A Hierarchical Edge Cloud Architecture for Mobile Computing

Liang Tong, Yong Li and Wei Gao
Department of Electrical Engineering and Computer Science
University of Tennessee at Knoxville
{ltong, yli118}@vols.utk.edu, weigao@utk.edu

***Abstract*—The performance of mobile computing would be significantly improved by leveraging cloud computing and migrating mobile workloads for remote execution at the cloud. In this paper, to efficiently handle the peak load and satisfy the requirements of remote program execution, we propose to deploy cloud servers at the network edge and design the edge cloud as a tree hierarchy of geo-distributed servers, so as to efficiently utilize the cloud resources to serve the peak loads from mobile users. The hierarchical architecture of edge cloud enables aggregation of the peak loads across different tiers of cloud servers to maximize the amount of mobile workloads being served. To ensure efficient utilization of cloud resources, we further propose a workload placement algorithm that decides which edge cloud servers mobile programs are placed on and how much computational capacity is provisioned to execute each program. The performance of our proposed hierarchical edge cloud architecture on serving mobile workloads is evaluated by formal analysis, small-scale system experimentation, and large-scale trace-based simulations.**

I. INTRODUCTION

One of the most important challenges in mobile computing is to address the contradiction between the increasing complexity of mobile applications and the limited local capabilities of mobile devices. A viable solution to this challenge is to leverage cloud computing and execute mobile applications remotely. Such remote execution benefits a large varieties of mobile applications, such as gesture recognition [8], voice control, recognition assistance [2] and mobile gaming [6].

Modern cloud computing services, such as Amazon EC2 and Microsoft Azure, are solely hosted by data centers and incapable of efficiently executing mobile applications due to the following reasons. First, many mobile applications are delay-sensitive and require immediate response [2]. Offloading mobile workloads to remote data centers or computing clusters, however, incurs long network transmission latency [7], which seriously impairs the mobile application performance. Second, current data centers mainly focus on serving enterprise users with high volumes of workloads [11], by providing virtualized cloud resources as Virtual Machines (VMs). These data centers handle each mobile application using a separate VM no matter how small its amount of workload is, but incur significant overhead for global VM provisioning and management due to the huge number of mobile applications using the cloud. Such overhead may even exceed the expense of mobile program execution itself and overload the data centers during peak hours.

This work was supported in part by the National Science Foundation (NSF) under grants CNS-1526769 and CNS-1527612, and was supported by the Army Research Office (ARO) under grant W911NF-15-1-0221.

Therefore, there are pressing needs to redesign the cloud architecture to serve the mobile workloads with better efficiency and scalability. Recent research efforts propose to deploy an intermediate cloud layer, so-called cloudlets, at the network edge [7], [9]. Since the distance between mobile devices and the cloud is shortened, the network latency accessing cloud computing services could be significantly reduced. However, these existing designs inappropriately consider the edge cloud as a black box and assumes unlimited cloud computing resources at all times. Mobile users, hence, may experience excessive cloud access latency when they are served by local cloudlets with limited computing capabilities and large amounts of peak load. Provisioning more capacities on cloudlets would help reduce such cloud access latency, but seriously impairs the efficiency of cloud resource utilization during off-peak hours [12].

In this paper, we address the aforementioned challenge and improve the efficiency of cloud resource utilization by organizing the edge cloud servers into a hierarchical architecture. Instead of serving mobile users directly using a flat collection of edge cloud servers, our basic idea is to opportunistically aggregate and serve the peak loads that exceed the capacities of lower tiers of edge cloud servers to other servers at higher tiers in the edge cloud hierarchy. As a result, we are able to serve larger amounts of peak loads with the same amount of computational capacities being provisioned at the edge. We developed analytical models to compare the performance and resource utilization efficiency between flat and hierarchical designs of the edge cloud, and provided theoretical results showing the advantage of hierarchical edge cloud architecture.

Furthermore, we ensure efficient utilization of the computing resources at different tiers of edge cloud servers to better serve the peak load, by developing optimization algorithms that adaptively place the mobile workloads at different edge cloud servers. The proposed workload placement algorithms focus on deciding which the edge cloud servers mobile programs are placed on and how much computational capacity is provisioned to execute each program. The major challenge of such algorithm design, however, lies in the difficulty of appropriately balancing between the computation and communication delays in the edge cloud. Placing mobile programs at high-tier servers reduces contention on computing resources and improves the program execution performance, but also spends longer time on transmitting the program states over the network. To address this challenge and find the optimal solution of workload placement, we first consider a scenario which has only one server at each tier of the edge cloud. Afterwards, we aggregate the decisions of workload placement

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

at different edge cloud servers together.

To the best of our knowledge, our work is the first to design the edge cloud and further improve its performance in such a hierarchical manner. Our detailed contributions are as follows:

- We design a hierarchical edge cloud architecture which enables aggregation of the peak loads across different tiers of cloud servers. We propose an analytical model to compare the efficiency of cloud resource utilization between flat and hierarchical designs of edge cloud, and show that the hierarchical edge cloud has better efficiency serving the peak loads.
- We develop workload placement algorithms that adaptively place mobile workloads among different tiers of servers and decide how much computational capacity is provisioned to execute each program, so as to minimize the average program execution delay in the edge cloud.
- We conduct performance evaluation over both small-scale computing clusters and large-scale mobile workload traces. The results demonstrate that the proposed hierarchical edge cloud architecture outperforms the flat edge cloud architecture by more than 25% in terms of the average delay of program execution.

The rest of this paper is organized as follows. Section II reviews the related work. Section III provides a brief overview of the proposed hierarchical edge cloud architecture. Section IV describes the details of our analytical models. Section V presents the proposed workload placement problem in hierarchical edge cloud. Sections VI and VII evaluate our proposed design. Section VIII concludes the paper.

II. Related work

Although extensive research studies have been done in the area of mobile cloud computing, they are limited to reduce the network latency accessing cloud services, by supporting mobile code offload via live VM synthesis and migration. On one hand, efficient VM allocation algorithms have been developed for geo-distributed clouds with real-time request arrivals [4], and also support parallel method executions with multiple VM images [5]. On the other hand, transparent VM migration allows processing threads to freely migrate between different machines [1], and also ensures the delay requirement of VM migration to be satisfied. However, all these schemes inappropriately assume that the computing resources are unlimited at cloud servers.

Existing research also suggests to deploy small-scale cloud servers at the edge to reduce the overhead of VM provisioning and resource management. Ha et al. proposes rapid and precise provisioning of edge clouds under the control of associated mobile devices [3], so as to support delay-sensitive applications. These edge cloud servers are also considered as cloudlets to efficiently utilize the available resources in a wireless LAN [9]. However, these edge cloud servers are only organized in a flat architecture, which is incapable of handling the peak load exceeding the capacity of individual servers and significantly increases the cloud processing delay. Comparatively, in this paper we propose a hierarchical edge cloud architecture which is able to handle such peak loads more efficiently and promptly.

