# *Zenith*: Utility-aware Resource Allocation for Edge Computing

Jinlai Xu*, Balaji Palanisamy*, Heiko Ludwig† and Qingyang Wang‡
*School of Information Sciences, University of Pittsburgh, Pittsburgh, PA 15213, USA
Email: {jinlai.xu, bpalan}@pitt.edu
†Almaden Research Center, IBM Research, San Jose, CA 95120, USA
Email: hludwig@us.ibm.com
‡Computer Science and Engineering, Louisiana State University, Baton Rouge, LA 70803, USA
Email: qywang@csc.lsu.edu

***Abstract*—In the Internet of Things(IoT) era, the demands for low-latency computing for time-sensitive applications (e.g., location-based augmented reality games, real-time smart grid management, real-time navigation using wearables) has been growing rapidly. Edge Computing provides an additional layer of infrastructure to fill latency gaps between the IoT devices and the back-end computing infrastructure. In the edge computing model, small-scale micro-datacenters that represent ad-hoc and distributed collection of computing infrastructure pose new challenges in terms of management and effective resource sharing to achieve a globally efficient resource allocation. In this paper, we propose *Zenith*, a novel model for allocating computing resources in an edge computing platform that allows service providers to establish resource sharing contracts with edge infrastructure providers *apriori*. Based on the established contracts, service providers employ a latency-aware scheduling and resource provisioning algorithm that enables tasks to complete and meet their latency requirements. The proposed techniques are evaluated through extensive experiments that demonstrate the effectiveness, scalability and performance efficiency of the proposed model.**

***Index Terms*—fog computing; edge computing; resource allocation**

## I. Introduction

In the Internet of Things(IoT) [1] era, the demands for low-latency computing for time-sensitive applications (e.g., location-based augmented reality games, real-time smart grid management, real-time navigation using wearables) has been growing rapidly [2]. Edge Computing provides an additional layer of infrastructure to fill latency gaps between the IoT devices and the backend computing infrastructure. As shown in Figure 1, an Edge/Fog Computing model [3]–[9] provides an additional layer of computing infrastructure for storing and processing data at the edge, allowing low latency applications to meet their response time requirements effectively. The notion of Micro DataCenters(MDCs) [10] in an edge computing platform makes it possible for IoT applications to process data and access computational resources located closer to the endpoints, providing low response time guarantees to latency-sensitive applications that may operate on these platforms. Figure 1 shows that the Edge Computing layer represents the infrastructure located closer to the endpoints and includes a more geo-distributed collection of adhoc MDC resources spread geographically. The architecture primarily

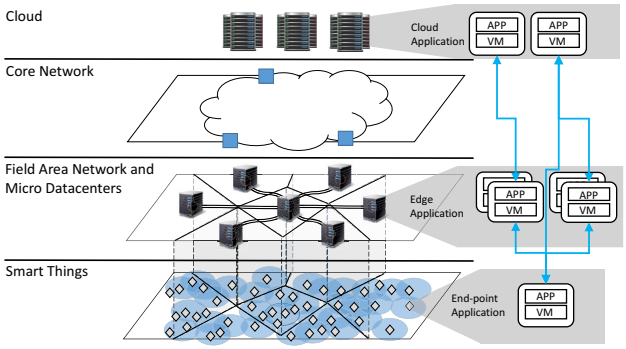



Fig. 1. Edge Computing Model

provides a solution for applications that have higher sensitivity to network latency or incur a higher network cost. Here, small-scale MDCs that represent ad-hoc and distributed collection of computing infrastructure pose new challenges in terms of management and efficient resource sharing towards to achieve a globally efficient resource allocation.

There has been a few recent work [11], [12] addressing the resource management and resource provisioning challenges in the edge computing model. A fundamental assumption in these solutions includes a tight coupling of the management of the Edge Computing Infrastructures(ECIs) with that of the service management performed by Service Providers(SPs), which means that the computational resources present at the edge MDCs are coupled and controlled directly by edge Service Providers(SPs). We argue that such a coupled model for management of Edge Computing Infrastructures (ECIs) by Service Providers (SPs) significantly limits the cost-effectiveness and the opportunities for latency-optimized provisioning of edge infrastructure resource to applications. When the management of the Edge computing infrastructures are controlled by the SPs, it results in an increased infrastructure cost and a decrease in the overall utilization of the system leading to poor cost-effectiveness.

In this paper, we propose *Zenith*, a new resource allocation model for allocating computing resources in an edge computing platform that allows edge service providers to establish

978-1-5386-2017-5/17 $31.00 © 2017 IEEE
DOI 10.1109/IEEE.EDGE.2017.15

resource sharing contracts with edge infrastructure providers *apriori*. Based on the established contracts, service providers employ a latency-aware scheduling and resource provisioning algorithm that enables tasks to complete and meet their latency requirements while achieving both global and local resource allocation efficiency and fairness. Unlike existing solutions that perform a coupled management of ECIs and SPs, *Zenith* employs a decoupled model where the management of ECIs is independent of that of the SPs which in turn provides increased resource utilization and minimizes job execution latency.

