# Accepted Manuscript

QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds

Ziyan Wu, Zhihui Lu, Patrick C.K. Hung, Shih-Chia Huang, Yu Tong, Zhenfang Wang

PII: S0167-739X(18)31754-0
DOI: https://doi.org/10.1016/j.future.2018.09.032
Reference: FUTURE 4467

To appear in: *Future Generation Computer Systems*

Received date : 24 July 2018
Revised date : 27 August 2018
Accepted date : 9 September 2018

Please cite this article as: Z. Wu, et al., QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.09.032

# QaMeC: A QoS-driven IoVs Application Optimizing Deployment Scheme in Multimedia Edge Clouds

Ziyan Wu[1], Zhihui Lu*[1], Patrick C. K. Hung[2], Shih-Chia Huang [3], Yu Tong[4], Zhenrai Wang[4]
[1,4]ziyanwu16,lzh, 17210240204, 13210240114 @fudan.edu.cn [1], [2] patrick.hung@uoit.ca schuang@ntut.edu.tw
[1]School of Computer Science, Fudan University, [1]Shanghai, China
[2] Faculty of Business and IT, University of Ontario Institute of Technology, Canada
[3]Department of Electronic Engineering, National Taipei University of Technology, Taiwan
[4]Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai,China

***Abstract*—Deploying applications to a centralized cloud for service delivery is infeasible because of the excessive latency and bandwidth limitation of the Internet, such as transporting all IoVs data to big data processing service in a centralized cloud. Therefore, multi-clouds, especially multiple edge clouds is a rising trend for cloud service provision. However, heterogeneity of the cloud service, complex deployment requirements, and large problem space of multi-clouds deployment make how to deploy applications in the multi-clouds environment be a difficult and error-prone decision-making process. Due to these difficulties, current SLA-based solution lacks a unified model to represent functional and non-functional requirements of users. In this background, we propose a QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds (QaMeC). Our scheme builds a unified QoS model to shield off the inconsistency of QoS calculation. Moreover, we use NSGA-II algorithm as the solution to the multi-clouds application deployment problem. The implementation and experiments show that our QaMeC scheme can provide optimal and efficient service deployment solutions for a variety of applications with different QoS requirements in CDN multimedia edge cloud environment.**

***Keywords—IoVs (Internet of Vehicles); IoT(Internet of Things); Optimizing Deployment; Cloud Computing; Edge Computing; Multi-clouds; QoS; CDN***

## I. INTRODUCTION

The new era of the Internet of Things is driving the evolution of conventional Vehicle Ad-hoc Networks into the Internet of Vehicles (IoVs). With the rapid development of computation and communication technologies, IoV promises huge commercial interest and research value, thereby attracting a large number of companies and researchers [1]. IoVs is expected to analyze and utilize the various information, especially multimedia inside and outside vehicles itself through wireless communication techniques. Currently, deploying applications to a centralized cloud for service delivery is infeasible because of the excessive latency and bandwidth limitation of the Internet, especially it is difficult to move all IoVs data to the centralized cloud for IoVs application. A promising approach to addressing the challenges for application deployment is "edge cloud" that pushes various computing and storage capabilities to multiple edge clouds. The edge cloud refers to building open cloud infrastructure in the network edge close to the clients or data source side. It offers network, computing and storage resources. It provides intelligent edge services to meet the critical needs of the digital industry, including IoT data localized analysis, agile connection, real-time traffic, data optimization, nearest calculation etc.

If customers use edge clouds, they usually use distributed multi-clouds architecture. Multi-clouds has become a hot topic in the past several years. In most cases, multiple types and brands of cloud deployment are not only reasonable but also able to offer better value than single cloud deployment. In the industry, more and more companies are implementing multiple cloud computing platform development strategies to avoid being limited to a single supplier, to enhance available service deliverability, to avoid arbitrage or maintain specific control over sensitive information. In one scenario, a user may choose Amazon Web services (AWS), simple storage service (S3) as storage, Rackspace OnMetal for cloud database, Google for data systems, and a private cloud based on OpenStack to manage sensitive data and applications. All these resources work together to establish one or more systems, allowing companies to meet specific needs.