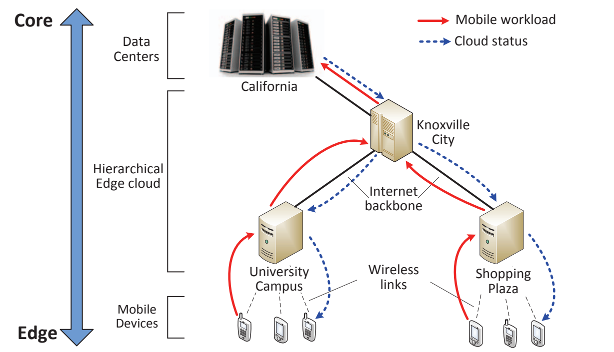


Fig. 1: The hierarchical edge cloud architecture

The most related work to our proposed workload placement algorithm is load balancing in cloud computing systems. Current load balancing solutions suggest to distribute incoming workloads among multiple servers, so as to prevent any server from becoming a single point of failure or being overloaded. Maguluri et al. [16] proposes a stochastic model for load balancing and scheduling in cloud computing clusters, and they also develop frame-based non-preemptive VM configuration policies which is proved to be nearly throughput-optimal. Chen et al. [17] proposes a resource intensity aware load balancing (RIAL) method in clouds that migrates VMs from overloaded physical machines to lightly loaded physical machines. Zhao et al. [19] proposes a heuristic clustering-based task deployment approach for load balancing using the Bayes Theorem. However, these load balancing schemes are mainly designed for data centers and only aim to optimize the computation delay of workload execution. In contrast, our proposed architecture of the hierarchical edge cloud distributes mobile workloads among different tiers of edge cloud servers, hence imposing special challenges on the tradeoff between the computation and communication delays that makes these traditional schemes hard to be used.

III. Overview of the Hierarchical Edge Cloud

The proposed architecture of hierarchical edge cloud is shown in Figure 1. The servers at the first tier of the hierarchy, which could usually be geo-distributed desktops or workstations, directly receive workloads from mobile devices via wireless links. These tier-1 servers, then, are connected to higher tiers of edge cloud servers and remote data centers through the Internet backbone.

As a result, servers on different tiers can share run-time information of mobile workloads with each other, so as to work in a cooperative manner. If the mobile workloads received by an edge cloud server exceed its computational capacity, the excessive amount of workloads will then be further offloaded to the server at the higher tier. In this way, we can serve larger amounts of peak loads by provisioning the same amount of computational capacity in the edge cloud. For example, if the provisioned capacity C is evenly distributed to two tier-1 servers, each server is only able to serve the peak load of $\frac{C}{2}$. In contrast, if we reduce the provisioned capacity of each tier-1 server to $\frac{C}{4}$ and use the remaining $\frac{C}{2}$ to provision another tier-2 server, each tier-1 server will be able to serve the peak load up to $\frac{3}{4}C$, as long as the peak loads at two tier-1 servers do not

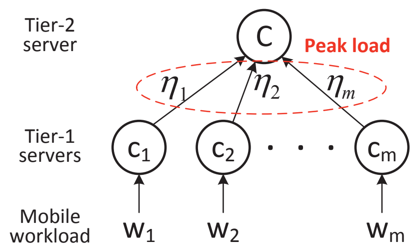


Fig. 2: Notations in a two-tier hierarchical edge cloud

appear at the same time. In Section IV, we present analytical studies that formally prove the advantage of the hierarchical edge cloud in serving the peak load.

Another major issue of designing the hierarchical edge cloud is how to place mobile workloads among different tiers of servers, and how much computational capacity should be allocated to execute each workload. The major challenge of such workload placement, which aims to minimize the average execution delay of all mobile workloads, is the difficulty of appropriately balancing between the computation and communication delays in the edge cloud. As shown in Figure 1, placing mobile programs at high-tier servers reduces contention on computing resources and improves the program execution performance, but also spends longer time on transmitting the program states to these servers via the network. In Section V, we developed optimization-based workload placement algorithms to address this challenge.

IV. Formal Analysis

In this section, we formally prove the advantage of an hierarchical edge cloud on improving the efficiency of cloud resource utilization when serving the peak load. Without loss of generality, we first develop an analytical model for a two-tier hierarchical edge cloud. Analysis results based on this two-tier model, then, can be easily generalized to more complicated topology of the hierarchical edge cloud.

A two-tier hierarchical edge cloud, as shown in Figure 2, consists of m tier-1 servers and one tier-2 server. c_i and w_i indicate the computational capacity and the amount of received mobile workload of the i-th tier-1 server, respectively. Both c_i and w_i are measured in units of CPU cycles. When $w_i > c_i$, a workload of $\eta_i = w_i - c_i$ will be offloaded to the tier-2 server, which has a capacity of C. In other words, the tier-2 server will aggregate the peak loads offloaded from all tier-1 servers.

A. Tier-1 Workload Model

Without loss of generality, we assume that $w_1, w_2, ..., w_m$ are independent and identically distributed (i.i.d.) random variables, and $\mathbb{P}(w_i \le x)$ shows the probability that the i-th tier-1 server can successfully serve its workload w_i when its computational capacity is x. Then, the probability that all tier-1 servers can serve their received mobile workloads is:

$$\mathbb{P}(w_1 \le c_1, w_2 \le c_2, .., w_m \le c_m) = \prod_{i=1}^{m} \mathbb{P}(w_i \le c_i).$$

When $w_i > c_i$, the workload of amount $\eta_i = w_i - c_i$ will be offloaded to the tier-2 server. Otherwise, $\eta_i = 0$ when $w_i \le c_i$. We hence have the cumulative distribution function (CDF) of η_i as

$$F_{\eta_i}(x) = \mathbb{P}(\eta_i \le x) = \begin{cases} \mathbb{P}(w_i \le x + c_i) & \text{if } x \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

B. Tier-2 Workload Model

As shown in Figure 2, the workload on the tier-2 server is $W = \sum_{i=1}^{m} \eta_i$. Then, we have the following theorem:

***Theorem** 1:* Let $F_m(x) = \mathbb{P}(W \le x)$ and $G_m(x) = \mathbb{P}(\eta_m \le x)$, we have

$$F_m(x) = F_{m-1}(x) * G_m(0) + \int_{0^+}^{x} F_{m-1}(x-t) * dG_m(t), \quad (1)$$

where

$$F_1(x) = G_1(x). \quad (2)$$

Proof: When $m = 1$, there's only one tier-1 server, so $F_1(x) = G_1(x)$. When $m > 1$, we divide the m tier-1 servers into two groups G_1 and G_2: where G_1 has server 1,2,...,$m-1$, G_2 has server m.

