A Provably-Efficient Online Algorithm for Re-Utilizing Unused VM Resources for Edge Providers

Qihang Sun
Dept. of Computer Science
University of Hong Kong
Hong Kong, China
qhsun@cs.hku.hk

Shaolei Ren
Dept. of Electrical and Computer Engineering
University of California, Riverside
Riverside, U.S.A.
sren@ece.ucr.edu

Chuan Wu
Dept. of Computer Science
University of Hong Kong
Hong Kong, China
cwu@cs.hku.hk

***Abstract*—In recent years, as a result of the rapidly growing volume of generated data, computation has been increasingly migrating from megascale data centers to Internet edges (a.k.a edge computing), for avoiding high latencies and overwhelmed bandwidths. Unlike centralized clouds, edge computing processes workloads generated by users nearby. Thus, due to the lack of statistical multiplexing from a large group of users, the resource demand at an edge data center exhibits more fluctuations, resulting in time-varying unused computation resources. In this paper, we propose our UNusEd spAred VM Re-uTilizing mecHanism, UNEARTH, to utilize different types of unused resources, such as storage, CPU, GPU, and so on, offered by an edge computing provider. Notably, the exact amount of unused VM resources is unknown before selling them. We evaluate the performance of our algorithms under realistic settings, showing that our proposed VM bundle allocation algorithm can achieve $(1+\frac{\Omega}{\Omega-1}\epsilon(eM^{\frac{1}{\Omega-1}}-1))$-approximation in the worst case compared with optimums; and overall, our algorithms outperform the existing and heuristic algorithms.**

***Index Terms*—edge computing, capacity prediction, resource allocation, approximation algorithm, regret analysis**

I. Introduction

In recent years, we have witnessed an explosive growth of distributed and latency-sensitive services, such as augmented reality (*e.g.*, Facebook AR camera), speech recognition (*e.g.*, Siri in iOS) and assisted driving. To meet their stringent latency requirements, deploying nearby computing infrastructures (a.k.a. edge nodes or edge computing) in proximity of end users instead of centralized remote data centers becomes a promising option [1]. For instance, edge nodes can be deployed inside base stations with compacted micro data centers structure, such as Vapor IO [2], thus delivering computation to Internet edges and achieving low end-to-end latency and high bandwidth. In Fig 1, we illustrate the hierarchy of edge nodes.

Edge nodes provide a cost-effective and scalable solution to edge service providers that aim at delivering services as fast and reliable as possible. Concretely, service providers can now rent edge computing resources on demand, instead of deploying their own servers everywhere [3]. On the one hand, edge nodes process workloads generated by users nearby to provide low-latency responses [4]. On the other hand, they also lack statistical multiplexing from a large group of users that would otherwise exist for centralized clouds. Thus, the resource demand at an edge data center exhibits more fluctuations, resulting in time-varying unused computation resources. For example, when there are more cars passing an intersection, more workloads are generated by these cars for assisted driving services deployed at an edge node; nonetheless, when there is little traffic at the intersection, there will be more idle resources.

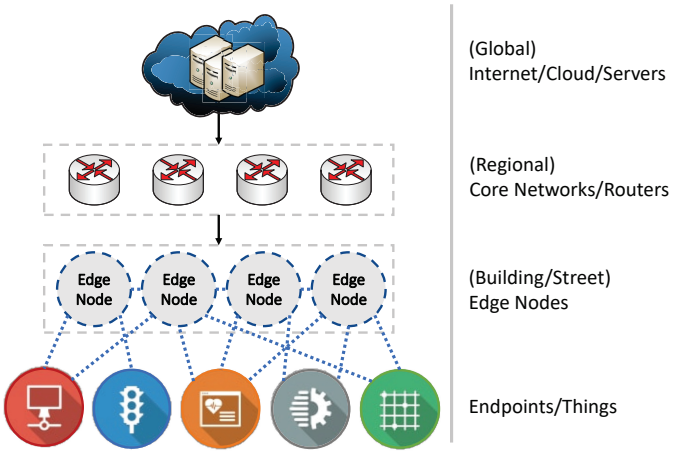

Fig. 1. The hierarchy of edge computing

Naturally, the unused resources can be re-used by other applications, if any, to increase the overall system utilization, which is a win-win solution for both service providers and edge node operators. Such resource re-allocation is already supported by the existing edge computing architecture. For example, European Telecommunications Standards Institute (ETSI) proposed an extension of the cloud computing paradigm in the edge of the network based on virtual machine (VM) technology [5]. Different resources, such as GPU, CPU, RAM, and so on, can be packed as different types of VM instances for ease of renting out for different applications. Nonetheless, the amount of unused resources can

