+1} \right\rceil$ problems with each problem across w time slots. Depending on T, the last few problems may have fewer than $w+1$ and w slots, respectively. Fig. 1 visualizes the case of $w=3$, where the bidirectional arrow indicates the length of a problem, the mark upon an arrow at a time slot implies that the solution obtained by solving the corresponding problem is applied to that time slot, and the mark across arrows means taking the average (only meaningful for AFHC).

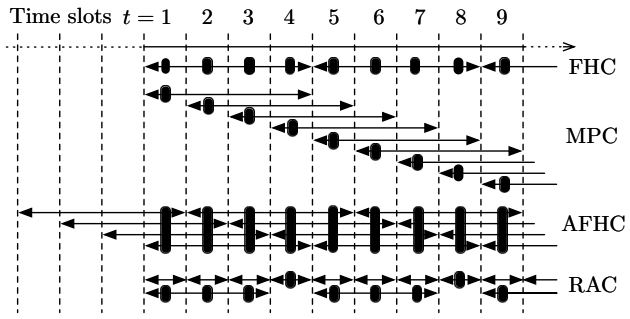

Fig. 1: Comparison of the four algorithms

The core of RAC is *regularization*, i.e., we replace the function $(\cdot)^+$ in the objective by a carefully-constructed logarithmic regularization function, and thus compose a regularized problem. The function we use is

$$R(x_t, \Delta) = \frac{1}{\ln\left(1+\frac{\Delta}{\varepsilon}\right)} \left((x_t + \varepsilon) \ln \frac{x_t+\varepsilon}{x_{t-1}+\varepsilon} - x_t \right),$$

where $x_t \leq \Delta, \forall t$, and $\varepsilon > 0$ is a configurable parameter. We apply this regularization function to $\hat{\mathbf{P}}_t$, the one-shot time slice of the problem $\hat{\mathbf{P}}$ defined at the time slot t, and transform it into the following regularized, convex problem $\widetilde{\mathbf{P}}_t$:

$$\begin{aligned} \min \quad & \widetilde{P}_t = \textstyle\sum_i \sum_{j\in\mathcal{J}_i} \left(\hat{a}_{it} x_{ijt} + b_i R\left(x_{ijt}, X_{ij}\right)\right) \\ & + \textstyle\sum_i \sum_{j\in\mathcal{J}_i} \left(\hat{c}_{jt} y_{ijt} + d_j R\left(y_{ijt}, Y_{ij}\right)\right) \\ & + \textstyle\sum_i \sum_{j\in\mathcal{J}_i} \left(\hat{e}_{it} z_{ijt} + f_i R\left(z_{ijt}, Z_{ij}\right)\right) \\ & + \textstyle\sum_i \sum_{j\in\mathcal{J}_i} \left(\hat{g}_{jt} w_{ijt} + h_j R\left(w_{ijt}, W_{ij}\right)\right) \\ \text{s.t.} \quad & \text{(1a)} \sim \text{(1f)}, \text{without "}\forall t\text{".}^1 \end{aligned} \tag{2a}$$

Besides, the following theorem enables us to use the optimal solution to $\widetilde{\mathbf{P}}_t$ as a solution to $\hat{\mathbf{P}}_t$:

Theorem 1. *$\widetilde{\mathcal{X}}_t$, the optimal solution to $\widetilde{\mathbf{P}}_t$, is feasible for $\hat{\mathbf{P}}_t$.*

[1]In fact, (1a), (1b), (1e) and (1f) are known as the "box constraints", and here we need to transform them to the corresponding "knapsack constraints" via the technique described in Section 2.2 of Reference [5].

C. Competitive Analysis

To derive the competitive ratio r of RAC, we establish the following chain of inequalities:

$$P(\{\hat{\mathcal{X}}_1, ..., \hat{\mathcal{X}}_w, \widetilde{\mathcal{X}}_{w+1}, \hat{\mathcal{X}}_{w+2}, ..., \hat{\mathcal{X}}_{2w+1}, \widetilde{\mathcal{X}}_{2w+2}, ...\}) \quad (3a)$$
$$\leq r_3 \hat{P}(\{\hat{\mathcal{X}}_1, ..., \hat{\mathcal{X}}_w, \widetilde{\mathcal{X}}_{w+1}, \hat{\mathcal{X}}_{w+2}, ..., \hat{\mathcal{X}}_{2w+1}, \widetilde{\mathcal{X}}_{2w+2}, ...\}) \quad (3b)$$
$$\leq r_2 r_3 \hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\}) \quad (3c)$$
$$\leq r_1 r_2 r_3 \hat{P}_{opt} \quad (3d)$$
$$\leq r_1 r_2 r_3 r_4 P_{opt}. \quad (3e)$$