Concretely, this paper makes the following contributions: first, we propose *Zenith*, a decoupled resource allocation model that manages the allocation of computing resources distributed at the edges independent of the service provisioning management performed at the service provider end. Second based on the model, we develop an auction-based resource sharing contract establishment and allocation mechanism that ensures truthfulness and utility-maximization for both the ECIPs (Edge Computing Infrastructure Providers) and SPs (Service Providers). Third, we develop a latency-aware task scheduling mechanism that allocates the resources committed in the contracts to specific jobs in the workloads. Finally, we evaluate the proposed techniques through extensive experiments that demonstrate the effectiveness, scalability, and performance of the proposed model.

The remainder of this paper is organized as follows. Section II provides a background of various existing edge computing solutions and motivates the proposed resource allocation model. In Section III, we present the *Zenith* architecture for decoupled resource management and introduce the system model. In Section IV, we present the proposed resource allocation framework that comprises of the contract establishment process and the task scheduling mechanism. Section V presents the performance evaluation of *Zenith* through extensive experiments. Section VI discusses the related work and we conclude in Section VII.

## II. Background & Motivation

There has been an increasing growth in modern low-latency computing applications using wearables and IoT technologies that include (i) augmented reality applications [13], (ii) real-time traffic control systems [14] that require low-latency responses to avoid potential collisions, (iii) real-time smart grid management systems [15] that aggregate data from geo-distributed sensors and control the grid in real time. Though Cloud Computing has been a very cost-effective solution [16] [17] to several computing needs, clouds fail to meet low latency requirements of modern computing applications that demand strict guarantees on response times. Edge Computing [5]–[8] complements the backend computing provided by clouds to fill the critical latency gaps between the endpoints and the Cloud.

To achieve efficient processing at the edge, smart gateways [18] and Micro DataCenters(MDCs) [10] are two key methods proposed in the literature. A smart gateway is a device which is placed at the edge of the network near the sensors. It provides a platform for the edge applications to intermediately operate the data from the endpoints to the Cloud or directly respond to the requests from the endpoint applications. A Micro Datacenter (MDC) is a data center which has a small number of resources and located close to the edge of the network to support Edge Computing services. MDCs are densely geo-distributed to provide a low and predictable latency infrastructure to the endpoint applications. Compared with smart gateways, MDCs are obviously more powerful and contain more servers and possess higher computing capacity than smart gateways. In addition, MDCs are more configurable than smart gateways as smart gateways are often purchased by the end user. Therefore the end users have the rights to control the smart gateways which limit the control for the service providers.

In this work, we consider MDCs as the source of computational resources in the edge computing platform and the goal of the MDCs is to support low latency applications at the edge, enabling them to meet stronger guarantees on response time. To effectively manage and leverage MDCs in an edge computing platform, there are several key challenges that need to be addressed. For instance, an effective resource management of MDCs should address (i) how to provision the application containers [19], [20] to serve jobs to maximize the utility of the services and (ii) how to schedule workloads on the application containers to both cover the demands and satisfy the latency constraints. While edge computing as a research area is emerging fast, there are a few prior efforts that discuss the above challenges [11], [12]. A fundamental assumption in these solutions includes a tight coupling of the management of the Edge Computing Infrastructures(ECIs) with that of service management by Service Providers(SPs), which means that the computational resources present at the edge MDCs are coupled and controlled directly by edge Service Providers(SPs). We argue that such a coupled model for management of Edge Computing Infrastructures (ECIs) by Service Providers (SPs) significantly limits the cost-effectiveness and the opportunities for latency-optimized provisioning of edge infrastructure resource to applications.

In contrast to existing solutions, our proposed model, *Zenith* decouples the infrastructure management from service management, enabling the ECIs to be managed by ECIPs independently of the service provisioning and service management at the SPs. Such a decoupled model enables ECIPs to join up to establish an Edge Computing Infrastructure Federation(ECIF) to provide resources to the Edge Computing applications provisioned and managed by the SPs. In addition, the model provides increased opportunities for resource consolidation and utilization as the geo-distributed ECIs can be jointly managed and allocated to maximize application utility and minimize cost. In the next section, we introduce the architectural details of *Zenith* and present its system model.

## III. Zenith: System Architecture and Model

We introduce the system architecture and describe the individual components of the *Zenith* system model.