be highly fluctuating and unknown in advance. Moreover, the unused resources can become unavailable at any time, when the workloads that reserve them increase. In other words, edge providers need to allocate unused VM resources without knowing the amount of available resources or any availability gurantee. These make the allocation of unused VMs very challenging.

Contributions of this work. In this paper, we design a novel **UN**us**E**d sp**A**red VM **R**e-u**T**ilizing mec**H**anism, called UNEARTH, to extract the different types of unused VM resources in edge nodes and allocate them to users as demand bundles. A salient feature of UNEARTH is that the VM resource capacity information is unknown before allocating. Such mechanism further improves the VM utilization of the edge computing provider and is well aligned with the emerging cloud computing trends [6], *i.e.*, the lower latency and the possibility of real-time applications. Our specific contributions are summarized as follows:

- We formulate and investigate the unused VM capacity allocation problem for edge computing providers, with the goal of maximizing the total valuation with accuracy capacity prediction. The innovation of this model lies in that: we consider an irrevocable VM bundle allocation under unknown resource capacity without prior knowledge of future signals.
- We analyze the proposed resource prediction algorithm (by rounding techniques) and prove its upper bound of the prediction error; moreover, we also analyze the proposed bundle allocation algorithm and prove its performance bound (by primal-dual) when allocating different types of VM resource to demand bundles.
- We compare our proposed mechanism, UNEARTH, with two heuristic methods and wide-used algorithms [7] and show that UNEARTH outperforms the benchmarks and achieves a performance ratio around 1.5.

II. Related Work

Edge computing is closely related to the offloading, which move the heavy computation workload to remote cloud data centers. This area has been investigated for more than a decade. In 2011, Chun and *et al.* proposed CloneCloud [8], which execute an application-level virtual machine as a device clone and run the programs in the remote computational clouds; however, sometimes it cannot run fast due to bandwidth and latency. In 2012, Kosta and *et al.* proposed, ThinkAir [9], a parallel version of a mobile offloading mechanism by utilizing multiple virtual machines simultaneously, for speeding up the executions. Also, Gordon and *et al.* proposed the runtime system, COMET [10], supporting thread-level offloading, executing unmodified application over distributed shared memory, which can offer significant speed-ups on some real applications on Google Play.

Recently, as the growth of high computation demanding applications, such as AI applications with huge amount generated data, deploying edge nodes near to devices is becoming an applicable solution. The devices can access rich computation resource without suffering from heavy latency from devices to cloud data centers. In [11], Satyanarayanan and *et al.* proposed the concept of edge-clouds, providing the nearby rich computational power, and in [12] Tong and *et al.* proposed a hierarchical edge cloud architecture, with multiple intermediate edge tiers between devices and data centers.

Besides the architecture of edge computing, resource allocation is also a promising topic. In 2016, Wang and *et al.* [13] investigated the joint offloading scheduling over multiple edge nodes instead of how to offloading task to one centralized node. They considered both the data movement and processing and proposed algorithms for achieving the best cost efficiency and improving the admission rate. In [14], Nguyen and *et al.* proposed a market-based allocation framework over geographically distributed edge nodes. Their algorithms not only maximize the resource utilization but also satisfy the budgets of services. They formulate the investigated problem with two objectives as an extension of the Eisenberg-Gale (EG) convex program and prove their algorithms can achieve equilibrium. They consider the one-slot scenario rather than the time coupling cases. In [15], Wang and *et al.* proposed MOERA to solve the online resource allocation problem in a realistic edge computing scenario, considering different types of costs together (including migration, operation, service quality, and reconfiguration). They utilize the "regularization" to handle the online scenario, and prove the performance ratio. However, in their online scenario, they assume the resource capacity of each node at each slot is known in advance.