We thus have $r = r_1 r_2 r_3 r_4$. We demonstrate that $r_1 = 1 + \max\{C(\{X_{ij}\}), C(\{Y_{ij}\}), C(\{Z_{ij}\}), C(\{W_{ij}\})\}$, $r_2 = r_3 = 1$, and $r_4 = \delta_u^2/\delta_l^2$, where $C(\{\Delta_{ij}\}) = \max_i |\mathcal{J}_i| \max_{i,j}\{(\Delta_{ij} + \varepsilon)\ln(1 + \frac{\Delta_{ij}}{\varepsilon})\}$, and $\varepsilon > 0$ is a configurable parameter.

We break our chain of inequalities into multiple parts. Using (3d) to bound (3c), we connect "online" and "offline", which is achieved via Theorem 2. Using (3c) to bound (3b), we connect "hyperopic" and "myopic", which is achieved via Theorem 3. Using (3b) to bound (3a), and (3e) to bound (3d), we connect "predicted" and "actual", which is achieved via Theorem 4.

To connect "online" and "offline", we reformulate $\hat{\mathbf{P}}$ as $\bar{\mathbf{P}}$ and derive the Lagrange dual problem $\bar{\mathbf{D}}$ for $\bar{\mathbf{P}}$. Using $\bar{P}$ and $\bar{D}$ to denote their objectives, respectively, we intrinsically have $\bar{D}(M(\{\widetilde{\mathcal{X}}_t, \forall t\})) \leq \bar{P}_{opt} = \hat{P}_{opt}$, due to weak duality, assuming M is a mapping that can map $\{\widetilde{\mathcal{X}}_t, \forall t\}$ to a feasible solution to $\bar{\mathbf{D}}$. Our main task becomes proving $\hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\}) \leq r_1 \bar{D}(M(\{\widetilde{\mathcal{X}}_t, \forall t\}))$ and also finding M. Below, we present the problems $\bar{\mathbf{P}}$ and $\bar{\mathbf{D}}$, the mapping M, and the lemmas and the theorem that establish $\hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\}) \leq r_1 \bar{D}(M(\{\widetilde{\mathcal{X}}_t, \forall t\}))$.

1) Formulation of the Problem $\bar{\mathbf{P}}$: To transform $\hat{\mathbf{P}}$ to $\bar{\mathbf{P}}$, we replace the functions $(\cdot)^+$ by new additional variables and add the additional constraints associated with such replacements.

$$\begin{aligned}
\min \quad & \bar{P} = \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{a}_{it} x_{ijt} + \sum_t \sum_i b_i u_{it} \\
& + \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{c}_{jt} y_{ijt} + \sum_t \sum_j d_j v_{jt} \\
& + \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{e}_{it} z_{ijt} + \sum_t \sum_i f_i m_{it} \\
& + \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{g}_{jt} w_{ijt} + \sum_t \sum_j h_j n_{jt} \\
\text{s.t.} \quad & (2a), \forall t, \\
& u_{it} \geq \textstyle\sum_{j\in\mathcal{J}_i} x_{ijt} - \sum_{j\in\mathcal{J}_i} x_{ijt-1}, \forall i, \forall t, \quad (4a) \\
& v_{jt} \geq \textstyle\sum_{i\in\mathcal{I}_j} y_{ijt} - \sum_{i\in\mathcal{I}_j} y_{ijt-1}, \forall j, \forall t, \quad (4b) \\
& m_{it} \geq \textstyle\sum_{j\in\mathcal{J}_i} z_{ijt} - \sum_{j\in\mathcal{J}_i} z_{ijt-1}, \forall i, \forall t, \quad (4c) \\
& n_{jt} \geq \textstyle\sum_{i\in\mathcal{I}_j} w_{ijt} - \sum_{i\in\mathcal{I}_j} w_{ijt-1}, \forall j, \forall t, \quad (4d) \\
& u_{it} \geq 0, m_{it} \geq 0, \forall i, \forall t, \quad (4e) \\
& v_{jt} \geq 0, n_{jt} \geq 0, \forall j, \forall t. \quad (4f)
\end{aligned}$$