The cloud market is complicated due to complex deployment requirements, as well as a variety of resource specifications. It is a tricky decision process and error-prone for the users to choose the optimal deployment with their own requirements, especially for multi-clouds deployment. There is no such a corresponding service mechanism in the cloud service system which helps users to decide how to select the fittest services for their own applications. It is hard for the users to evaluate SLA, which is a collection of contents and provisions. If there is a broker mechanism between users and cloud service providers, it will solve the problem. The mechanism can not only provide the most economical, suitable cloud services for users but also reduce resource fragments of cloud service providers, increasing resource utilization. This can lead to reaching more SLAs. Meanwhile, the introduction of such a mechanism further improves the service system, increasing the level of cloud services. The main reason why the broker mechanism doesn't exist is that the current multi-clouds systems lack a unified description of the QoS level - both functional and non-functional requirements for the users.

Edge computing allows data computing, storage, and service supply to be moved from central cloud to the local edge devices such as smartphones, smart gateways or routers and local PCs or micro-datacenter. Thus, edge computing supports IoT big data localized processing regarding high scalability, low delay, location awareness, and allowing of using local computing capabilities in real time. CDN (content delivery network or content distribution network) is a typical representative of edge computing, and CDN serves people mainly through accelerating distribution of pictures, video, and dynamic content to the edge end user. Now, with the

development of the Internet of things, a large number of things are deployed on the more edge of the network, and their uplink and downlink data need to be speeded up, such as sensors and IoVs. Akamai, the largest provider of CDN, has begun to propose a CDN solution for IoT. CDN's network needs to sink further to speed up the acceleration of IoT and IoVs[34]. In this paper, we present QaMeC: a QoS-driven IoVs application deployment scheme in multimedia edge clouds based on CDN. The broker system, proposed in this paper, builds a unified QoS model to shield off the inconsistency of QoS calculation process. Since QoS specification contains various functional and non-functional complex metrics, users have various requirements. The broker system can integrate multiple cloud providers' solutions for users to make optimal decisions to satisfy their requirements. Moreover, this can solve the deployment problem greatly for non-expert users who are not familiar with cloud computing.

Our main contributions in this paper are listed below:

- We propose a novel QoS-driven IoVs application service optimizing deployment scheme in CDN multimedia edge clouds environment (QaMeC).
- The proposed service demand model and QoS model can provide a complete description of user requirements. It gives a quantitative description of the service request and delivery. The QoS data is retrieved from the PoPs log data of the real CDN operator network, which ensures the objectivity for the users, overcoming the drawbacks of the SLA-based solution.
- The designed NSGA-II algorithm is applied to search for the best deployment plan for the users, to reduce the vast problem space of the combinatorial optimizing decision-making problem.

We organized the remainder of this paper is as follows.

In Section I, we introduce the background of our work and explain why our work is valuable. In Section II, we review the related works. In Section III, we present QaMeC architecture: QoS-driven IoVs application deployment scheme in multimedia edge clouds environment. In Section IV, we design the QoS models to help users to deploy service in multiple edge clouds environment. In section V, we define the problems and design the related algorithms. In section VI, we evaluate our QaMeC application deployment scheme on a real multiple edge clouds environment, including OpenStack and CDN. Finally, in Section VII we conclude our works and discuss possible future work.