III. Preliminaries and Problem Formulation

We consider an edge computing provider, hosting considerable number of servers in base stations, which can provide different types of VM resources from the reusable edge resources, such as CPU, GPU, storage, and so on: the set of different types of VM resources denoted by $[M]$. We assume that the resources can be instantly reallocated when they stand idle. At each time, the edge computing provider might have different types of VM resources, which can be allocated to users (*i.e.*, companies) which are willing to deploy their applications and services in edge nodes for improving their user experience. The whole time span is divided into multiple time slots (*e.g.*, multiple hours per slot) to allocate, which is denoted by $[T]$. In each slot $t \in [T]$, let $C_m^{(t)}$ denote the remaining amount of type-m VM resource $m \in [M]$ at time t. Let $[N]$ denote the set of users who want the edge resources. For each user $n \in [N]$, it can submit a set of demand bundles for applying different types of VM resource at each time t. Without the loss of generality, each user submits at most K bundle at one slot, and let $[K]$ denote the index set of demand bundles. We formulate the bundle $k \in [K]$ of user n at time t as follows:

$$B_{n,k}^{(t)} = \{w_{n,k}^{(t)}, \{r_{n,k,m}^{(t)}, \forall m \in [M]\}\}$$

where $w_{n,k}^{(t)}$ denotes the valuation of $B_{n,k}^{(t)}$, and $r_{n,k,m}^{(t)}$ denotes the required number of type-m VM resource of $B_{n,k}^{(t)}$.

TABLE I
NOTATION TABLE

Var	Definition
M	# of different types of VMs
N	# of users
K	# of bundles
$B_{n,k}^{(t)}$	the k_{th} bid of user n at slot t
$w_{n,k}^{(t)}$	the valuation of the k_{th} bid of user n at slot t
$r_{n,k,m}^{(t)}$	the type-m VM demand of the k_{th} bid of user n at slot t
$C_m^{(t)}$	the (predicted) amount of unused type-m VM resource at slot t
$R_m^{(t)}$	the maximum type-m VM demand of one bid at slot t
$\Omega^{(t)}$	the maximum ratio between the amount of VMs and the maximum resource demand of one bid at slot t
VBAA	**V**M **B**undle **A**llocation **A**lgorithm
UNEARTH	**UN**usE**d** sp**A**red VM **R**e-u**T**ilizing mec**H**anism

In our model, $C_m^{(t)}$ cannot be precisely known before allocated, and we need to predict it; thus the allocated VM resource might exceed the actual amount of unused resource. In this case, we introduce backup VM resource to handle the over-allocation, which comes from elsewhere, like the public cloud, other edge computing provider nearby, and *etc*. The amount of backup type-m VM resource at time t is denoted by $S_m^{(t)}$, and the cost of backup type-m VM resource is denoted by b_m. Let $x_{n,k}^{(t)}$ denote the indicator of whether or not the submitted bundle k of user n is accepted at slot t. We formulate the problem of allocation available VM resource as follows:

$$\text{maximize} \quad \sum_{t\in[T]}\Big(\sum_{n\in[N]}\sum_{k\in[K]} w_{n,k}^{(t)}x_{n,k}^{(t)} - \sum_{m\in[M]} b_m\cdot S_m^{(t)}\Big) \quad (1)$$

subject to:

$$\sum_{k\in[K]} x_{n,k}^{(t)} \le 1, \forall t\in[T], \forall n\in[N] \quad (2)$$

$$\sum_{n\in[N]}\sum_{k\in[K]} r_{n,k,m}^{(t)}x_{n,k}^{(t)} \le C_m^{(t)} + S_m^{(t)}, \forall t\in[T], \forall m\in[M] \quad (3)$$

$$x_{n,k}^{(t)} \in \{0,1\}, \forall t\in[T], \forall n\in[N], \forall k\in[K] \quad (4)$$

$$S_m^{(t)} \ge 0, \forall t\in[T], \forall m\in[M] \quad (5)$$

IV. MECHANISM WITH UNCERTAIN CAPACITIES

We introduce our mechanism, UNEARTH, with uncertainty of VM resource, for maximizing social welfare, which decides $x_{n,k}^{(t)}$ and $S_m^{(t)}$ for all $n\in[N]$, for all $k\in[K]$, and for all $m\in[M]$, at $t\in[T]$.