If there is no workload offloaded by G_2, this means all the workloads come from G_1, the probability that the tier-2 server can serve its workload (W) is

$$\mathbb{P}_1(x) = \mathbb{P}(\sum_{i=1}^{m-1} \eta_i \le x) \cdot \mathbb{P}(\eta_m = 0) = F_{m-1}(x) * G_m(0).$$

If there is a workload of amount $t (t > 0)$ offloaded by G_2, then workloads offloaded by G_1 can not exceed $x - t$ because x is computational capacity of the tier-2 server. When t is fixed, the probability that the tier-2 server can serve its workload (W) is hence

$$\mathbb{P}(\sum_{i=1}^{m-1} \eta_i \le x - t) \cdot \mathbb{P}(\eta_m = t).$$

Since $0 < t \le x$, this probability is

$$\mathbb{P}_2(x) = \int_{0^+}^{x} \mathbb{P}(\sum_{i=1}^{m-1} \eta_i \le x - t) d\mathbb{P}(\eta_m \le t),$$

and thus $F_m(x) = \mathbb{P}_1(x) + \mathbb{P}_2(x)$. ■

Theorem 1 shows how to analytically derive the characteristics of W, given the knowledge about workloads received by all tier-1 servers.

C. Provisioning of Edge Cloud Capacity

Built on the above workload models, we further analyze the advantage of hierarchical edge cloud on improving the efficiency of cloud resource utilization. As shown in Figure 2, the amount of capacity C is provisioned to the tier-2 server. Correspondingly when we provision the flat edge cloud using the same amount of capacity, C will be provisioned to tier-1 servers.

To analyze the difference between flat and hierarchical designs of edge cloud, we first have the following lemma:

***Lemma** 1:* For any $c_1^*, c_2^* \in [0, C], c_1^* + c_2^* = C$, we have

$$\mathbb{P}(\eta_1 + \eta_2 \le C) \ge \mathbb{P}(\eta_1 \le c_1^*) \cdot \mathbb{P}(\eta_2 \le c_2^*). \quad (3)$$

Proof: We define two sets Υ_1 and Υ_2 as

$$\begin{cases} \Upsilon_1 &= \{(\eta_1, \eta_2) | \eta_1 + \eta_2 \le C\} \\ \Upsilon_2 &= \{(\eta_1, \eta_2) | \eta_1 \le c_1^*, \eta_2 \le c_2^*\}. \end{cases}$$

It is obvious for any element $\{t_1, t_2\} \in \Upsilon_2$, we also have $\{t_1, t_2\} \in \Upsilon_1$, which means $\Upsilon_2 \subseteq \Upsilon_1$. Therefore we have

$$\mathbb{P}(\eta_1 + \eta_2 \le C) \ge \mathbb{P}(\eta_1 \le c_1^*) \cdot \mathbb{P}(\eta_2 \le c_2^*).$$

■

Lemma 1 can be further extended to Lemma 2:

***Lemma** 2:* For any $c_i^* \in [0, C](i = 1, 2, ..., m)$ and $\sum_{i=1}^m c_i^* = C$, we have

$$\mathbb{P}(\sum_{i=1}^m \eta_i \leq C) \geq \prod_{i=1}^m \mathbb{P}(\eta_i \leq c_i^*). \tag{4}$$

Proof: Lemma 2 can be proved by recursively using Lemma 1:

$$\begin{aligned}\mathbb{P}(\sum_{i=1}^m \eta_i \leq C) &\geq \mathbb{P}(\sum_{i=1}^{m-1} \eta_i \leq \sum_{i=1}^{m-1} c_i^*) \cdot \mathbb{P}(\eta_m \leq c_m^*) \\ &\geq \mathbb{P}(\sum_{i=1}^{m-2} \eta_i \leq \sum_{i=1}^{m-2} c_i^*) \\ &\cdot \mathbb{P}(\eta_{m-1} \leq c_{m-1}^*) \cdot \mathbb{P}(\eta_m \leq c_m^*) \\ &\geq ... \geq \prod_{i=1}^m \mathbb{P}(\eta_i \leq c_i^*).\end{aligned}$$

■

Lemma 2 immediately leads to the following theorem:

***Theorem** 2:* For any $\alpha_i \in [0, 1](i = 1, 2, ...m)$ and $\sum_i^m \alpha_i = 1$, we have

$$\mathbb{P}(\sum_{i=1}^m \eta_i \leq C) \geq \prod_{i=1}^m \mathbb{P}(w_i \leq c_i + \alpha_i C) \tag{5}$$

Proof: Since $\eta_i = w_i - c_i$, Eq. (5) is a special case of Eq. (4) by substituting c_i^* in Eq. (4) with $\alpha_i C$. ■

Theorem 2 shows that, with a fixed amount of computational capacity being provisioned, the hierarchical edge cloud always has a higher chance to successfully serve the peak loads. From Lemma 2 and Theorem 2, we further have the following corollary:

***Corollary** 1:* If we partition $\{\eta_i\}$, $(i = 1, 2, ..., m)$ into $n(n \leq m)$ mutually exclusive groups $\{S_1, S_2, ..., S_n\}$, for any $c_j^* \in [0, C]$ $(j = 1, 2, ..., n)$ and $\sum_{j=1}^n c_j^* = C$, we have

$$\mathbb{P}(\sum_{i=1}^m \eta_i \leq C) \geq \prod_{j=1}^n \mathbb{P}(\sum_{k \in S_j} \eta_k \leq c_j^*) \tag{6}$$

Corollary 1 indicates that, the efficiency of resource utilization in the two-tier edge cloud will be maximized when there is only one tier-2 server. When $n = m$ in Eq. (6), the number of tier-1 and tier-2 servers are the same, the flat and hierarchical designs of edge cloud will be equivalent to each other. Obviously, the above conclusions we obtained from Theorem 2 and Corollary 1 will also hold for hierarchical edge clouds with more than two tiers. For example, to construct a three-tier edge cloud, we could simply use $\sum_{i=1}^m \eta_i$ as the workload of a tier-2 server and repeat the above analysis recursively.