## II. Related Work

The consumption of resources and services from multiple clouds or edge-clouds for reasons like high availability, cost reductions or special features is a natural evolution from in-silo clouds. Several middlewares are already available for multiple Clouds. However, due to the complexity of the technical solutions, their approaches are quite different and a classification is needed to guide the potential users. The paper [1] looks to the reports on multiple cloud topics and proposes a specific taxonomy. It identifies the ready-to-use software and services and classifies them according the taxonomy. The famous network industry top conference Infocom2014 first set up a Cross-cloud Workshop, which gathered some of the experts and scholars to discuss the rise of cross-cloud technology, specifically pointed out some new challenges and problems in this field. Dana Petcu [2] et al. summarizes the present requirements and main technical challenges of multi-clouds architecture, and the current multi-clouds architecture development tools, such as JCloud, LibCloud, Delta -Cloud, and the paper describes that the cloud service quality guarantee is currently the controversial topics in multi-clouds scenarios, and suggests a model-driven method as a feasible solution for complicated tasks under the multi-clouds architecture. Our paper also proposes a QoS-driven application optimizing deployment scheme in multi-clouds environments. Felix Cuadrado et al. [4] illustrate the major challenges of fully implementing the multi-clouds architecture application. Now most cloud providers offer heterogeneous API. A cross-cloud infrastructure is designed as the federation of multiple cloud datacenters, offered by potentially multiple providers, with homogeneous APIs for acquiring virtual resources on demand. This model benefits application providers, which expect to reduce cost and avoid vendor lock-in. In [5], Wagle devised a broker layer to consider both SLA commitment and service delivery for cloud services recommendation. There is a survey by Bandar [6] which compares the existing brokers and highlights the key features that must be available regarding optimizing QoS and SLA. Our paper uses the Broker approach to provide a unified QoS representation to connect multiple heterogeneous clouds. Hitoshi Yabusaki [7] points out federating various clouds enables to utilize datacenters in various geographical regions regardless of their services. The response time can be reduced by replicating the applications and related data at datacenters near the terminals by considering the factors of delay (e.g., data synchronization, distribution of multi-tier applications, and influence of other applications). They design the mechanism of wide area tentative scaling (WATS) to improve the response time in a phased manner by repetitively replicate a part of the application and related data at other datacenters and selecting a better organization. Evaluation results showed that WATS successfully decreased the response time in a phased manner. Indeed, reducing the response time of multi-clouds applications is a common concern for developers and users. In ACM Computing Surveys (CSUR) 2014, Adel and Buyya published papers [8] "Interconnected Cloud Computing Environments: Challenges Taxonomy", made a comprehensive summary and survey on the interconnected cloud technology. This survey initially discusses all the relevant aspects motivating cloud interoperability. Furthermore, it classifies possible cloud interoperability scenarios and architectures. The spectrum of challenges and obstacles that the inter-cloud realization is faced with are covered, including resource supply provisioning, mobile portability, SLA, security, monitoring, economy, network, and autonomics. In our paper, we mainly focus on the SLA issue between cloud providers and application service providers. This paper [9] presents the MUSA deployer models, which help developers to express their security requirements, and a deployer tool that automatically provides cloud security services to offer Security SLAs. Duplyakin et al. [10] present an environment that is in charge of multi-clouds deployment rebalancing by terminating instances, in lower-preferred clouds and launching replacement instances in higher-preferred clouds