A. Loss Minimization of Available VMs Prediction

Let $C_m^{(t)*}$ denote the actual number of unused type-m VM resource at slot t. If we know the resource capacity, the problem is a famous NP-hard winner determination problem (WDP) with multiple users with multiple bundles. However, we do not know $C_m^{(t)*}$ before allocating resources to submitted bundles at slot t—*i.e.*, $\{B_{n,k}^{(t)}, \forall n\in[N], \forall k\in[K]\}$, and we need to predict the resource first. During the prediction, there are three possible cases about the predicted capacity $C_m^{(t)}$ at slot t:

(1): $C_m^{(t)} = C_m^{(t)*}$—the perfect prediction. In this case, we predict the exact resource capacity, and thus the amount of allocated VM resource will not exceed the actual amount of unused type-m VM resource of edge nodes m. Therefore, no backup VM resource is needed.

(2): $C_m^{(t)} < C_m^{(t)*}$—the under-prediction. In this case, we predict a smaller resource capacity, and thus users are allocated at most $C_m^{(t)}$ type-m VM resource in total, less than the actual amount (*i.e.*, $C_m^{(t)*}$). Compared with case (1), smaller welfare is incurred due to the under-prediction of capacities.

(3): $C_m^{(t)} > C_m^{(t)*}$—the over-prediction. In this case, we predict a larger resource capacity, and thus allocated resource to users might be more than the available type-m VM resource. Therefore, we need the backup resource. As the higher cost of backup type-m VMs resource compared with case (1), smaller welfare is incurred due to the backup resource cost.

Prediction Loss Minimization. We aim at minimizing the prediction loss $|C_m^{(t)*} - C_m^{(t)}|$, enabling to bound the performance gap of the long-term prediction errors. We formulate the loss minimization problem (for predicting the amount of unused VM resource) as follows:

$$\text{minimize}_{C_m^{(t)},\forall t\in[T]} \quad \sum_{t\in[T]}\Big|C_m^{(t)*} - C_m^{(t)}\Big| \quad (6)$$

Our algorithm solves the problem by utilizing online gradient descent (*i.e.*, OGD). At slot t, our algorithm decides the prediction $C_m^{(t)}$ by the accumulated prediction error of the past $t-1$ slots. In the long-term perspective, our algorithm can ensure the overall regret of prediction (*i.e.*, the difference between the prediction and the benchmark), is sub-linear to the optimal static predictor (*i.e.*, the benchmark). In the online learning field, OGD is still a widely-used and effective method for making a prediction (being ensure the sub-linear regret).

Randomized Rounding. In our scenario, $C_m^{(t)}$ represents the amount of unused type-m VM resource, packed as a VM instance, which is indivisible. The original OGD for convex assumption cannot solve our problem appropriately. Therefore, we need to bridge the fractional number in OGD and the integer number in our setting and ensure that they have the same value (in expectation). We introduce a fractional number $\theta_m^{(t)}$ (using in OGD) and the corresponding distribution $\mathcal{D}_{t,m}(\theta_m^{(t)})$ for achieving the conversion from the fractional number to an integer, defined as follows:

$$\mathcal{D}_{t,m}(\theta_m^{(t)}) = \begin{cases} Pr[C_m^{(t)} = \lfloor\theta_m^{(t)}\rfloor + 1] = \theta_m^{(t)} - \lfloor\theta_m^{(t)}\rfloor \\ Pr[C_m^{(t)} = \lfloor\theta_m^{(t)}\rfloor] = 1 - \theta_m^{(t)} + \lfloor\theta_m^{(t)}\rfloor. \end{cases} \quad (7)$$

Following the distribution show in (7), $|C_m^{(t)*} - \theta_m^{(t)}|$ is equal to the loss $|C_m^{(t)*} - C_m^{(t)}|$ in expectation. The detailed proof is shown in technical report [16].

B. Allocation with Unused VM Resource Prediction

As shown in **Alg.** 1, our algorithm already predicts the number of unused VM resource before making allocation decisions. And then, based on the resource prediction, we proposed our allocation algorithm for WDP to decide which submitted bundles are accepted.