V. Optimal Workload Placement

The analytical model in Section IV demonstrates the advantage of hierarchical edge cloud, but cannot ensure efficient execution of mobile programs in the cloud, because it greedily consumes the resources of low-tier servers without appropriately placing mobile workloads to other tiers of the edge cloud. In this section, we focus on deciding which edge cloud servers mobile programs are placed on and how much computation capacity is provisioned to execute each program, so as to optimize the performance of executing all mobile programs. Our proposed algorithm builds on existing schemes [5], [18] partitioning a mobile application. We consider each partition as a separate computing task and hence enable execution of a mobile application on multiple edge cloud servers. We first consider a scenario which has only one server at each tier of the edge cloud. Afterwards, we aggregate the decisions of workload placement at different edge cloud servers together.

When there is only one server at each tier of the edge cloud, we formulate the workload placement problem as a mixed nonlinear integer programming (MNIP) problem, with integer variables indicating the locations of workloads being placed and non-integer variables indicating the amount of capacity allocated to execute each workload. Then, we solve this MNIP problem in two steps. First, we fixed the integer variables and transform the MNIP problem to a convex optimization problem. Second, based on the non-integer variable being obtained in the first step, we further solve a combinatorial optimization problem which only contains the integer variables.

A. Problem Formulation

We first formulate the workload placement problem of edge cloud which has only one server at each tier. Without loss of generality, we assume there are m computing tasks with amounts of computations $\{w_1, w_2, ..., w_m\}$ and data sizes of program states $\{s_1, s_2, ..., s_m\}$, and n servers at the edge cloud with capacity $\{c_1, c_2, ..., c_n\}$ (i.e., CPU cycles per second). We formulate the problem of workload placement to be a mixed integer programming problem as

$$\begin{aligned}&\min f = \sum_{i=1}^m (\frac{w_i}{\lambda_i^{\gamma_i} c_{\gamma_i}} + (\gamma_i - 1)\frac{s_i}{B_{\gamma_i}}), \\ &\texttt{s.t.} \ \sum_{i \in \mathcal{O}_j} \lambda_i^j = 1, j = 1, 2, ..., n,\end{aligned} \tag{7}$$

Where γ_i indicates the server that task i is placed on, $\lambda_i^{\gamma_i}$ indicates the percentage of server γ_i's computational capacity being allocated for task i, and $\mathcal{O}_j$ denotes the set of computing tasks placed at server j. $w_i/\lambda_i^{\gamma_i} c_{\gamma_i}$ hence measures the computation delay of task i's execution. On the other hand, $(\gamma_i - 1)\frac{s_i}{B_{\gamma_i}}$ measures the communication delay of transmitting program i's states to server γ_i, which is determined by the network bandwidth B_{γ_i} allocated to γ_i.

B. Problem Transformation

The major challenge of solving Eq. (7) is that the value of $\lambda_i^{\gamma_i}$ depends on the corresponding γ_i. Hence, we first study the optimization problem in Eq. (7) when γ_i is known to each task i. We then have the following lemma:

***Lemma** 3:* When each γ_i is fixed, the optimization problem in Eq. (7) is a convex optimization problem.

Proof: For simplicity we use λ_i to substitute $\lambda_i^{\gamma_i}$, then f in Eq. (7) is a function of variables $\{\lambda_1, \lambda_2, ..., \lambda_m\}$. For each pair of (λ_i, λ_j), we have

$$\frac{\partial^2 f}{\partial \lambda_i \partial \lambda_j} = \begin{cases} \frac{1}{2} w_i c_{\gamma_i}^{-1} \lambda_i^{-3}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Hence the Hessian matrix $\mathbf{H} = (\frac{\partial^2 f}{\partial \lambda_i \partial \lambda_j})_{m \times m}$ of f is a symmetric and positive definite matrix. According to [22], function f is strictly convex if its Hessian matrix $\mathbf{H}$ is positive definite for all the variables. Since the constraint in Eq. (7) is linear, the optimization problem in Eq. (7) is a convex optimization problem. ■

Lemma 3 can be further extended to Theorem 3:

Theorem 3: When each γ_i is fixed, the optimal value of f is as follows:

$$f_{\min} = \sum_{j=1}^{n} \Big[\frac{(\sum_{i\in\mathcal{O}_j}\sqrt{w_i})^2}{c_j} + \sum_{i\in\mathcal{O}_j}(j-1)\frac{s_i}{B_j}\Big], \quad (9)$$

The corresponding optimal solution to Eq. (7) is

$$\lambda_i^* = \frac{\sqrt{w_i}}{\sum_{i\in\mathcal{O}_{\gamma_i}}\sqrt{w_i}} \quad (10)$$

Proof: Based on the convexity of the optimization problem in Eq. (7), the corresponding Lagrangian of f is:

$$\begin{aligned} L(\boldsymbol{\lambda},\boldsymbol{\mu}) &= \sum_{i=1}^{m}\Big(\frac{w_i}{\lambda_i^{\gamma_i}c_{\gamma_i}} + (\gamma_i-1)\frac{s_i}{B_{\gamma_i}}\Big) \\ &\quad - \sum_{j=1}^{n}\mu_j\Big(\sum_{k\in\mathcal{O}_{\gamma_i}}\lambda_k - 1\Big). \end{aligned} \quad (11)$$

By solving the KKT condition [22] of Eq. (11), we can obtain the optimal solution $\lambda_i^*(i=1,2...,m)$ and the corresponding optimal value $f_{\min}$. ∎

Theorem 3 shows that, when the placement for each workload is fixed, the corresponding capacity provisioning problem for these workloads is deterministic. Hence, we transform the workload placement problem to be a combinatorial optimization problem. For each possible workload placement, we can solve the corresponding convex optimization problem based on Theorem 3.

C. Design of Workload Placement Algorithm

Based on the above problem transformation, we then develop workload placement algorithms to determine the optimal values of $\{\gamma_i\}$. When there are m computing tasks and n servers, the time complexity to exhaustively search the solution space is $O(n^m)$. To further decrease the searching complexity and obtain the optimal solution, we propose to adopt the branch-and-bound approach. The basic idea of the branch-and-bound approach is to partition a given problem into a number of subproblems. This process of partitioning is usually called branching and its purpose is to establish subproblems that are easier to solve than the original problem because of their smaller size or more amenable structure. The subproblems usually constitute a search tree, where the non-leaf nodes represent subproblems, the leaves indicate the end of a branch and the edges represent decisions we made in each subproblem.