### A. System Architecture

As shown in Figure 1, the proposed system uses a layered architecture [21]. In the bottom layer, the smart things represent the end devices (e.g. sensors, smart gateways, smart phones) that act as the endpoints in the Edge Computing platform. The field area network layer is the layer where MDCs and the Edge Computing services are placed. The core network layer provides the back bone of the wide area network connecting the field area networks at the edge with the cloud's large-scale datacenters that may be located at a farther distance from the local field area network. In the resource allocation model of *Zenith*, the service management and the infrastructure management are decoupled. In other words, the service management is handled by the Service Provider(SP) that determines the provisioning decisions such as (i) where to place the containers to meet the latency requirements of the services, (ii) how many tasks in the workload are scheduled to a single container and (iii) increasing the number of containers to support the oncoming workloads. The infrastructure management is performed by the Edge Computing Infrastructure Provider(ECIP) which invests and operates the infrastructures for supporting the services placed at the edges. The ECIPs are federated to set up an Edge Computing Infrastructure Federation(ECIF) which provides a resource market for the SPs wanting to deploy edge computing services at the edge. Each ECIP manages several adhoc MDCs which can be densely geo-distributed. The resources of MDCs are leased or provisioned on-demand to SPs by agreeing on a contract agreement between the ECIPs and SPs. The contract may include the duties and rights between the ECIP managing the resource and the SP that uses the resources.

### B. System Model

We next describe the system model for *Zenith* in five steps: first, we describe the features of the Service Providers(SPs) that provide Edge Computing services. Next, we represent the features of Edge Computing Infrastructure Providers(ECIPs) which provide infrastructure services for Edge Computing. We then illustrate the region division process for simplifying the resource discovery problems. After that, the agreements and the responsibilities of the coordinator are presented and finally, we discuss the role of the contract manager which is part of the *Zenith* model to manage the resource sharing contracts that are agreed between the ECIPs and SPs.

*1) Service Provider:* We consider there are $N$ SPs that require edge computing infrastructure to support their services. For simplicity, we assume that for each SP $i \in [1, N]$, it only runs one service. This model can be easily extended to one SP running multiple services with additional small changes. For each service, there is a quantifiable service demand of the SPs in each geographic region for every discrete time slot of a day.

The application container [19], [20](a configured VM integrated with the service software) has several requirements such as CPU consumption, memory size, network bandwidth and latency requirement. We use the workload demand to estimate the required number of the application containers to service the workload. Therefore, when the demand increases, the SPs begin to start adding more containers to serve the workload.

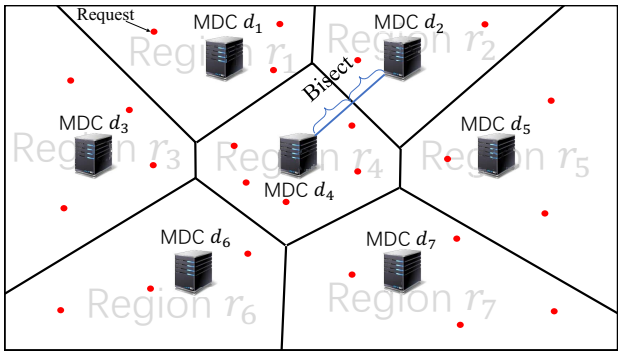



Fig. 2. An illustration of a WVD in Zenith with seven MDCs

*2) Edge Computing Infrastructure Provider:* Each ECIP handles a large number of highly geo-distributed Micro DataCenters (MDCs) and each MDC is operated by one ECIP. We assume each MDC $d \in [1, R]$ has several servers for the infrastructure service. It has a server list $M_d$ which contains all the servers controlled by the MDC $d$. The capacity of one server $m \in M_d(\tau)$ is $C_d^m$. For simplicity, we assume that every container consumes equal resources for running the application service. The capacity for one server, $C_d^m$ can be also represented as the number of containers which can be run on the server.

*3) Regions Division:* The problem of choosing the right MDCs to minimize the latency for every end-point and every Edge Computing service in the geographic map is intractable:

**Theorem 1.** *The placement decision of which MDCs should host which edge computing application in order to maximize utility in terms of response time and bandwidth is NP-hard.*

*Proof.* In order to show that the problem is NP-Hard, we first simplify the problem by assuming that each MDC can only run one container and each SP only needs to place several containers to some of the MDCs and we show that the simplified version is NP- Hard. Here, the optimization problem is to minimize the response time (latency) between the edge containers to the end-users. We can map the containers and users as the facilities and the MDCs as the locations in a Quadratic Assignment Problem (QAP) [22] which is shown to be an NP-Hard problem. As the simplified problem is a QAP problem which is intractable, the original container placement decision problem is also NP-Hard. □

For solving the problem, we use Weighted Voronoi Diagrams (WVD) [23] a technique widely used in GIS, sensor networks and wireless networks [24] for making placement decisions to maximize the utility function. The use of Weighted Voronoi Diagrams (WVD) simplifies the latency minimizing problem to a map division problem which can be solved by buiding the WVD in a polynomial time. In the example shown in Figure 2, the geo-location is divided into seven regions by

the WVD generating algorithm with the seven MDCs as the sites in WVD. With default condition that all the sites have equal weights, the polygon of each region divides the map and all the positions in the polygon are close to the site of the region. Therefore, for each smart thing, the nearest MDC which can serve its request at the edge is located in the region where the smart thing belongs to. This method simplifies the model, especially the process of estimating the latency to the end users by dividing the map into several regions and registering an area to one region. The Voronoi Diagrams are predetermined by considering the location of the MDCs as the sites and the expected workload of the services. The weight for each MDC can be calculated as the ratio of the capacity of the MDC to the historical workload amount in the nearby area (e.g., an area within 30-mile radius). Thus, a micro datacenter which has a higher workload pressure in the adjacent area will handle a smaller region in the WVD.