Algorithm 1: Online Unused VM Resource Re-Utilizing Mechanism

Input : $M, C_{\max}, \mathbf{B}$
Output : $\mathbf{x}, \mathbf{S}$
Initialize: $i = 0;\ \theta(0)_m = 0;$
1 **for** $t = 1, 2, \ldots, T$ **do**
  // Run OGD to predict the number of unused VMs
2 **if** $t \geq 2^i$ **then**
3 $i = i + 1;$
4 $\sigma = 2^i;$
5 **end**
6 **for** $m = 1, \cdots, M$ **do**
7 $\eta = \frac{C_{\max}}{\sqrt{2\sigma}};$
8 $\theta_m^{(t)} = \theta_m^{(t-1)} - \eta \nabla g(\theta_m^{(t-1)});$
9 $\mathcal{D}_{t,m}(\theta_m^{(t)}) =$ $\begin{cases} \mathbf{Pr}[C_m^{(t)} = \lfloor \theta_m^{(t)} \rfloor] = 1 - \theta_m^{(t)} + \lfloor \theta_m^{(t)} \rfloor \\ \mathbf{Pr}[C_m^{(t)} = \lfloor \theta_m^{(t)} \rfloor + 1] = \theta_m^{(t)} - \lfloor \theta_m^{(t)} \rfloor \end{cases}$
10 $C_m^{(t)} \sim \mathcal{D}_{t,m}(\theta_m^{(t)});$
11 **end**
  // Decide the bundle allocation
12 $\mathbf{x}(t) = \mathsf{VBAA}(\mathbf{B}(t), \mathbf{C}(t));$
  // Decide the usage of backup VMs
13 **for** $m = 1, \cdots, M$ **do**
14 $S_m^{(t)} = [\sum_{n \in [N]} \sum_{k \in [K]} r_{n,k,m}^{(t)} x_{n,k}^{(t)} - C_m^{(t)*}]_+;$
15 **end**
16 **end**

Algorithm procedure. At the beginning of slot t, we first predict the number of unused type-m VM resource by online gradient descent (OGD) (line 2-11). According to the predicted number (*i.e.*, $C_m^{(t)}$), we decide which bundles are accepted to allocated (*i.e.*, $x_{n,k}^{(t)}$). The gradient of (6)—$\nabla g(C_m^{(t)})$ is defined as follows:

$$\nabla g(C_m^{(t)}) = \begin{cases} 1 & \text{if } C_m^{(t)} > C_m^{(t)*} \\ 0 & \text{if } C_m^{(t)} = C_m^{(t)*} \\ -1 & \text{if } C_m^{(t)} < C_m^{(t)*} \end{cases} \tag{8}$$

In line 12, we decide the bundle allocation $\mathbf{x}(t)$; and in line 13-15, we compute the exact amount of backup resource $S_m^{(t)}$ which we used at the end of slot t.

C. Algorithm for VM Bundle Allocation

We next describe the unused **VM** **B**undle **A**llocation **A**lgorithm, VBAA, in detail, which is shown as a subfunction of **Alg.** 1 (line 12). In our scenario, we allocate the integer number of VM resources with satisfying multiple capacity constraints, and thus solving it with performance guarantee becomes a challenging problem. In VBAA, we are inspired by the primal-dual framework [17] and ensure the performance guarantee. We present the primal problem and the dual problem for each slot. The primal problem is shown as follows:

$$\text{maximize} \quad \sum_{n \in [N]} \sum_{k \in [K]} w_{n,k}^{(t)} x_{n,k}^{(t)} \tag{9}$$

subject to:

$$\sum_{k \in [K]} x_{n,k}^{(t)} \leq 1, \forall n \in [N] \tag{9a}$$

$$\sum_{n \in [N]} \sum_{k \in [K]} r_{n,k,m}^{(t)} x_{n,k}^{(t)} \leq C_m^{(t)}, \forall m \in [M] \tag{9b}$$

$$x_{n,k}^{(t)} \in \{0, 1\}, \forall n \in [N], \forall k \in [K] \tag{9c}$$

The corresponding dual problem is as follows:

$$\text{minimize} \quad \sum_{n \in [N]} \lambda_n^{(t)} + \sum_{m \in [M]} C_m^{(t)} \mu_m^{(t)} \tag{10}$$

subject to:

$$\lambda_n^{(t)} + \sum_{m \in [M]} r_{n,k,m}^{(t)} \mu_m^{(t)} \geq w_{n,k}^{(t)}, \forall n \in [N], \forall k \in [K] \tag{10a}$$

$$\lambda_n^{(t)}, \mu_m^{(t)} \geq 0, \forall n \in [N], \forall m \in [M] \tag{10b}$$

where $\lambda_n^{(t)}$ is the dual variable according to (9a), and $\mu_m^{(t)}$ is the dual variable according to (9b). As the set of constraints (9c) is a subset of (9a), we need not introduce a dual variable for it.