to satisfy user preferences. They consider three rebalancing policies: 1) only idle excess instances are terminated, 2) excess instances are terminated gracefully, and 3) worker instances are aggressively terminated, even if they are running user jobs. To verify the effectiveness of their rebalancing strategy, they evaluate these policies in a master worker environment deployed across multiple NSF FutureGrid clouds and test the ability of the policies to rebalance multi-clouds deployments appropriately, and analyze trade-offs. Castillo et al. [11] carries out the integration of OpenStack-based platforms into larger, heterogeneous multi-clouds infrastructures, taking the EU FP7 BonFIRE project as an integration use case. Ultimately, they aim to contribute to the state of the art and provide guidelines to integrators trying to federate Open Stack testbeds into more complex architectures. Wu, Zhe et al.[12] try to recognize the opportunity for aggressively minimizing user-perceived latencies by deploying web services across multiple cloud services. With the aid of measurements over 5 weeks from 265 PlanetLab sites to three popular cloud services, they demonstrated that web services that span multiple cloud services can reduce latencies by over 20% for users in up to 50% of prefixes. Furthermore, they showed that users in several regions will experience high latencies even if web services take advantage of multiple cloud services, and that multi-clouds deployments will necessarily have to replicate data to optimize user-perceived latencies. Our paper also focuses on optimizing the selection of cloud providers to reduce the latency for users to visit application in a multi-clouds environment. Although service allocation based on SLA has been well investigated in cloud computing so far, the new upcoming issues regarding to utilize multiple clouds has led new challenges. Therefore, the paper [13] deploys and manages distributed cloud applications through the combination of TOSCA and CAMP. In [14], Alshammari et al. point out the advantages of data recovery in cost and reliability in the multi clouds environment. Farokhi et al.[15] looks at the service selection and allocation in a multi-clouds, as a delivery model of multiple clouds, from the perspective of SaaS provider. The designed framework assists SaaS providers to find suitable multi-clouds infrastructure services which best satisfy their requirements while handling SLA issues. They present an overview of the complete system and describe how the services are selected and the corresponding SLAs are monitored to detect the SLA violations. The orchestration of application components across heterogeneous cloud providers is a complicated process. In [16], Jie Yang et al. propose a resource allocation policy that maintains the highest level of security using the genetic algorithm. The hybrid cloud(private cloud plus public clouds) is a major form of multi-cloud. In the paper[17], to cost-effectively withstand flash crowds with soft guarantee, Niu Yipei. et al. propose a solution that makes intelligent and efficient decisions on scheduling requests in the hybrid cloud and adjusting the capacity of the public cloud. In the paper[18], Niu Yipei et al. utilize the queueing theory to evaluate the average response time and explore the tradeoff between performance and cost in the hybrid cloud. By taking advantage of Lyapunov optimization techniques, they design an online decision algorithm for request distribution which achieves the average response time arbitrarily close to the theoretically optimum and controls the outsourcing cost based on a given budge. The simulation results demonstrate in a hybrid cloud, their method can reduce cost of e-commerce services as well as guarantee performance when encountering flash crowds. In [19], liu fangming et al. propose a cost-effective service for hybrid cloud applications, which selects the best public cloud for out-sourcing and adapts cloud price changes dynamically, along with provisioning global load balancing. The system uses a two-tier load balancing mechanism, provisioning virtual machine (VM) and cloud level load balancing. Existing multi-cloud solutions cannot well address the performance issue, their networking performance is degraded by the slower cloud, Peng haowen et al.[20] provide affirmative answers through the design and implementation of UniDrive, a CCS app that synergizes multiple CCSs into a multi-cloud with better sync performance, reliability, and security.

At the same time, the centralized cloud architecture needs to be marginalized and decentralized. Therefore, multiple edge clouds architectures are emerging. In **Error! Reference source not found.**, the authors presented Nebula: a distributed cloud infrastructure that uses voluntary edge resources for both computing and data storage. They described the lightweight Nebula architecture that enables distributed data-intensive computing through some optimizations, including location-aware data and computation placement, replication, and recovery. The authors verified Nebula's performance on an emulated volunteer platform that spanned over 50 PlanetLab nodes distributed across Europe and showed how MapReduce can be deployed and run on Nebula, as the standard data-intensive framework. They verified that Nebula MapReduce is robust for a wide array of failures and substantially outperforms other wide-area versions based on a BOINC-like model.

Currently, the centralized cloud is facing increasing difficulty to handle the IoT (including IoVs) big data while moving all IoT data to the cloud. Edge computing allows data computing, storage, and service supply to be moved from central cloud to the local edge devices such as smartphones, smart gateways or routers and local PCs or micro-datacenter. Thus, edge computing supports IoT big data localized processing regarding high scalability, low delay, location awareness, and allowing of using local computing capabilities in real time. Ola Salman et al. [22] proposed that the primary objective of *Mobile Edge Computing* (MEC) solution was export of central cloud capabilities to user's proximity for decreasing latency, augmenting available bandwidth and decreasing traffic load on the core network.