Without loss of generality, we assume that the workloads $\{w_1, w_2, ..., w_m\}$ are sequentially determined to be placed. Therefore, the subproblem of the workload placement problem is: given the workload placement of $\{w_i\}(i<k)$, which server should w_k be placed on. Based on this subproblem definition, we have the following lemma:

Lemma 4: Letting f_k indicate the total amount of delays for the k tasks having been branched, the lower bound on the total amount of delays for the remaining $m-k$ tasks is:

$$u_k = f_k + \sum_{i=k+1}^{m}\Big[\min_{1\le j\le n}\Big\{\frac{w_i}{c_j} + (j-1)\frac{s_i}{B_j}\Big\}\Big]. \quad (12)$$

Proof: In Eq. (12) all the remaining $m-k$ tasks are provisioned with 100% capacity of a server, therefore u_k provides a lower bound on the total amount of delays for the remaining $m-k$ tasks. ∎

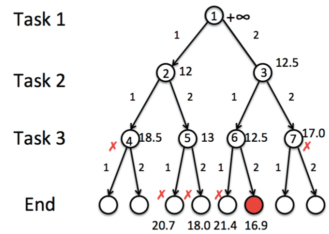


Fig. 3: The branch and bound method

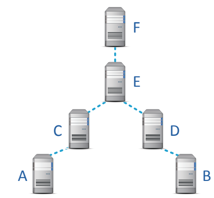


Fig. 4: Aggregation of optimal results

Based on Lemma 4, the branching process is a depth-first search over subproblems placing subsets of workloads, with the ones of the smallest lower bounds being searched first. Each time when a leaf node in the search tree is reached, all other branches with larger values of lower bounds are pruned. The optimal solution will be found until all the leaf nodes have been either searched or pruned.

We further illustrate this branching process using an example in Figure 3 where $m=3$ and $n=2$. $\{w_1,w_2,w_3\} = \{3,4,5\}$,$\{c_1,c_2\} = \{1,2\}$, and the communication delay between tier-1 and tier-2 servers for the 3 tasks are $\{2,3,3\}$. The number by the side of each node indicates the lower bound of this subproblem computed from Eq. (12) and the number on each edge indicates the decision of workload placement. For example, when task 1 is placed on tier-1, the lower bound of total delay on node 2 is $\frac{3}{1}+\frac{4}{1}+\frac{5}{1}=12$, when it is placed on tier-2, the lower bound on node 3 is $\frac{3}{2}+2+\frac{4}{1}+\frac{5}{1}=12.5$, since $12<12.5$, we further branch the next subproblem until we reach a leave node being labeled as red in Figure 3. Then, the leftmost and rightmost branches are automatically pruned because their lower bound is larger than 16.9. Therefore, the optimal solution of this example is $\{2,1,2\}$ with the total delay 16.9.

Algorithm 1: The simulated annealing process

Randomly generate an initial solution $\mathbf{x}_{old}$.
$T \leftarrow T_{max}$
while $T > T_{min}$ **do**
  Generate a new solution $\mathbf{x}_{new}$
  $\Delta d = f(\mathbf{x}_{new}) - f(\mathbf{x}_{old})$
  if $\Delta d < 0$ **then**
    $\mathbf{x}_{old} \leftarrow \mathbf{x}_{new}$
  end
  else if $Rand(0,1) < exp(-\frac{\Delta d}{T})$ **then**
    $\mathbf{x}_{old} \leftarrow \mathbf{x}_{new}$
  end
  $T \leftarrow \alpha \cdot T$
end

D. Aggregation of Optimization Results

As shown in Figure 4, multiple servers may be deployed at each tier of the edge cloud. We will aggregate the solutions to workload placement over different servers, to ensure the global optimality of workload placement. Our basic idea is to allocate the computational capacity of an edge cloud server on demand, when it receives mobile workloads from multiple low-tier servers. For example in Figure 4, server E allocates its capacity when it receives the workloads from servers C and D. Since there is one and only one optimal solution for a branch-and-bound searching process, the optimal allocation at a server that minimizes the cumulative delay of executing mobile programs is also unique.

The key challenge of aggregating the optimization results is how to allocate computational capacity of a higher-tier server to each branch of the edge cloud. Let $\mathbf{x}$ denotes the capacity allocation vector of a higher-tier server to its children in the edge cloud hierarchy, a straightforward solution to find out the optimal allocation is numerical iteration. However, traditional methods such as the Newton's method cannot ensure global optimality due to non-monotonicity of delay variation. Instead, we adopt the Simulated Annealing (SA) algorithm [23].

The SA process is described in Algorithm 1. In each iteration, we generate a new iteration state $\mathbf{x}_{new}$ from the previous state $\mathbf{x}_{old}$. The difference of total delay, denoted as Δd, is computed by solving smaller-scale optimization problems in Eq. (7) over different sets of low-tier servers. $\mathbf{x}_{new}$ will be immediately accepted if $\Delta d < 0$. Otherwise, we have a probability $exp(-\frac{\Delta d}{T})$ to accept $\mathbf{x}_{new}$, where T is the simulated temperature and decreases in each step of the iteration by a cooling parameter α $(0 < \alpha < 1)$. The length of the iterations in SA is determined by the initial temperature T_{max}, the terminating temperature T_{min} and the cooling parameter α. At the beginning T equals to T_{max}. T then decreases in each iteration, when $T = T_{min}$ the SA iterations terminate. Consequently, the SA process has no bounded time complexity. Instead, its time complexity is determined by the cooling parameter α, whose impact will be studied via extensive experiments in Section VII.

VI. System Experimentation

In this section, we implement the proposed hierarchical edge cloud architecture over a small-scale computing cluster and evaluate its performance, and compare the performance of hierarchical edge cloud with that of the flat edge cloud in the following settings. Without loss of generality, we measure the cloud performance using the average completion time of computing tasks. The smaller the completion time is, the larger amount of workload can be served by the edge cloud in the unit amount of time, indicating a larger amount of computational capacity being provided by the edge cloud.

- **Workload rate**: Each tier-1 server receives a number of new computing tasks every 5 seconds. The number of tasks, which indicates the *workload rate*, is determined every time following the same Poisson distribution.
- **Provisioned capacity**: The computational capacity provisioned to an edge cloud server is measured by the number of computing threads running concurrently on the server.

A. System Setup

We use multiple Dell OptiPlex 9010 desktop computers as edge cloud servers, and each server has an Intel i5-3475s@2.9GHz CPU and 8GB RAM. We use scale-invariant feature transform (SIFT) of images as computation tasks being submitted to edge cloud servers. SIFT is an algorithm in computer vision to detect and describe local features in images [24]. To study the usage of computational capacity of running this algorithm on servers, we create different numbers of computing threads to run SIFT over images with a size of 100KB, and the result is shown in Figure 5. When the number of concurrent threads is less than 4, the number of images processed per minute increases linearly. When the number of threads exceeds 4, the number of images processed per minute keeps stable. Hence, we consider 4 concurrent threads as the maximum capacity being provisioned to one server.