As shown in **Alg.** 2, in each slot, we decide accepted bundles according to the predicted number of unused spared type-m VMs in slot t—*i.e.*, $C_m^{(t)}$ where $R_m^{(t)} = \max_{n \in [N], k \in [K]} r_{n,k,m}^{(t)}$ and $\Omega^{(t)} = \min_{m \in [M]} \{\frac{C_m^{(t)}}{R_m^{(t)}}\}$, representing the minimum ratio among the ratios of a specific type of VM capacity to the maximum VM demand of this type VM in one bundle.

Theorem 1. *In time slot t, **Alg.** 2 computes a feasible solution for* (9) *and achieve α-approximation ratio where $\alpha = 1 + \frac{\Omega^{(t)}}{\Omega^{(t)}-1} \epsilon^{(t)} (eM^{\frac{1}{\Omega^{(t)}-1}} - 1)$ and $\epsilon^{(t)} = \max_{k_1 \in [K], k_2 \in [K]} \{\frac{r_{n,k_1,m}^{(t)}}{r_{n,k_2,m}^{(t)}}\}$.*

The detail proof is shown in technical report [16].

Theorem 2. *In each time slot, the time complexity of **Alg.** 2 is $O(NK + N(M + NM))$, where N is the number of users, M is the number of different types of VMs, and K is the maximum number of bundle of one user.*

The detail proof is shown in technical report [16].

V. Performance Evaluation

We now evaluate our mechanism shown in **Alg.** 1 and **Alg.** 2, compared with existing benchmarks and the optimum.

Algorithm 2: VM Bundle Allocation Algorithm with Predicted Capacity (*i.e.*, VBAA)

Input : $\mathbf{B}^{(t)}, \mathbf{C}^{(t)}$
Output : $\mathbf{x}^{(t)}$
Initialize: $\mathcal{W} = \varnothing,\ \mu_{\text{base}} = Me^{(\Omega^{(t)}-1)}$
$x_{n,k}^{(t)} = 0,\ \lambda_n^{(t)} = 0,\ \mu_m^{(t)} = \frac{1}{C_m^{(t)}}, \forall n, \forall m$

while $\sum_{m\in[M]} C_m^{(t)}\mu_m^{(t)} < \mu_{\text{base}}$ and $|\mathcal{W}| < N$ **do**
  forall $n \notin \mathcal{W}$ **do**
    $k(n) = \arg\max_{k\in[K]} w_{n,k}^{(t)}$;
  end
  $n_{\max} = \arg\max_{n\in[N]:n\notin\mathcal{W}} \frac{w_{n,k(n)}^{(t)}}{\sum_{m\in[M]} r_{n,k(n),m}^{(t)}\mu_m^{(t)}}$;
  $x_{n_{\max},k(n_{\max})}^{(t)} = 1$;
  $\mathcal{W} = \mathcal{W} \cup \{n_{\max}\}$;
  $\lambda_{n_{\max}}^{(t)} = w_{n_{\max},k(n_{\max})}^{(t)}$;
  forall $m \in [M]$ **do**
    $\mu_m^{(t)} = \mu_m^{(t)} \cdot \mu_{\text{base}}^{\ \frac{r_{n_{\max},k(n_{\max}),m}^{(t)}}{C_m^{(t)} - R_m^{(t)}}}$
  end
end

A. Simulation Setup

Unused VM Resource Capacities. By default, we consider an edge computing provider offers 30 different types of VMs, and the maximum amount of each type VM is 1000, resource capacity similar to an edge provider with 30 Vapor Chamber [18]. In default, the initial ratios of the unused VM resource amount to the VM capacity of each type of VM are uniformly randomly chosen from [0,1], which continuously changes through time. We use the normal distribution N(0,0.3) to simulate the VM utilization changes between two adjacent slots in default.