2) Formulation of the Problem $\bar{\mathbf{D}}$: We derive the Lagrange dual problem $\bar{\mathbf{D}}$ for $\bar{\mathbf{P}}$. Using Greek letters to denote the dual variables and omitting the constraints to save space, we have the objective function $\bar{D} = \bar{D}_\gamma + \bar{D}_\theta + \bar{D}_\mu + \bar{D}_\pi + \bar{D}_\rho$, where

$$\begin{aligned}
\bar{D}_\gamma &= \textstyle\sum_t \sum_i \hat{\lambda}_{it} \gamma_{it}, \\
\bar{D}_\theta &= \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} (\hat{\lambda}_{it} - X_{ij}) \theta_{ijt}, \\
\bar{D}_\mu &= \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} (\hat{\lambda}_{it} - Y_{ij}) \mu_{ijt}, \\
\bar{D}_\pi &= \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} (\hat{\lambda}_{it} - Z_{ij}) \pi_{ijt}, \\
\bar{D}_\rho &= \textstyle\sum_t \sum_i \sum_{j\in\mathcal{J}_i} (\hat{\lambda}_{it} - W_{ij}) \rho_{ijt}.
\end{aligned}$$

3) Mapping: We construct a mapping M to map $\widetilde{\mathcal{X}}_t$ to a corresponding feasible solution at t to the problem $\bar{\mathbf{D}}$. More specifically, for the dual variables σ_{it}, τ_{jt}, ϕ_{it} and ψ_{jt} at t that correspond to (4a), (4b), (4c) and (4d), we have $\sigma_{it} = b_i M'(\widetilde{x}_{ijt}, X_{ij})$, $\tau_{jt} = d_j M'(\widetilde{y}_{ijt}, Y_{ij})$, $\phi_{it} = f_i M'(\widetilde{z}_{ijt}, Z_{ij})$ and $\psi_{jt} = h_j M'(\widetilde{w}_{ijt}, W_{ij})$, where the function $M'(\cdot,\cdot)$ is defined as $M'(x_t, \Delta) = \frac{1}{\ln(1+\frac{\Delta}{\varepsilon})} \ln \frac{\Delta+\varepsilon}{x_{t-1}+\varepsilon}$. For the rest of the variables of $\bar{\mathbf{D}}$ at t, we simply let them equal the corresponding dual variables of $\widetilde{\mathbf{P}}_t$. While having its optimal solution $\widetilde{\mathcal{X}}_t$, $\widetilde{\mathbf{P}}_t$ also has its Lagrange dual problem and the dual solution. Note that the constraints of $\widetilde{\mathbf{P}}_t$ have an obvious one-to-one mapping to the constraints of $\bar{\mathbf{P}}$, except (4a), (4b), (4c) and (4d). Thus, except these constraints, there is a direct, one-to-one mapping between the variables of $\widetilde{\mathbf{P}}_t$ and the variables of $\bar{\mathbf{D}}$.

All the above constitute the mapping M. By leveraging $\widetilde{\mathbf{P}}_t$'s Karush-Kuhn-Tucker (KKT) conditions it can be verified that these constructed variables for $\bar{\mathbf{D}}$ satisfy the constraints of $\bar{\mathbf{D}}$.

4) Bounding: We put $\{\hat{\mathcal{X}}_t, \forall t\}$ into $\hat{P}$ and put $M(\{\hat{\mathcal{X}}_t, \forall t\})$ into $\bar{D}$. We can then bound the former by a constant times the latter. We can actually derive the following results.

Lemma 1. *The operational cost in $\hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\})$ is bounded by $\bar{D}$, i.e., $\sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{a}_{it} \widetilde{x}_{ijt} + \sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{c}_{jt} \widetilde{y}_{ijt} + \sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{e}_{it} \widetilde{z}_{ijt} + \sum_t \sum_i \sum_{j\in\mathcal{J}_i} \hat{g}_{jt} \widetilde{w}_{ijt} \leq \bar{D}$.*