Weisong et al.[23] described that the success of IoT and rich cloud services have helped to create the need for edge computing, in which data processing occurs in part at the network edge, rather than completely in the centralized cloud. Edge computing could address concerns such as latency, mobile devices' limited battery life, bandwidth costs, security, and privacy. Stream Processing Frameworks (SPF, e.g., Apache Storm) often failed in addressing certain requirements of IoT systems. Apostolos et al.[24] described topology-aware SPF extensions, which can eliminate latency requirement violations and reduce cloud-to-edge bandwidth consumption to 1/3 comparing to Apache Storm.

Service deployment in multiple edge clouds must contain service composition. Composite services typically involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises to complete a multi-step business interaction. Compared to single cloud service, composite cloud service, which integrates multiple cloud services, can offer more value. While researching composite cloud service is in its early stage, there is related research in Service Level Agreement (SLA) based web service selection in cloud environment [5] [25] . These research propose many methods of selecting the appropriate composition of services, many of which use AI planning algorithm. However, these existing methods only consider the functional requirement, neglecting non-functional requirements. There are also some preliminary results concerning QoS based service selection. The paper [26] is the review of related research in selection methods. In the paper [27], we propose a CDN multi-clouds resource allocation scheme based on real CDN log data analysis on Spark. We firstly design a QoS model and run long-term deployment algorithm to deploy resources at the minimum cost while keeping good QoS. Secondly, we make predictions on requests and allocate resource by prediction result in short term. Thirdly, we run the extended algorithm to handle inaccurate prediction when the number of requests is high and use pre-copying algorithm to decide which content to deploy in the new VMs. From the evaluation result, long-term deployment algorithm can reduce the cost with the same QoS, compared with the actual situation and the prediction is accurate in most time and the extension algorithm can make up for the inaccuracy and responds timely. In paper [28], we propose and evaluate IoTDeM, which is an extended IoT b… data-oriented model for predicting MapReduce performance in… multiple edge clouds. IoTDeM can predict MapReduce jobs' total execution time in a general implementation scenario with varying reduce amounts and cluster scales in Hadoop 2.0. Through choosing more representative features to represent a job, the IoTDeM model selects a cluster scale as a crucial parameter to further extend LWLR model. The experiments show IoTDeM can effectively predict the total execution time of MapReduce applications with the average relative error of less than 10% in Hadoop 2, rather than Hadoop1. In paper[29], chen min et al. propose an innovative paradigm called Cognitive Internet of Vehicles (CIoV) … enhance transportation safety and network security by mining effective information from both physical and network data space. They focus on crucial cognitive design issues from three perspectives, namely, intra-vehicle network, inter-vehicle network and beyond-vehicle network. Simulations are then conducted to prove the effect of CIoV. In paper[30], chen min et al. propose a new concept of computing task caching and design the optimal computing task caching policy. Furthermore, joint optimization of computation, caching, and communication on the edge cloud, dubbed Edge-CoCaCo, is proposed based on an alternating iterative algorithm. In paper[31], chen min et al. design an innovative framework of task offloading for mobile edge computing in Software-Defined Ultra-Dense Network. By deploying controller at macro cell BS, the global information about mobile devices, base stations, edge cloud and tasks can be obtained, and thus enabling the optimal task offloading of mobile devices

## III. QAMEC ARCHITECTURE

There are many kinds of software that supports application and service automation deployment over multiple cloud services. However, most of them only provide a management interface, without giving optimizing deployment plans.