We assume that the predetermined WVD divides the map into $R$ sub-regions. Each region $r \in [1, R]$ only contains one MDC and the nearest MDC for every position in region $r$ is the MDC $d = r$ in that region. We also assume that the expected workload distribution for each time slot $\tau$ is $\lambda_i^r(\tau)$ for SP $i$ in the region $r$. The workload distribution contains all the workloads coming from the region. We use $\lambda_i^p(\tau) \in \lambda_i^r(\tau)$ to represent the workload coming from a particular position $p$ in region $r$. As we primarily consider the workload which needs real-time serving, $\lambda_i^p(\tau)$ is often the upper bound of the workload during the time slot $\tau$ from position $p$.

*4) Coordinator:* The coordinator is a third-party service which is trusted by the ECIPs and SPs in the system and it is responsible for providing a platform for the ECIPs to trade resources with the SPs. There is an agreement which is committed with the coordinator before ECIPs and SPs join the federation. The agreement stipulates the rights and duties of the three parties, the coordinator, the SPs and the ECIPs.

*5) Contract Manager:* The contract manager is a component associated with both the SP and ECIP to manage the resource sharing contracts agreed by them. Its responsibility is to manage the resource sharing contracts and observe the contracts' status. For the SP which buys resources, it must pay for the contract and has the right to observe the performance of the resource that is allocated to it. For the ECIP which sells resources, it must guarantee the performance of the resources which are leased to the buyers. It collects the payments from the buyers. The contract is an agreement between the SP and the ECIP to lease resources from the ECIP's MDC which is effective in a particular time period with a particular constraint such as latency and availability.

## IV. Zenith: Resource Allocation

In this section, we present the proposed resource allocation techniques for ECIPs and SPs to establish relationships(contracts) with each other and discuss the job scheduling technique employed in *Zenith*.

### A. Contracts Establishment

The key idea behind the contract establishment process is to match the demands (e.g. workload and revenue) of the SPs and the supplies(e.g. capacity and operating cost) of the MDCs. The SPs want to maximize their utility of serving the customer with a better quality of service to potentially gain more profits. The MDCs want to maximize their utility (revenue) by renting their servers to more SPs and the SPs who can pay more.

For the sake of model simplicity, in this subsection, we only model the resource sharing problem for one MDC though the model is generic to be extended to the scenarios where there are multiple MDCs. As the WVD algorithm divides the map into several regions, the problem of finding the MDC with the lowest latency is transformed into a problem of determining which region a smart thing belongs to. In each region, every SP estimates the workload in that region based on historical workload information and statistical prediction. It then bids for the resource for running the application containers for serving the workload in that region. The bid is decided by the workload demand, $\lambda_i^r(\tau)$, the latency requirement and the estimated utility that the SP can gain from running the service in the MDC for serving the customers.

*1) Utility of SPs:* First, we model the utility of the SP to run the service on the edge. Here, we consider services having higher requirements for latency such as location-based augmented reality games [13] and intelligent traffic light control [25]. The utility of the SP can be expressed by the gain in changing the execution of the real-time service from the cloud to the edge, which we represent by the function:

$$u_i^p(\tau) = f(l_{pd}(\tau)) - f(l_{pi}(\tau)) \tag{1}$$

where $f(x)$ is a function which estimates the utility that can be obtained by providing the service with a latency $x$. It can be approximated by an affine utility function which translates the user-perceived criterion(latency) into utility(e.g., revenue), $f(x) = -ax+b$ where $a$ and $b$ are the parameters in the affine utility function and $l_{pd}(\tau)$ represents the latency between the position $p$ where the workload comes from and the MDC $d$ in time slot $\tau$. Here $l_{pi}(\tau)$ represents the latency between the mega datacenter of SP $i$ and the position $p$.

*2) Utility of ECIPs:* For the EFIP, its objective is to earn higher revenue by providing the infrastructure to SPs. So the utility for the EFIP is obviously the profit that it can obtain by renting the resource to the SPs. For each MDC, the utility function $u_d(\tau)$ can be defined as the profit of selling the resource:

$$u_d(\tau) = \sum_{m}^{M_d} (C_d^m * \pi_d^s(\tau) - Cost_d^m(\tau)) \tag{2}$$

where $\pi_d^s(\tau)$ is the sell price in time slot $\tau$ for MDC $d$, $Cost_d^m(\tau)$ is the fluctuating operating cost of server $m$ in MDC $d$ in time slot $\tau$.

*3) Bidding Strategy:* For SP, the bidding strategy is to bid by the true value that the SP believes for the resources, which is represented by the utility function we discussed above. The

bid can be represented as $< b_i^p(\tau), \lambda_i^p(\tau) >$, where $b_i^p(\tau)$ represents the bid price for each position the workload comes from. The bid price $b_i^p(\tau)$ can be estimated by the utility of SP $i$ in time slot $\tau$ for running the service on the edge instead of on the cloud. So the bid price for each position $p$ can be calculated as $b_i^p(\tau) = u_i^p(\tau)$.