Bundle Configurations. For each bidding bundle, the required VM types are randomly selected at most ten types of VMs from these 30 types, and the required number of VMs is randomly distributed over $[20, 100]$. In default, the (normalized) value density of each bundle is distributed over $[0, 1]$ and the value density of backup VM is 1.5 times of the maximum value density of bundles.

Comparisons. We first evaluate the performance of the entire performance (including prediction and allocation together), and then we evaluate these two parts separately, for showing how each parts contribute the overall performance. We compare the overall performance of the whole on-demand resource allocation system with heuristic algorithms using widely used predictors. Besides, we compare our predictor with the static optimum (SOP) and the exponentially weighted moving average (EWMA) [7], which is widely used in machine learning optimizers (*e.g.*, Adam and Momentum), showing the prediction loss. Moreover, we compare our algorithm, **Alg.** 2, with two heuristic algorithms (ordered by per-unit value—Greedy-1—and ordered by bid value—Greedy-2) and the optimum (OPT).

B. Evaluation Results

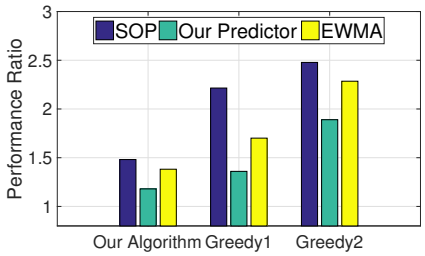

Fig. 2. Performance ratio: different VM allocation algorithms and different capacity predictors.

Overall Performance.

Impact of different predictors and different allocation algorithms: In **Fig.** 2, we show the performance ratio, which is the ratio of optimum to the objective value of (1), achieved over different predictors and different VM allocation algorithms. We can see that our algorithm (*i.e.*, **Alg.** 1) outperform the benchmarks (i.e., closer to 1 the optimum). The difference of performance comes from two aspects: **1.** the resource predictor: In each group of bars, we can see that a more accurate predictor (*i.e.*, less prediction loss) has the better performance because a larger prediction loss causes the more usage of the backup resource (more expensive). **2.** the bundle allocation algorithm: Under the same predictor, the results show that our bundle allocation algorithm has better performance than benchmarks, which means that our algorithm can allocate resource more efficiently.

Prediction Loss.

In **Fig.** 3, we compare the prediction loss (defined in (6)) achieved over different predictors, such as the static optimum (SOP), our predictor, and the exponentially weighted moving average (EWMA). We can see that our predictor has less prediction error than baseline predictors. Even the SOP (*i.e.*, optimum static predictor) choose the best in static, as its prediction cannot change when the amount of resource fluctuates, its prediction loss is worse than our predictor and EWMA. Moreover, the EWMA (*i.e.*, exponentially weighted moving average predictor) only averages the few of past slots with an exponential decay, but our predictor can adjust further our prediction more sensitively according to the gradient.

Performance Ratio.

Impact of different amount of demands: In **Fig.** 4, we evaluate the impact of demand amount, by multiplying the demand amount of bundles by a factor (from 0.5 to 2 times than the default setting). When each bundle requires more resource, a worse allocation influences more about the overall performance (*i.e.*, the more careful allocation is needed when allocating a larger bundle). And our algorithm considers the amount of allocated resources when making decisions; and the results also show that our allocation algorithm still ensures better allocations than benchmarks when the demand is increasing.

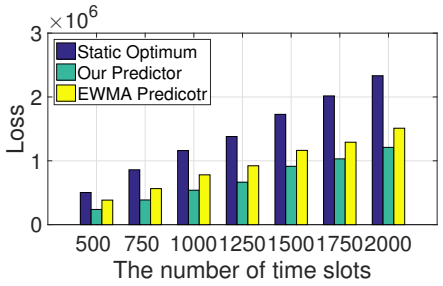

Fig. 3. Accumulated loss: static optimum, our predictor, and EWMA.