In smart city IoVs applications, vehicles connected in the IoT have many challenges in data collection, transmission and processing. Here, we propose QaMeC-the QoS-driven IoVs application deployment scheme in multimedia edge clouds environment to help users make optimized deployment decisions. In this paper, we mainly focus on the IoVs application deployed on CDN multiple multimedia edge clouds. CDN is a typical edge cloud. A CDN is a geographically distributed network of proxy servers and their data centers. The nodes of the CDN include the upper layer backbone and the lower layer edge nodes. The edge nodes of CDN are called PoP (point of presence). Therefore, the target scenario is that one IoVs service provider will deploy their service on CDN multiple PoP edge clouds.

Figure1 describes the total QaMeC architecture, including three layers in the system.

The first layer is the Client Layer. User interface makes it convenient for the users to propose their service deployment requirements for hardware, software, QoS etc.

The second layer is Broker layer. This layer has several components. This part is the key to carry out multi-clouds IoVs application deployment driven by QoS.

In the Broker layer, service demand model represents user's service deployment requests. We map user's service deployment requirement to two-dimensional vector matrix, one vector is the identified serial number of PoP, the other vector is time T. Each element of the matrix represents the amount of visiting a PoP point at a certain time T.

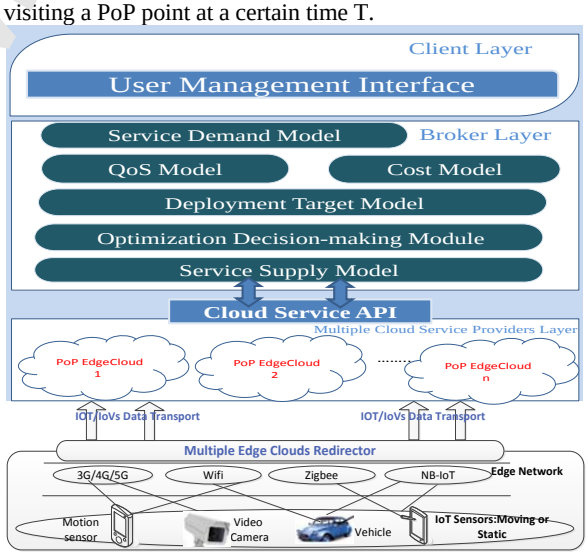



Figure 1. QoS-driven IoVs Application deployment Architecture in multimedia edge clouds environment

QoS Model is responsible for quantifying QoS, whose evaluation data is from the monitors located in every PoP edge cloud. Deployment target model defines specific service configuration goals for users, such as the deployment target for

CPU-intensive services or I/O-intensive services. Cost model will calculate users' consumption of multiple edge cloud resources. Service supply model will provide the comprehensive supply amounts of multiple edge cloud services. Optimization decision-making module is the core module in this layer. This module will choose the appropriate algorithms to implement optimal service selection. In order to implement QoS-driven application optimization deployment, we need to take service supply model and cost model as the input of the decision-making algorithm with the constraint of QoS model and deployment target model.

The third layer is multiple cloud service provider layer. The cloud service providers expose their cloud service API to the broker. The broker will access to cloud service API to deploy applications in multiple PoP edge cloud providers. Every PoP edge cloud is the basic unit of CDN services. These PoPs can serve the data transmission of IoT or IoVs nodes, as shown in Figures1.

The flowchart of our QaMeC scheme is described in Fig. 2. First, a user (such as an IoVs application service provider) will put forward their requirements based on their service attributes. In addition, the user will put forward service quality (QoS) level they care about mostly such as budgets, service responsiveness, and service availability and so on.

Next, the system will model user's request using service demand model. Then, based on the specific QoS requirements put forward by users, the system will call service demand model and deployment target model, and then call QoS model and Cost model to build the integrated model and provide the input for the following optimizing deployment decision-making module.