For MDC, the sell bid is set to the operating cost, which means if the bid wins, the MDC can at least break even the cost. The sell bid can be represented as $s_d(\tau) = Cost_d(\tau)$, where $Cost_d(\tau)$ represents the operating cost of running one application container for MDC $d$ in time slot $\tau$.

### *B. Determining Winning Bids*

After designing the bidding strategy of the SPs and MDCs, we next design our algorithm for determining the winning bids as shown in Algorithm 1. The winning bids decision algorithm is based on the McAfee mechanism [26]. It guarantees *truthfulness* and *budget balance* for the auction. *Truthfulness* provides a huge benefit for designing the auction which simplifies the bidding strategies for all the participants. If the auction mechanism satisfies *truthfulness*, it ensures that the strategy which bids with the true value is the dominant strategy among all the other strategies. The *budget balance* is a feature which guarantees that the auctioneer will not subsidize for the auction, which means the payment from the buyers is always more than the payment to the sellers.

**Algorithm 1:** Algorithm for winners selection

**Input :** MDC #: $d$;
Buy bids: $B(\tau) = \{< b_1^{p_1}(\tau), \lambda_1^{p_1}(\tau) >, < b_1^{p_2}(\tau), \lambda_1^{p_2}(\tau) >, ..., < b_2^{p_1}(\tau), \lambda_2^{p_1}(\tau) >, ...\}$;
Operating Cost: $Cost_d(\tau)$
**Output:** Clearing Buying Price: $\pi_d^b(\tau)$ Clearing Selling Price: $\pi_d^s(\tau)$;
Auction decision:
$X^r(\tau) = \{< x_1^{p_1}(\tau) >, < x_1^{p_2}(\tau) >, ..., < x_2^{p_1}(\tau) >, ...\}$;
Sort $B(\tau)$ in descending order by the bid price per container: $\bar{b}_i^p(\tau) = b_i^p(\tau)/\lambda_i^p(\tau)$;
Initially, set current buy price $b$ as $\bar{b}_i^r(\tau)$ as the first bid (highest price) in $B(\tau)$. number of trading containers $h = 0$, bid index $i = 1$;
**while** $b \geq Cost_d(\tau)$ **do**
    if $h + \lambda_i^p(\tau)$ is larger than the capacity of MDC, $\sum_m^{M_d} C_d^m$, or $i + 1$ is equal to the size of the number of buy bids: break;
    $b = b_i^p(\tau)$;
    $h+ = \lambda_i^p(\tau)$;
    $i++$;
**end**
$\rho = (b_{i+1}^p(\tau) + Cost_d(\tau))/2$ ;
**if** $b_i^p(\tau) \geq \rho \geq Cost_d(\tau)$ **then**
    All the first $i$ buyers win with price per container:;
    $\pi_s^d(\tau) = \pi_b^d(\tau) = \rho$;
**end**
**else**
    All the first $i - 1$ buyers win with buy price per container: $\pi_b^d(\tau) = b_i^d(\tau)$;
    The sell price per container is $\pi_s^d(\tau) = Cost_d(\tau)$;
**end**

The decision of the auction is indicated by a set of indicators, $X^r(\tau)$. The buying bid $b_i^p(\tau)$'s indicator is set to be $x_i^p(\tau) = 1$ if bid $b_i^p(\tau)$ wins.

The time complexity of the winner deciding algorithm can be shown as $O(|B(\tau)| \log |B(\tau)|)$ as the computation complexity is determined by the initial sorting of the bids, which is heavier than the computation deciding the winning bids which has the computation complexity $O(|B(\tau))$.

Next, we present the proposed contracts establishing algorithm based on the winning bids decision ( Algorithm 1). The basic idea behind the resource sharing auction framework is to maximize the utility for the MDCs as well as for the ECIPs and the SPs in a fair manner. As discussed earlier, the objective of SPs is to increase their profit by maximizing the utility of serving the customer with better service. The ECIPs want to provide more edge computing resources to the SPs to increase the revenue. The auction process for a given slot $\tau$ can be presented as three steps. In Step 1, The SPs estimate the workload for each position in every region $r$ to get the estimated workload, $\lambda_i^p(\tau)$. They send buy bids, $< b_i^p(\tau), \lambda_i^p(\tau) >$, where the bid price $b_i^p(\tau)$ is estimated from the utility it can gain from providing the service on the edge in region $r$, to the coordinator. In Step 2, the coordinator decides the winning bids using the Algorithm 1 for every region $r$. Each winner establishes a resource sharing contract with the owner of the MDC $d$. Finally in Step 3, if the auction is cleared without the MDC $d$ selling all the resources, the MDC $d$ will attend the next round of the auction. In the next round of the auction, if SP $i$ has the workload that is not satisfied, it sends the buy bids $< b_i^p(\tau), \lambda_i^p(\tau) >$ where $p \in r$ to randomly choose an adjacent region $r\prime$ of region $r$. For the MDCs which have remaining available resources, the process operates the next round of auction from *Step 1* until all the buy bids are satisfied or all the resources of MDCs are allocated.