Lemma 2. *The revenue loss in $\hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\})$ is also bounded:*

$$\begin{aligned}
&\textstyle\sum_t \sum_i b_i (\sum_{j\in\mathcal{J}_i} \widetilde{x}_{ijt} - \sum_{j\in\mathcal{J}_i} \widetilde{x}_{ijt-1})^+ \leq C(\{X_{ij}\})(\bar{D}_\gamma + \bar{D}_\theta), \\
&\textstyle\sum_t \sum_j d_j (\sum_{i\in\mathcal{I}_j} \widetilde{y}_{ijt} - \sum_{i\in\mathcal{I}_j} \widetilde{y}_{ijt-1})^+ \leq C(\{Y_{ij}\})(\bar{D}_\gamma + \bar{D}_\mu), \\
&\textstyle\sum_t \sum_i f_i (\sum_{j\in\mathcal{J}_i} \widetilde{z}_{ijt} - \sum_{j\in\mathcal{J}_i} \widetilde{z}_{ijt-1})^+ \leq C(\{Z_{ij}\})(\bar{D}_\gamma + \bar{D}_\theta + \bar{D}_\pi), \\
&\textstyle\sum_t \sum_j h_j (\sum_{i\in\mathcal{I}_j} \widetilde{w}_{ijt} - \sum_{i\in\mathcal{I}_j} \widetilde{w}_{ijt-1})^+ \leq C(\{W_{ij}\})(\bar{D}_\gamma + \bar{D}_\mu + \bar{D}_\rho),
\end{aligned}$$

where $C(\{\Delta_{ij}\}) = \max_i |\mathcal{J}_i| \max_{i,j}\{(\Delta_{ij} + \varepsilon)\ln(1 + \frac{\Delta_{ij}}{\varepsilon})\}$.

Theorem 2. *$\hat{P}(\{\widetilde{\mathcal{X}}_t, \forall t\}) \leq r_1 \bar{D}(M(\{\widetilde{\mathcal{X}}_t, \forall t\})) \leq r_1 \bar{P}_{opt} = r_1 \hat{P}_{opt}$. Using $C(\cdot)$ defined as above, we have $r_1 = 1 + \max\{C(\{X_{ij}\}), C(\{Y_{ij}\}), C(\{Z_{ij}\}), C(\{W_{ij}\})\}$.*

Remark: Lemmas 1 and 2 are proved via $\widetilde{\mathbf{P}}_t$'s KKT conditions. Also, we do not expect a large $|\mathcal{J}_i|$, e.g., Amazon CloudFront uses the closest core cloud only and so $|\mathcal{J}_i| = 1, \forall i$; the resource caps do not have to be the absolute numbers of VMs or servers, and can be normalized as the fractions of the entire capacity, where the workload should also be measured by the relative amount of resources needed rather than the absolute number of user requests. Thus, r_1 can be quite a reasonable value larger than 1 by setting an appropriate $\varepsilon > 0$.

To complete the derivation, we present Theorems 3 and 4, which are proved by some basic algebra and transformations:

Theorem 3. *Given feasible solutions $\{\mathcal{X}_1, ..., \mathcal{X}_T\}$ to $\hat{\mathbf{P}}$ and integers τ, κ, where $1 \leq \tau < \kappa \leq T$, we have the following inequality: $\hat{P}(\{\mathcal{X}_1, ..., \mathcal{X}_{\tau-1}, \hat{\mathcal{X}}_\tau, \hat{\mathcal{X}}_{\tau+1}, ..., \hat{\mathcal{X}}_{\kappa-1}, \mathcal{X}_\kappa, ..., \mathcal{X}_T\}) \leq \hat{P}(\{\mathcal{X}_1, ..., \mathcal{X}_T\})$, where $\{\hat{\mathcal{X}}_\tau, \hat{\mathcal{X}}_{\tau+1}, ..., \hat{\mathcal{X}}_{\kappa-1}, \mathcal{X}_\kappa\}$ minimizes $\hat{\mathbf{P}}_{\tau..\kappa}$, with the objective function $\hat{P}_{\tau..\kappa} = \sum_{t=\tau}^{\kappa} \hat{P}_t$. Thus,*

setting $\tau = (n-1)(w+1)+1$, $\kappa = n(w+1)$, $n = 1, 2, ...$, *we have* $\hat{P}(\{\hat{\mathcal{X}}_1, ..., \hat{\mathcal{X}}_w, \widetilde{\mathcal{X}}_{w+1}, \hat{\mathcal{X}}_{w+2}, ..., \hat{\mathcal{X}}_{2w+1}, \widetilde{\mathcal{X}}_{2w+2}, ...\}) \leq \hat{P}(\{\hat{\mathcal{X}}_t, \forall t\})$.