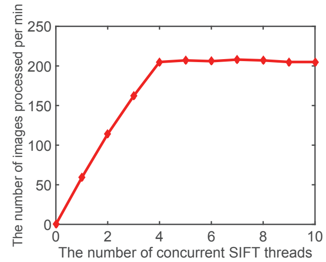


Fig. 5: Computational capacity of a server for SIFT

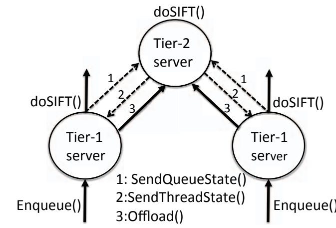
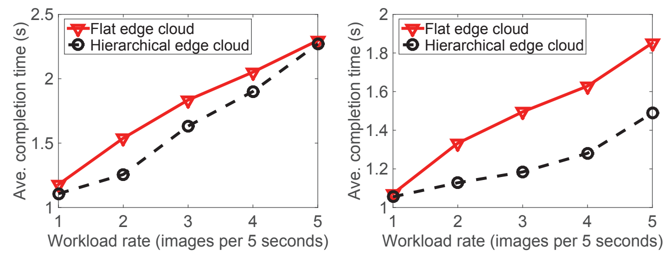


Fig. 6: Implementation of hierarchical edge cloud



(a) Provisioned capacity = 2

(b) Provisioned capacity = 4

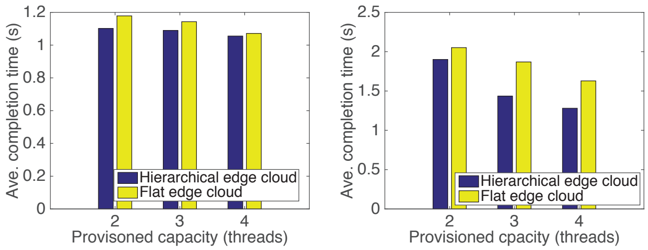
Fig. 7: Average completion time over different amounts of workloads

We then implement a hierarchical edge cloud with three servers, two of which are used as tier-1 servers and another is used as the tier-2 server. Figure 6 illustrates our implementation. Each tier-1 server has a workload queue. Whenever a task arrives, the server will invoke `Enqueue()` to put the workload into the queue. If the queue is not empty and there is a spare computing thread on the tier-1 server, one task will be popped out from the queue and processed. Otherwise if the tier-1 server is fully loaded, it will call `SendQueueState()` to check the availability of the tier-2 server. Only after having received the reply from the tier-2 server which invokes the `SendThreadState()` function and confirmed that the tier-2 server has a spare computing thread available, the tier-1 server will offload the task to the tier-2 server. In cases that two tier-1 servers are competing for the sole available computing thread at the tier-2 server, each of them has a probability of 0.5 to have their tasks to be processed by the tier-2 server.

The initial capacity of each tier-1 server is 1, i.e., each server only allows one thread at any time. We compare the performance of hierarchical and flat edge cloud designs by provisioning different amounts (C) of capacity to the edge cloud. For hierarchical edge cloud, C indicates the capacity being provisioned to the tier-2 server. For flat edge cloud, $C/2$ is provisioned additionally to each tier-1 server. Each experiment lasts five minutes.

B. Experiment results

Two sets of experiments are conducted. First, we fix the provisioned capacity and change the workload rate form 1 to



(a) Workload rate = 1

(b) Workload rate = 4

Fig. 8: Average completion time over different amounts of provisioned capacities

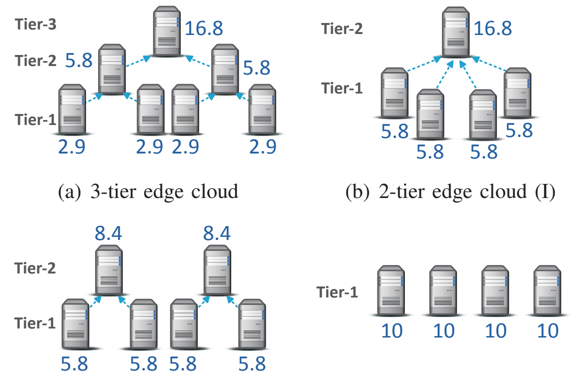
5 to evaluate the impact of different amounts of workloads on the edge cloud performance. Second, we fix the workload rate and change the provisioned capacity from 2 to 4 to evaluate the difference of cloud performance under different capacity situations. The results are shown in Figure 7 and Figure 8, respectively.

Case 1: Light workload, i.e., the workload rate is low with respect to the computational capacities at edge cloud servers. In this case, the performance difference between the flat and hierarchical edge clouds is small, because most computing tasks could be completed by tier-1 servers themselves. For example in Figure 8(a) when the workload rate is 1, the hierarchical edge cloud can reduce the average completion time by 0.1s or less, no matter how much capacity is provisioned to the tier-2 server.

Case 2: Heavy workload, i.e., the workload rate is higher with respect to to the computational capacities at edge cloud servers. In this case, the proposed hierarchical edge cloud can significantly reduce the average completion time of computing tasks by up to 30%. As shown in Figures 7(b) and 8(b), especially when a large amount of capacity is provisioned at the tier-2 server, there is large difference between the performance of flat and hierarchical designs of the edge cloud. The major reason for this difference is that the tier-2 server is able to efficiently aggregate the peak loads from all tier-1 servers, and hence utilizes the provisioned capacity to process computing tasks at a much higher efficiency.

Case 3: System overload, i.e., the workload being generated is too much to be processed by the edge cloud servers. In this case, hierarchical edge cloud can reduce the completion time compared to the flat edge cloud, but the difference is very small. As Figure 7(a) shows, when provisioned capacity is 2 but the workload rate is 5, the average completion time on both two clouds are almost the same. This is because, when the workload rate is too high or the provisioned capacity is too small, both of tier-1 servers will receive their peak loads simultaneously and offload the peak loads to the tier-2 server. This will further overload the tier-2 server, resulting in longer completion time over all computing tasks.

All the experiments above are conducted with two tier-1 servers. Note that when there are more tier-1 servers, given the same amount of capacity being provisioned to the tier-2 server, our proposed hierarchical edge cloud would achieve further reduction of the average completion time of computing tasks. The basic reason is that in this case, each tier-1 server in the flat edge cloud will have less capacity, and hence have smaller chance to successfully serve the peak load.



(a) 3-tier edge cloud

(b) 2-tier edge cloud (I)

(c) 2-tier edge cloud (II)

(d) Flat edge cloud

Fig. 9: Edge cloud topologies for simulations. The number by each server indicates the amount of capacity being provisioned.