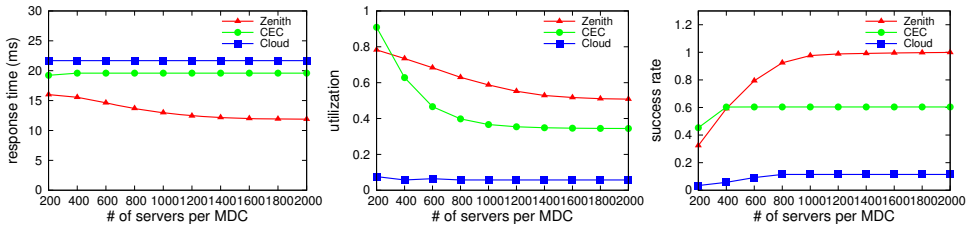
The above sequence of steps is operated for each time slot $\tau = 1$ to $\tau = T$ where $T$ is the maximum time slot for consideration. The result of the auction establishes the utility-maximizing contracts between the SPs and MDCs for the effective time slot from $\tau = 1$ to $\tau = T$.

After the previous steps, each SP has a set of contracts established with the ECIPs. The contracts are denoted by $Contract_i^r(\tau) = \{Contract_i^{d_1}(\tau), Contract_i^{d_2}(\tau), ...\}$ for serving the workload $\lambda_i^p(\tau) \in \lambda_i^r(\tau)$, where $d_1, d_2, ...$ are the index of the MDCs, $Contract_i^{d_1}(\tau) =< \pi_b^d(\tau), \pi_s^d(\tau), C_i^d(\tau) >$ is the contract which is established by SP $i$ with MDC $d$ for effective in time slot $\tau$, where $\pi_b^d(\tau)$ is the clear buying price for the contract, $\pi_b^d(\tau)$ is the clear selling price for the contract and $C_i^d(\tau)$ is the capacity of the resources in the contract.

### *C. Provisioning*

The contracts establishment algorithm solves the problem of allocating the resources for each SPs in a utility-maximizing manner. After the establishment of the contracts, the SPs hold resources which are densely geo-distributed at the edge of the network. The provisioning process is required for each SP to schedule the tasks to the containers on the MDCs to handle the requests of its services at the edge. The provisioning algorithm here decides how to place tasks on the containers in the MDCs of the established contracts in a contracts-aware manner.

In the provisioning process, the SP needs to react for the workload changes in a real-time manner. A reactive provisioning and task scheduling algorithm is needed to place the tasks on the application containers and decide the placement




(a) Response time (b) Utilization (c) Success Rate

Fig. 3. Impact of number of Servers per MDC

of the application containers to the right MDCs which can both optimize the cost of using the resource and the service performance that can potentially increase the application experience to the user. Our task scheduling algorithm aims at minimizing the network latency between the end nodes and the MDCs where the application container is hosted.

## V. Evaluation

We experimentally evaluate the effectiveness of *Zenith* in terms of job response times, success rate of meeting response time guarantees and resource utilization levels at the edge micro datacenters.

### A. Setup

The simulation uses a geographic map of 3000 miles *3000 miles size and randomly chooses locations from the map to place the MDCs. The WVD (Weighted Voronoi Diagram) is generated from the map with the locations of the MDCs as the sites. The latency used in the experiments is estimated using the distance-based model presented in [27]. This linear model estimates the latency based on the distance between the two points.

We consider that each region contains one micro data center and each MDC has 1000 servers in the default setting. The server has the same performance as that of the IBM server x3550 (2 x [Xeon X5675 3067 MHz, 6 cores], 16GB). Each server hosts up to 5 application containers at a given time. The location of the MDC is randomly chosen, and the timezone of the MDC is determined by the location. The geographic map area is divided into four time zones evenly to simulate the time-varying aspect of the dynamic electricity pricing. The electricity price is generated based on the hourly real-time electricity price from [28]. We use the distribution of the data in 2015 from NationalGrid's hourly electricity price to simulate the fluctuation of the real electricity market.

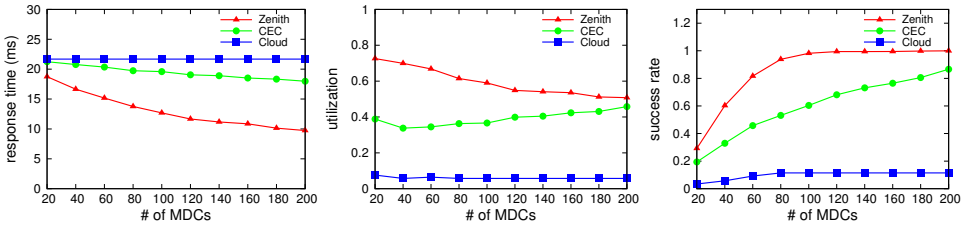
The default workload generates job requests of low-latency data processing tasks (of 100 bytes in size) and the container running at the MDC processes the request. The distribution of the workload is uniform throughout the map for the default setting. We consider that all workloads are response time sensitive, which means that if the response time exceeds the constraint, the task is considered to be a failed task. The default response time constraint is 30ms. We compare *Zenith* with two candidate mechanisms: (i) Coupled Edge Computing (CEC) mechanism, in which each MDC is owned by one of the SPs and the workload to the MDCs comes only from the SP that owns the MDC; (ii) conventional cloud-based (Cloud) solution in which the workload is processed at the large-scale datacenters placed on the left and right ends of the map.