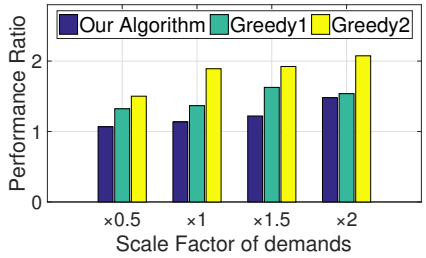

Fig. 4. Performance ratio: different amount of demands.

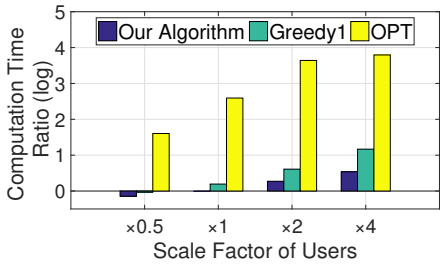

Fig. 5. Computation time ratio: different VM allocation algorithms.

Computation Time.

In **Fig.** 5, for showing the computation performance when the problem scale is increasing, we compare our algorithm (*i.e.*, **Alg.** 2) with the greedy algorithm and the optimum by using MILP solver (*i.e.*, MATLAB optimizer intlinprog). The ratio is defined as the computation time divides by the time of our algorithm under default setting (the results shown in **Fig.** 5 are under $\log_{10}(\cdot)$ for ease to read). In the default setting, the greedy algorithm and the optimizer for optimum are respectively 1.5 and 391 times slower than our algorithm. When the number of users is four times more than the default setting, the computation time of our algorithm is 3.43 times than before; and the greedy algorithm and the optimizer for optimum is respectively 14 and 6226 times slower than our algorithm. As the MILP aims to find the optimal solution of mixed integer programming, which is an NP-hard problem, it spends lots of time. Moreover, the heuristic algorithms find the best choice for each bundle of each user, it needs to check the capacity constraints one by one, but our algorithm utilizes the primal-dual framework and checks the validation of dual constraints instead of capacity constraints. The results show that our algorithm is much faster than the optimum solver, and the computation time almost increases linearly over the increase in the number of users.

VI. Concluding Remarks

This paper proposes UNEARTH, a novel unused VM resource re-utilizing mechanism that involves the idea to improve the resource utilization of edge computing providers. We provide an online unused VM resource prediction algorithm inspired by online gradient descent, and we also provide an approximation algorithm for VM bundles allocation under the primal-dual technique. In the theoretical analysis, we prove the prediction error of our prediction algorithm and the approximation ratio of our VM bundle allocation algorithms. In the evaluation, we validate our analysis and show that our algorithm components outperform the heuristic and existing benchmarks.

Acknowledgments

This work was supported in part by grants from Hong Kong RGC under the contracts HKU 17204715, 17225516, C7036-15G (CRF) and the U.S. NSF under grants CNS-1551661 and ECCS-1610471.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, pp. 1628–1656, 2017.
[2] "Vapor IO," https://www.vapor.io/cdn-cloud-providers/.
[3] "AWS Greengrass - Amazon Web Services," https://aws.amazon.com/greengrass/.
[4] "Kinetic Edge – Vapor IO," https://www.vapor.io/kinetic-edge/.
[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
[6] "DataBank Plans Wireless Tower Data Center Services for Edge Computing," http://www.datacenterknowledge.com/business/databank-plans-wireless-tower-data-center-services-edge-computing.
[7] "Exponential moving average," https://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average.
[8] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.
[9] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 945–953.
[10] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET: Code offload by migrating execution transparently," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. Hollywood, CA: USENIX, 2012, pp. 93–106.
[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.
[12] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.
[13] L. Wang, L. Jiao, D. Kliazovich, and P. Bouvry, "Reconciling task assignment and scheduling in mobile edge clouds," in *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–6.
[14] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, 2018.
[15] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser, "Moera: Mobility-agnostic online resource allocation for edge computing," *IEEE Transactions on Mobile Computing*, 2018.
[16] "Technical Report," https://www.dropbox.com/s/uyiqjx2q9382qr5/icc_technical_report.pdf?dl=0.
[17] N. Buchbinder, J. S. Naor *et al.*, "The design of competitive online algorithms via a primal–dual approach," *Foundations and Trends® in Theoretical Computer Science*, vol. 3, no. 2–3, pp. 93–263, 2009.
[18] "Vapor Chamber Specs," https://www.vapor.io/chamber/.