VII. Trace-based Simulation

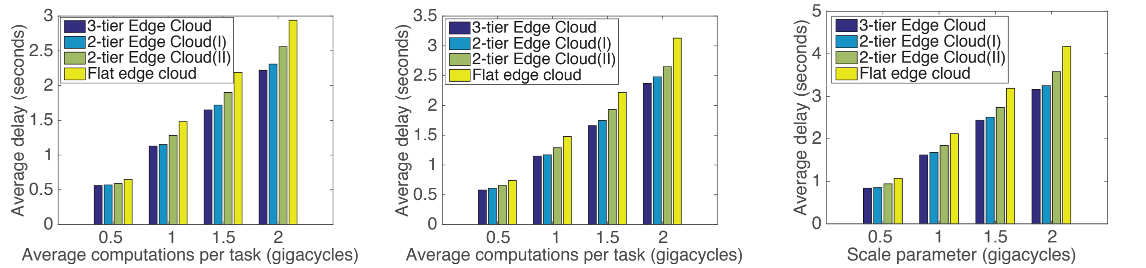
In this section, we conduct simulation experiments over a larger topology of hierarchical edge cloud, and evaluate the performance of our proposed workload placement algorithm. We measure the performance of edge cloud using the average delay of the computing tasks, which includes both computation and communication delay.

A. Experiment Setup

We adopt a real data center network trace from Wikipedia [20] to determine the amount of computing tasks being received by each tier-1 edge cloud server. This trace records the arrival times of 10% of all user requests to Wikipedia during the period between 09/19/2007 and 01/02/2008, and the requests of each user is independent from all others. For each tier-1 server, we randomly select one segment from the trace that contains 1000 user requests, and count the number of user requests in every 5 milliseconds. Each count is then used as the number of workloads arrived in every 5 seconds at each tier-1 server. Then, we vary the following settings in our experiments:

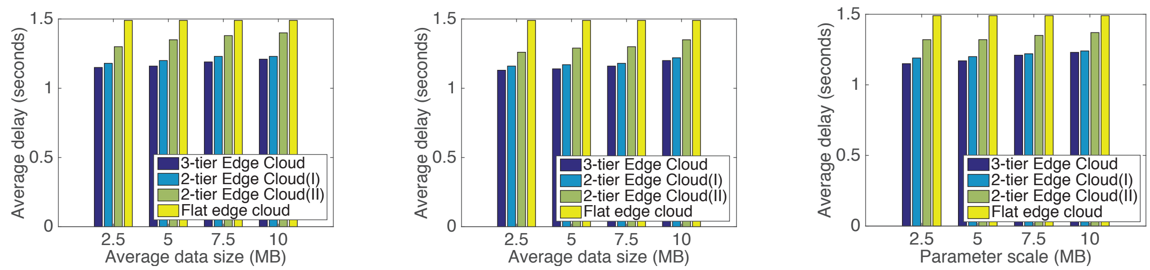
- **Amount of computations**: For each computing task being received by a tier-1 server, we specify its amount of computation needed as the number of CPU cycles. We still use SIFT as the application being executed. Since SIFS has a similar computation requirements with face recognition application in [21] which needs 1 gigacycles, we use 1 gigacycles as the default amount of computation for all tasks. The specific amount of each task is then determined independently as random numbers following different distributions.
- **Data size**: Each computing task is also associated with a data size, which indicates the size of program states being sent to the edge cloud. Each workload size is also generated from a probabilistic distribution.

In our experiments, mobile workloads are received by 4 tier-1 edge servers. As shown in Figure 9, our experiments are then conducted over different settings of the hierarchical edge clouds with different topologies and computational capacity being provisioned. Every two cloud servers are connected via a 100Mbps network link. A total amount of 40 GHz computational capacity (in terms of the amount of CPU cycles



(a) Uniform distribution (variance = 0.08) (b) Normal distribution (variance = 0.1) (c) Pareto distribution (shape parameter = 3)

Fig. 10: Average delay over different amounts of computations in workloads



(a) Uniform distribution (variance = 0.33) (b) Normal distribution (variance = 1) (c) Pareto distribution (shape parameter = 3)

Fig. 11: Average delay over different data sizes of workloads

per second) is provisioned. Particularly, in the 3-tier edge cloud, each tier-1 server is provisioned with 2.9 GHz capacity, which is the same with the CPU frequency of servers we use in hardware experiments. Thus, if a computing task with 1 gigacycles is offloaded to a tier-1 server, its execution time will be $1/2.9$ seconds. In other topologies with less numbers of cloud tiers, we enlarge the capacity of tier-1 servers accordingly.

B. The Effect of Computation Amounts

We first evaluate the performance of our proposed workload placement algorithm with different amounts of computations that are included in mobile workloads. The data size for each workload is generated from a normal distribution with an average of 5 MB. We generate the amount of computations of each workload using three distributions: uniform distribution, normal distribution and Pareto distribution. For each distribution, we adjust its parameter and evaluate the performance. The cooling parameter α is set to 0.9, and the initial temperature and terminating temperature are set to 100 and 0.01 respectively.

The results are shown in Figure 10. The 3-tier hierarchical edge cloud can reduce the average task completion delay by 30%, compared to that of the flat edge cloud. Comparatively, the delay reductions achieved by the 2-tier edge clouds are about 25%, mainly due to its incapability of aggregating enough peak loads from lower tiers. In particular, the 3-tier hierarchical edge cloud outperforms 2-tier edge cloud (I) which has only one tier-2 server, although these topologies have the same computational capacity. The major reason is that the 3-tier edge cloud provisions more capacity on the higher tiers and less capacity on the first tier, leading to a more efficient iterative aggregation of the peak load. Figure 10 shows that by aggregating peak loads from the lower tiers and optimizing the placement of these workloads, our proposed hierarchical architecture dramatically reduces the total delay of task completion.

Meanwhile, Figure 10 indicates that average delay of the workloads is affected by their amounts of workload computations. For all the four topologies, larger amounts of computations lead to more delay reduction. When the average amount of computation increases to 2 gigacycles, the delay reduction could be up to 40%.

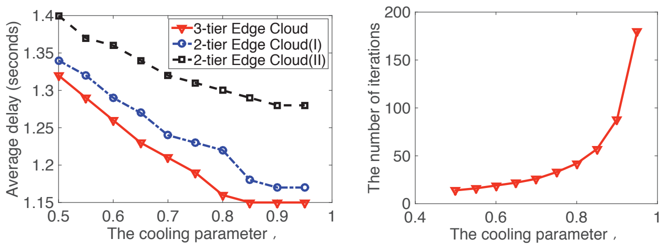
C. The Effect of Data Size

In this section, we evaluate the performance of the hierarchical edge cloud over different data sizes of workloads. The amount of computation per task is generated from a normal distribution with an average amount of 1 gigacycles per task. We generate the data size of each workload subject to three distributions: uniform distribution, normal distribution and Pareto distribution, then adjust the parameters of these distributions. We use the same setting on the SA process as that in the previous section.