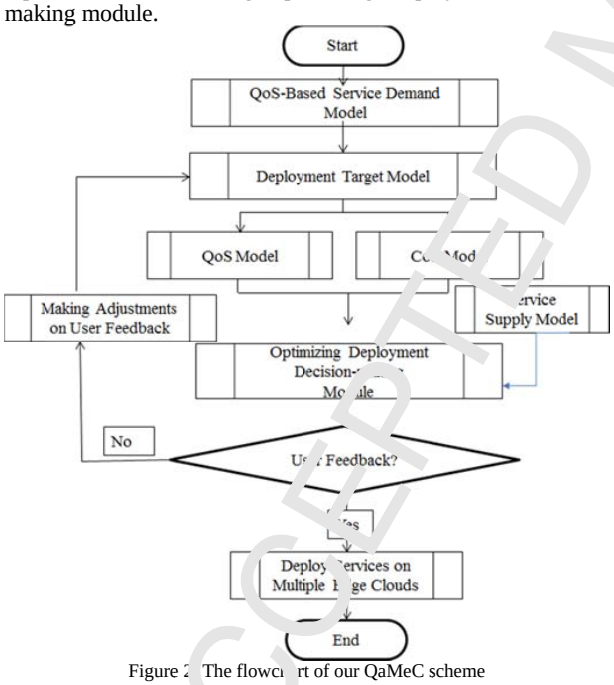



Figure 2. The flowchart of our QaMeC scheme

After the above procedures, the system will call its core module, which is optimization decision-making module. In this part, the system will first find the candidate solutions based on QoS model and cost model, referring to the deployment target model and the service supply model. Then the decision-making algorithm will find the optimal application deployment plan. The plan will be shown to the user for further adjustment. After confirmation, the applications will be deployed on multiple edge clouds based on the decided plan.

## IV. QOS-DRIVEN MULTIMEDIA EDGE CLOUDS IOVS APPLICATION DEPLOYMENT MODELS

The first step of multimedia edge clouds IoVs application deployment is to model cloud resources and QoS requirements. The next step is to select the most suitable cloud service, minimizing deployment and service cost and maximizing QoS. It's a multi-objective optimization problem. Therefore, we first construct QoS metrics model, deployment target model, service demand model, service supply model, service capacity model, unit service price model. CDN is our utilized multimedia edge clouds for IoVs communication. The CDN goal is to distribute service spatially relative to end-users to provide high availability and high performance. Therefore, the functional requirements of CDNs is to speed up a large portion of the Internet content distribution to the clients, including web objects (text, graphics and scripts), downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks. The following QoS metrics model are related with the non-functional requirements of CDN edge clouds, such as Throughput, Responsiveness. Since a service provider's application will be located in a PoP or multiple PoPs. In Table I, we summarize the following notations before designing the QoS model:

TABLE I. SUMMARY OF NOTATIONS

| Notation | Description |
|---|---|
| $Q$ | QoS vector |
| $G$ | Graph of service providers |
| V | Set of PoPs |
| E | The links between PoPs |
| D | Service Demand Model |
| $d_i(t)$ | The traffic demands of PoP I at t |
| $s_{i,j}(t)$ | fraction of service demands at PoP i supplied by PoP j at time t |
| C | Service capacity vectors |
| $C_i$ | $c_i$ denotes the limit capacity of a PoP |
| $Cost_{overall}$ | The overall cost of a given deployment plan |
| B | Budget of the deployment |

### A. QoS Model

For every deployment plan, the system will maximize the QoS of deployed service. The QoS performance is denoted as $Q = (Q_1, Q_2, \dots, Q_p)$, where $Q_u$ can be any QoS metrics that can characterize some aspects of the service's QoS.