Theorem 4. *Using* $\hat{\mathcal{N}}_t/\delta_l$ *as the predicted inputs for* t, *we have* $P(\{\hat{\mathcal{X}}_1, ..., \hat{\mathcal{X}}_w, \widetilde{\mathcal{X}}_{w+1}, \hat{\mathcal{X}}_{w+2}, ..., \hat{\mathcal{X}}_{2w+1}, \widetilde{\mathcal{X}}_{2w+2}, ...\}) \leq \hat{P}(\{\hat{\mathcal{X}}_1, ..., \hat{\mathcal{X}}_w, \widetilde{\mathcal{X}}_{w+1}, \hat{\mathcal{X}}_{w+2}, ..., \hat{\mathcal{X}}_{2w+1}, \widetilde{\mathcal{X}}_{2w+2}, ...\})$. *Besides, if* $\hat{\mathcal{N}}_t \backslash \{\hat{\lambda}_{it}, \forall i\} \ll \mathcal{N}_n$, *then we have* $\hat{P}_{opt} \leq \delta_u^2/\delta_l^2 P_{opt}$.

IV. Evaluation

A. Evaluation Setup

Cloud, SLA, and Resource: For core clouds, we use the 19 locations of the Equinix data centers [2] in the US; for edge clouds, we use all cities in the continental US with a population larger than 200,000 in 2013 [1]. Among 111 such cities, 10 have been occupied by Equinix data centers, and we assume an edge cloud at each of the rest 101 locations. We find the geographically closest core cloud(s) for an edge cloud to form the SLA [14]. We consider servers and VMs.

Workload: We use the job submission trace at the Google data center in 2011 [3]. We aggregate the number of jobs on a per-hour basis and use the first 400 hours of the trace in our evaluations, as in Fig. 2. We treat one hour as one time slot. We replicate such workload at every edge cloud.

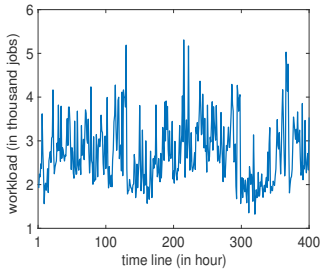

Fig. 2: Google data center trace

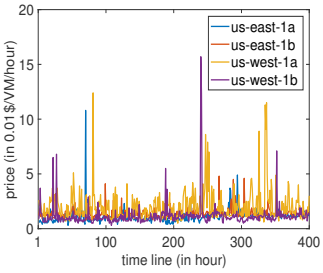

Fig. 3: Amazon spot VM price

Electricity Price: Electricity price is used as the unit operational cost for servers. For the cloud locations in a region covered by an RTO (Regional Transmission Organization), we synthesize the dynamic electricity prices, following the means and the standard deviations of the hourly wholesale electricity markets of that RTO in 2009 [14]. For any cloud not covered by an RTO, we assume it attaches to the same RTO as its geographically closest cloud that has an RTO.

VM Price: VM price is used to calculate the unit operational cost for VMs. We recorded Amazon EC2's ".xlarge" spot VM price every few minutes for the "US East (N. Virginia)" and "US West (Northern California)" clouds in November 2014. For either of them, we have two series of dynamic, hourly-averaged prices for two availability zones, shown in Fig. 3. For each of our cloud locations, we find the closer Amazon cloud and assign one of its two price series randomly. The VM "price" for cloud customers is actually not the VM "cost" for the cloud provider. Energy occupies 10%∼20% out of all the operational cost of a cloud data center [16]. We set 15%, and normalize our VM prices by our electricity data.

Unit Revenue Loss: Firstly, as different services may have different amount of revenue loss per reconfiguration, we vary the reconfiguration price of VMs in our evaluations, relative to the time-averaged VM price. Secondly, we set the reconfiguration price of servers as about one order of magnitude larger than that of VMs, roughly based on the time to boot a server versus the time to boot a VM. For simplicity, we set the same server reconfiguration price for all clouds and also the same VM reconfiguration price for all clouds.

Resource Cap: At each edge cloud, we set the capacity as 1.2 times its peak workload, and then we divide this capacity over the one or multiple core clouds, depending on the SLA, to set the VM cap for its workload going to each core cloud. At each core cloud, we set the VM cap for the incoming workload from each eligible edge cloud, depending on the SLA, as the same VM cap for that workload at that originating edge cloud. Corresponding to each VM cap, we calculate the server cap assuming every server hosts 50 VMs.