### B. Experiment Results

To evaluate the performance efficiency of *Zenith*, we perform three sets of experiments: first, we study the impact of the number of servers in MDCs on the average response time of tasks, the average utilization at the MDCs and the overall success rate of the tasks. Second, we study the impact of the number of MDCs in the geographic map. Finally, we analyze the impact of different response time constraints on the perceived performance efficiency.

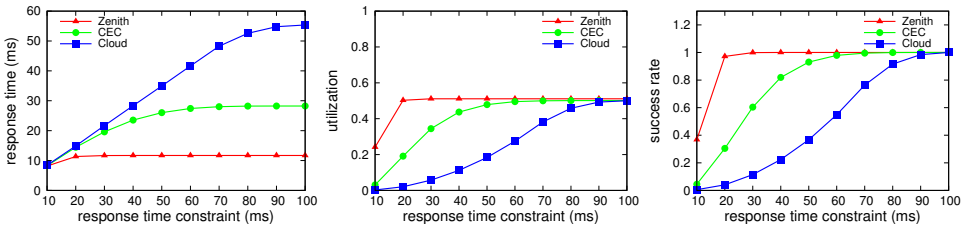
*1) Impact of No. of servers in MDCs:* In this experiment, we compare the performance of the mechanisms with different number of servers in each MDC. The number of servers per MDC is increased from 200 to 2000 in the evaluation. For the Cloud-based mechanism, the total number of servers present in the two large-scale datacenters are increased in such a way that they have the same number of total servers as the total number of servers in all MDCs. As shown in Figure 3a, the y-axis is the average response time of the tasks. The x-axis is the number of servers per MDC. We find that with the increased ability to share resources among MDCs, *Zenith* achieves the best result compared to CEC and Cloud mechanisms even when the number of servers is low. As shown in Figure 3b, *Zenith*, in general, can achieve higher utilization of the MDCs except when the resources are scarce such as when one MDC only handles 200 servers. In Figure 3c, we observe that the success rate of the tasks increases with increasing the number of servers. In addition, the success rates of CEC and Cloud schemes do not reach 100%. This is due to the fact that even when the resources are available, these schemes suffer from reduced proximity between the tasks and the MDCs assigned to them. Here, the response time constraints cannot be met with the nearest datacenters. From the above experiments, we can see that *Zenith* performs significantly better than CEC and Cloud mechanisms with respect to response time, resource utilization and success rate.

*2) Impact of No. of MDCs:* We next study the performance of *Zenith* with different number of MDCs present in the geographic map. The number of MDCs is increased from 20




(a) Response time (b) Utilization (c) Success Rate

Fig. 4. Impact of number of MDCs




(a) Response time (b) Utilization (c) Success Rate

Fig. 5. Impact of latency constraints

to 200. For the CEC mechanism, the MDCs are divided into 10 even groups such that each group is owned by one SP. As shown in Figure 4a, the x-axis is the number of MDCs. We find that the response time decreases from 20ms to about 10ms for *Zenith* as increasing the number of MDCs decrease the average distance between the endpoints and the MDCs which results in a decrease in response time. For the CEC mechanism also, the response time decreases from around 21ms to about 18ms. Here, we note that the Cloud-based mechanism is not influenced by the number of MDCs as the placement of the large-scale datacenters are fixed. In Figure 4b, we compare the resource utilization levels at the MDCs and we find that *Zenith* achieves higher utilization than CEC and Cloud mechanisms. As shown in Figure 4c, the success rate increases with increasing the number of MDCs. Here, *Zenith* performs significantly better than the two other mechanisms.

*3) Impact of Response Time Constraints:* Finally, we study the performance of *Zenith* to analyze the impact of different response time constraints. In this experiment, we increase the mean response constraint from 10ms to 100ms. As shown in Figure 5a, the x-axis is the response time constraint. The obtained response time increases for all the three mechanisms as increasing the response time constraint provides additional flexibility for task scheduling to satisfy more workloads with longer distance which results in an increase in the obtained response time. In Figure 5b, the utilization of all the three mechanisms are compared and we find that it increases with an easier response time constraint. For the Cloud mechanism, when the response time constraint increases significantly, it also achieves similar performance as *Zenith* and CEC. This is due to the fact that when the response time constraint is relaxed, a cloud solution allows the tasks to be transferred and executed in remotely located large-scale datacenters leading to a higher success rate. Figure 5c shows that the success rate increases with extending the limitations of the response time constraints. Here the Cloud solution also attains 100% success rate as *Zenith* and CEC. From the above experiments, we observe that *Zenith* performs significantly better than CEC and Cloud when the response time constraints are significant.