As shown in Figure 11, our proposed hierarchical edge cloud significantly reduces the delay by 25% with different data sizes of workloads. Although larger data sizes increase the communication delay for transmitting workloads to high tiers, our proposed workload placement algorithm can still optimize the placement of these workloads and further reduces the total delay. Particularly, the proposed 3-tier edge cloud also outperforms other hierarchical edge clouds due to the same reason in Section VII-B. Note that the variation of the data sizes have no impact on the total delay under the flat edge cloud because there is no communication delay between tiers in a flat architecture.

D. The Impact of SA Cooling Parameter

In this section, we study the impact of the SA cooling parameter α on the performance and time complexity of workload placement in the hierarchical edge cloud. We generate



(a) The average delay (b) The number of iterations

Fig. 12: The effect of the cooling parameter

the amount of computation of each workload as a random number subject to normal distribution $\mathcal{N}(1, 0.1)$, and each data size with a normal distribution $\mathcal{N}(5, 0.1)$ as well. We then adjust the value of cooling parameter (α) and solve the workload placement problem in different hierarchical edge clouds as shown in Figure 9. We set the initial temperature and terminating temperature to 100 and 0.01.

The experiment results are shown in Figure 12. Since a larger value of α helps the SA process to converge, it further reduces the total delay of task completion. On the other hand, when the value of α is too small (< 0.5), the iterations may not be enough for the SA process to converge, incurring higher delay of task completion. Besides, due to the speed-up of the cooling process, it is unlikely to accept a temporarily worse solution, and it is much easier for the SA process to fall into a local optima. Figure 12(a) also shows that the 3-tier edge cloud has the lowest delay compared to other 2-tier edge clouds under different cooling parameters, even though the result has not reached the convergent value. The reason is that although the cooling process accelerates, the 3-tier edge cloud can still aggregate more peak loads from lower tiers compared to the 2-tier edge clouds.

The corresponding number of SA iterations with different values of α are shown in Figure 12(b). Since we use the same initial temperature, cooling parameter and terminating temperature in the SA process for the three hierarchical edge clouds, they have the same number of iterations when their SA processes terminate. Figure 12(b) shows that a higher value of α leads to a larger number of SA iterations because when the value of α increases, the temperature needs more steps to be cooled down to the terminating temperature.

The results from Figure 12(a) and Figure 12(b) show that there exists a tradeoff between the performance and overhead of workload placement in the hierarchical edge cloud. Using a larger value of cooling parameter helps the SA process converge and reduces the task completion delay, but also incurs a larger number of SA iterations and increases the computational overhead of workload placement. In practice, our design offers the cloud system administrator the full flexibility to adjust such parameter at run-time.

VIII. Conclusion

In this paper, we propose a hierarchical edge cloud architecture to efficiently and promptly serve the peak loads from mobile users. The advantage of this hierarchical architecture over traditional flat edge cloud is demonstrated through formal analysis and systematic experimentation. We further develop workload placement algorithms to efficiently execute mobile programs in the edge cloud. The effectiveness of the proposed algorithm is verified by trace-based simulations.

References

[1] Gordon, M. S., Jamshidi, D. A., Mahlke, S., Mao, Z. M., and Chen, X. Comet: Code offload by migrating execution transparently. In *Proceedings of USENIX OSDI*, 2012.

[2] Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., and Satyanarayanan, M. Towards wearable cognitive assistance. In *Proceedings of ACM MobiSys*, 2014.

[3] Ha, K., Pillai, P., Richter, W., Abe, Y., and Satyanarayanan, M. Just-in-time provisioning for cyber foraging. In *Proceeding of ACM MobiSys*, 2013.

[4] Hao, F., Kodialam, M., Lakshman, T., and Mukherjee, S. Online allocation of virtual machines in a distributed cloud. In *Proceedings of IEEE INFOCOM*, 2014.

[5] Kosta, S., Aucinas, A., Hui, P., Mortier, R., and Zhang, X. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proceedings of IEEE INFOCOM*, 2012.

[6] Lee, K., Chu, D., Cuervo, E., Wolman, A., and Flinn, J. Delorean: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *Proceedings of ACM MobiSys*, 2014.

[7] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, vol. 99, 2011.

[8] Shahzad, M., Liu, A. X., and Samuel, A. Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In *Proceedings of ACM MobiCom*, 2013.

[9] Verbelen, T., Simoens, P., De Turck, F., and Dhoedt, B. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the 3rd ACM Workshop on Mobile Cloud Computing and Services*, 2012.

[10] Zhang, J., Ren, F., and Lin, C. Delay guaranteed live migration of virtual machines. In *Proceedings of IEEE INFOCOM*, 2014.

[11] Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. Workload analysis and demand prediction of enterprise data center applications. In *Proceedings of the IEEE 10th International Symposium on Workload Characterization*, 2007.

[12] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. A View of Cloud Computing. In *Communications of the ACM*, 53(4):50-58, 2010.

[13] Wang, M., Meng, X., and Zhang, L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *Proceedings of IEEE INFOCOM*, 2011.

[14] Breitgand, D., Epstein, L. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In *Proceedings of IEEE INFOCOM*, 2012.

[15] Verma, A., Bagrodia, J., Jaiswal, V. Virtual machine consolidation in the wild. In *Proceedings of the 15th International Middleware Conference*, 2014.

[16] Maguluri, S., Srikant, R., Ying, L. Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters. In *Proceedings of IEEE INFOCOM*, 2012.

[17] Chen, L., Shen, H., Sapra, K. RIAL: Resource Intensity Aware Load Balancing in Clouds. In *Proceedings of IEEE INFOCOM*, 2014.

[18] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of ACM MobiSys*, pages 49–62, 2010.

[19] Zhao, J.,Yang, K.,Wei, X., Ding, Y.,Hu, L., Xu, G. A Heuristic Clustering-based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment. In *IEEE Transactions on Parallel and Distributed Systems*, 2015.

[20] Urdaneta, G., Pierre, G, and Van, M. Wikipedia Workload Analysis for Decentralized Hosting. In *Elsevier Computer Networks*, vol.53, 2009.

[21] Xu,C Decentralized Computation Offloading Game for Mobile Cloud Computing. In *Parallel and Distributed Systems, IEEE Transactions on*, vol.26, 2014.

[22] Boyd, S. and Vandenberghe, L. Convex optimization. *Cambridge university press*, 2009.

[23] Kirkpartrick, S., Gelatt, C., Vecchi, M. Optimization by Simulated Annealing. In *Science, New Series*, vol.220, 1983.

[24] Lowe, D. Object Recognition from Local Scale-invariant Features. In *Proceedings of the International Conference on Computer Vision*, 1999.