Inaccurate Prediction: We inject prediction noises to all our time-varying inputs, including the workload, electricity prices, and VM prices. For simplicity, we use the same parameter δ ($0 \leq \delta \leq 1$). At each time slot, we modify the accurate values of the workload and the prices by incorporating a noise value drawn from a Truncated Normal Distribution lower-bounded by the corresponding accurate value times $\delta_l = 1 - \delta$ and upper-bounded by that times $\delta_u = 1 + \delta$, with the mean as 0 and the standard deviation as δ.

B. Evaluation Results

We look at the "empirical competitive ratio" and the execution time. We implement FHC, MPC, AFHC, and RAC.

Influence of Unit Revenue Loss: Fig. 4 shows the total cost incurred when the prediction window equals 10 time slots and the noisy inputs fall within 0.9∼1.1 times the accurate inputs. RAC works superiorly to beat its counterparts. AFHC performs worse than MPC, which aligns with existing studies [6], [12], despite that it has better theoretical performance. As the revenue loss per reconfiguration grows, the total cost goes larger for all algorithms; RAC has consistent performance and becomes even more advantageous.

Influence of Prediction Window Size: Fig. 5 compares the total cost as the prediction window size varies, with the same noise as in Fig. 4. As the prediction window becomes larger, all algorithms move closer to the offline optimum with accurate inputs. The costs of FHC, MPC, and AFHC decrease by about 3.4, 3.6, and 3.9, respectively, as the prediction window grows from 2 to 10 time slots. RAC has the least cost.

Influence of Prediction Noise: Fig. 6 exhibits the total cost incurred as the predicted inputs contain different amount of noise. As δ increases, the predictions become less accurate. All algorithms have higher total cost compared to the offline optimum which has accurate inputs. RAC is robust and insensitive to the noise, as the corresponding cost changes much slower than others, increased by only about 0.5; MPC and AFHC behave better, increased by about 1.4 and 2.3, than FHC which is the most influenced, increased by about 2.8.

Execution Time: Fig. 7 depicts the average execution time for each algorithm to make resource allocation decisions for

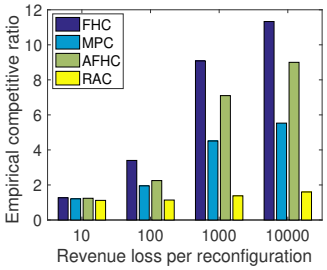

Fig. 4: Total costs for different reconfiguration prices

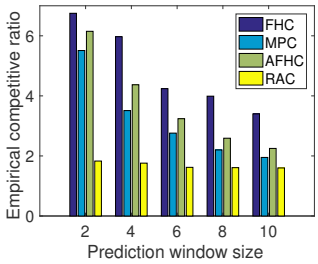

Fig. 5: Total costs for different prediction windows

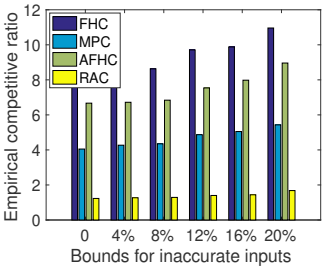

Fig. 6: Total costs for different prediction noises

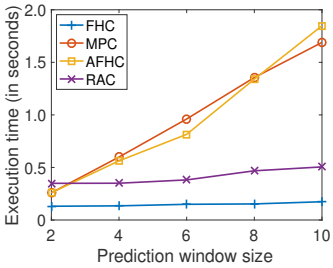

Fig. 7: Average execution time per time slot

one single time slot, invoking the interior point method as the optimization solver, on an up-to-date commodity computer. All algorithms complete in 2 seconds, much faster than booting a VM or server which can take tens of seconds. MPC and AFHC have more execution time, growing almost linearly with the prediction window size; FHC and RAC have stable execution time, and RAC has moderate time increase compared to FHC.

V. Related Work

For data centers, Lin et al. proposed AFHC [12] to toggle servers; Tu et al. [15] studied the online retrieving of electricity from both the power grid and the local generators; Jiao et al. [11] allocated and reconfigured resources for both unpredictable and perfectly predictable workload; Chen et al. [6] devised Committed Horizon Control for the noisy inputs.