## VI. Related Work

Edge Computing has gained significant attention from the distributed systems community in the recent years. While the concept of edge and fog Computing is still in their early years of development, there has been several notable research efforts on this emerging topic. Bonomi et al. [21] discuss the concept of fog Computing and there has also been several other related developments in the broader area of edge and fog computing. Such efforts include the development of Cloudlets [4] proposed by Satyanarayanan et al. and the work on mobile edge computing [29] which is an extension of the effort on mobile cloud computing [30]. The primary benefit of edge computing comes from its ability to offer low latency computing resources on the fly for applications that have strict latency requirements. Edge computing is also beneficial in situations when a large number of small computing nodes need to deliver data to a cloud. Therefore applications such as IoT (Internet-of-Things), AR (Augmented reality) and VR (Virtual reality) benefit the most from modern edge computing solutions. There have been many research efforts studying the benefits of edge computing in these areas. Satyanarayanan et al. present GigaSight [31], an Internet-scale repository of crowd-sourced video content. Want et al. [32] discuss the technologies that enable IoT using edge computing. As the field is still emerging, there has been only few efforts addressing the problem of resource allocation in edge computing platforms. Aazam et al. [12] propose a model

for SPs to estimate the amount of services for each MDC in the edge computing platform. Do et al. [11] propose a system for allocating fog computing resources to minimize the carbon footprint. The solution is based on a distributed algorithm that employs the proximal algorithm and alternating direction method of multipliers(ADMM). All of the above solutions assume that the ECIs are tighlty coupled with the SPs and are controlled by the SPs which limits the resource sharing and latency benefits of the edge computing model. To the best of our knowledge, the work presented in this paper is the first research effort focusing on resource allocation for edge computing using a decoupled resource management model that manages the allocation of MDC resources independent of the service provisioning performed at the SPs.

## VII. Conclusions

In this paper, we propose *Zenith*, a resource allocation model for allocating computing resources in an edge computing platform. In contrast to conventional solutions, *Zenith* employs a new decoupled architecture in which the infrastructure management at the Edge Computing Infrastructures (ECIs) is performed independent of the service provisioning and service management performed by the service providers (SPs). Based on the proposed model, we present an auction-based mechanism for resource contract establishment and a latency-aware scheduling technique that maximizes the utility for both ECIPs and SPs. The proposed techniques are evaluated through extensive experiments that demonstrate the effectiveness, scalability and performance efficiency of the proposed model.

## References


[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Network*, vol. 31, no. 1, pp. 52–58, 2017.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.

[4] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE, 2014, pp. 1–9.

[5] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[6] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer, 2014.

[7] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE, 2016, pp. 1–8.

[8] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[9] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[10] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[11] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *Information Networking (ICOIN), 2015 International Conference on*. IEEE, 2015, pp. 324–329.

[12] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*. IEEE, 2015, pp. 687–694.

[13] S. K. Ong and A. Y. C. Nee, *Virtual and augmented reality applications in manufacturing*. Springer Science & Business Media, 2013.

[14] R. Kitchin, "The real-time city? big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, 2014.

[15] P. Siano, "Demand response and smart gridsa survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461–478, 2014.

[16] P. Kudtarkar, T. F. DeLuca, V. A. Fusaro, P. J. Tonellato, and D. P. Wall, "Cost-effective cloud computing: a case study using the comparative genomics tool, roundup," *Evolutionary Bioinformatics*, vol. 6, p. 197, 2010.

[17] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–12.

[18] M. Aazam, P. P. Hung, and E.-N. Huh, "Smart gateway based communication for cloud of things," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. IEEE, 2014, pp. 1–6.

[19] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures–a technology review," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, 2015, pp. 379–386.

[20] B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke, and O. H. Hoe, "Evaluation of docker as edge computing platform," in *Open Systems (ICOS), 2015 IEEE Confernece on*. IEEE, 2015, pp. 130–135.

[21] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[22] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European journal of operational research*, vol. 176, no. 2, pp. 657–690, 2007.

[23] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted voronoi diagrams and randomization to variance-based k-clustering," in *Proceedings of the tenth annual symposium on Computational geometry*. ACM, 1994, pp. 332–339.

[24] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, 2001, pp. 139–150.

[25] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[26] R. P. McAfee, "A dominant strategy double auction," *Journal of economic Theory*, vol. 56, no. 2, pp. 434–450, 1992.

[27] R. Goonatilake and R. A. Bachnak, "Modeling latency in a network distribution," *Network and Communication Technologies*, vol. 1, no. 2, p. 1, 2012.

[28] N. Grid, "Large general tou," https://www.nationalgridus.com/niagaramohawk /business/rates/5_hour_charge.asp, accessed Jan. 4, 2016.

[29] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[30] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[31] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

[32] R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," *Computer*, vol. 48, no. 1, pp. 28–35, 2015.