For microgrids, Lu et al. [13] and Hajiesmaili et al. [9] investigated the intermittent energy and the co-generation of electricity and heat, and determined the on/off status and the output levels of energy sources to minimize the cost and to meet the predicted electricity and/or the heat demand.

For Virtual Network Functions (VNFs), Zhang et al. [17] and Fei et al. [8] employed online learning techniques to predict the traffic demand while minimizing the prediction errors, and developed online optimization algorithms using predicted demand for VNF purchase and/or deployment.

These works study important problems in various scenarios, but they are inapplicable to our problem in general. Most of them never consider the complicated resource dependency, except [9], [13], [15], let alone the edge-core cloud structure. They often assume accurate predictions and cannot capture inaccurate ones analytically, except [6], [8], [9], [17]. Those [9], [13], [15], [17] based on ski-rental problems with a fixed or bounded rental price fall short, as our problem has a varying, possibly unbounded "rental" price and also the "covering" constraints which could break the ski-rental-based performance guarantee [7]; it is also unclear whether or how the ski-rental idea could address the "covering" constraints. [6] is based on a statistical noise model of variances and correlations, different from our bounded prediction models. [17] and [8] jointly conduct prediction and optimization, and compare their algorithms against the offline optimization under the best static predictor. We do not compare to any predictor but to the accurate ground truth, which can be considered as a perfect dynamic predictor; the use of "regret" is also known to be incompatible with the competitive ratio in our paper [4].

VI. Conclusion

We study the online resource control problem in distributed global and local clouds. We design a novel predictive control algorithm that use the inaccurate predictions of future inputs on the fly with worst-case performance guarantees. We adopt real-world data to conduct evaluations and exhibit that our algorithm performs consistently superior to existing algorithms, incurring only negligible execution time overhead.

References

[1] "1000 Largest US Cities by Population with Geographic Coordinates," https://gist.github.com/Miserlou/c5cd8364bf9b2420bb29.

[2] "Equinix Acquires Verizon Data Center Colocation Facilities," http://www.equinix.com/equinix-acquires-verizon-data-centers/.

[3] "Google Cluster Data," https://github.com/google/cluster-data.

[4] L. L. H. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman, "A tale of two metrics: Simultaneous bounds on competitiveness and regret," in *ACL COLT*, 2013.

[5] N. Buchbinder, S. Chen, and J. S. Naor, "Competitive analysis via regularization," in *ACM-SIAM SODA*, 2014.

[6] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, "Using predictions in online optimization: Looking forward with an eye on the past," in *ACM SIGMETRICS*, 2016.

[7] L. Epstein and H. Zebedat-Haider, "Rent or Buy Problems with a Fixed Time Horizon," *Theory of Computing Systems*, vol. 56, no. 2, pp. 309–329, 2015.

[8] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive vnf scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM*, 2018.

[9] M. H. Hajiesmaili, C.-K. Chau, M. Chen, and L. Huang, "Online microgrid energy generation scheduling revisited: The benefits of randomization and interval prediction," in *ACM e-Energy*, 2016.

[10] I.-H. Hou, T. Zhao, S. Wang, and K. Chan, "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *ACM MOBIHOC*, 2016.

[11] L. Jiao, A. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed Online Resource Allocation in Multi-tier Distributed Cloud Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2556–2570, 2017.

[12] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *IEEE IGCC*, 2012.

[13] L. Lu, J. Tu, C.-K. Chau, M. Chen, and X. Lin, "Online energy generation scheduling for microgrids with intermittent energy sources and co-generation," in *ACM SIGMETRICS*, 2013.

[14] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the Electric Bill for Internet-Scale Systems," in *ACM SIGCOMM*, 2009.

[15] J. Tu, L. Lu, M. Chen, and R. K. Sitaraman, "Dynamic provisioning in next-generation data centers with on-site power production," in *ACM e-Energy*, 2013.

[16] C. Wang, B. Urgaonkar, A. Gupta, L. Y. Chen, R. Birke, and G. Kesidis, "Effective Capacity Modulation as an Explicit Control Knob for Public Cloud Profitability," in *IEEE ICAC*, 2016.

[17] X. Zhang, C. Wu, Z. Li, and F. C. Lau, "Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *IEEE INFOCOM*, 2017.

[18] S. Zhao, X. Lin, D. Aliprantis, H. N. Villegas, and M. Chen, "Online multi-stage decisions for robust power-grid operations under high renewable uncertainty," in *IEEE INFOCOM*, 2016.