In economics, utility function can be used to represent preference. It measures consumer satisfaction. In this paper, we use utility function to measure user satisfaction towards cloud services. We use exponential function to emulate user preference towards a particular QoS standard. Every sub-goal is represented by $u_i(Q_i)$ . The function value is the corresponding utility coefficient. $q_i(x)$ is the value of QoS metrics model of corresponding service item. The utility coefficient is between 0 and 1. When $q_i(x)$ implies satisfaction , $u_i = 1$ . When $q_i(x)$ implies non-satisfaction , $u_i = 0$ . Users may have different psychological preference towards different service objects. $\alpha_n$ is the psychological preference that is set by the users. By linearly map the QoS vector to a utility, we can model the satisfiability for the QoS.

$$\mathrm{U} = \sum_{n=1}^{p} \alpha_n u_i(Q_i) \quad (1)$$

At the same time, users should input their acceptable range of each metric, which can be denoted by $Q_i^*$.

### *B. Model of Deployment Target*

We use a weighted directed graph G(V, E) to represents the infrastructure we want to deploy on, where V is the set of vertices denoting the PoPs. $|\mathrm{V}| = \mathrm{k}$. E is the set of edges denoting links between PoPs. The weight is a QoS vector between two PoPs. We define these concepts as following:

$$G = \{\mathrm{V}, \mathrm{E}\}$$
$$V = \{v_i \mid 0 \le i \le k\}$$
$$\mathrm{E} = \{(v_i, v_j) \mid 0 \le \mathrm{i}, \mathrm{j} \le \mathrm{k}\} \quad (2)$$

For every $(v_p, v_q) \in \mathrm{E}$, we assign a QoS vector that characterize the service quality that the users of $v_p$ can expect of the service that $v_q$ can get provide. The measurement is done between every pair of PoPs, so G is a full connected graph.

### *C. Model of Service Demand*

When deploying an application, we should consider how much amount the service is demanded

Consider a time period of $\mathrm{T} = \{1, 2, \cdots, n\}$. We denote the traffic demands of PoP i at time t using the following set:

$$D = \{d_i(t) \mid 0 \le i \le k, \forall t \in T\} \quad (3)$$

### *D.* Model of Service Supply

When the users request service, the demand will send to the PoPs according to a service plan. For instance, 80% of the service demand of the users of Shanghai will served by the PoP located in Shanghai. The rest of the service demand will be scheduled to other area like Jiangsu. While a specific user's request can be scheduled dynamically based on real-time infrastructure status, there is a long-term plan how it can be served in general.

Next, we define a three-dimensional variable named traffic supply fraction as:

$$S = \{s_{i,j}(t), 0 \le i, j \le k, \forall t \in T\} \quad (4)$$

Where $s_{i,j}(t)$ denotes the fraction of service demands at PoP i supplied by PoP j at time t. Each PoP can supply service to itself.

From the definition of s, we can derive a constraint:

$$\sum_{i=1}^{k} s_{i,k}(t) = 1 \quad (5)$$

### *E. Model of Service Capacity*

When deploying applications, we should consider the maximum capacity that a specific PoP can provide. The amount of resources we allocate for an application should not exceed its maximum capacity.

We use a capacity vector to represent the of service capacity of Service Providers as:

$$\mathrm{C} = [c_1, c_2, \cdots, c_k] \quad (6)$$

in which $c_i$ denotes the limit capacity of a given PoP. c is an abstraction determined by the servers and network equipment that the service providers can provide.

At any time, the service deployed at a PoP cannot exceed its maximum capacity:

$$\forall i, k, \; d_i(t)\, s_{i,k}(t) \le c_k \quad (7)$$

### *F. Model of Unit Service Price*

When allocating the resource, we should consider unit service price of a cloud provider that can have. The unit service price comprised of leasing of cpu, memory and bandwidth. The cost model can vary depending on the application type. There are many factors that will influence the cost of a PoP in different places. We abstract these details by assigning a unit service price $U_i$ to every PoP $V_i$. We estimate the cost of *a service* provided by a PoP by:

$$Cost_i = \sum_{t=1}^{T} s_{i,k}(t)\, d_i\, U_i \quad (8)$$

From the definition above, we can calculate the overall cost: $Cost_{overall} = \sum_{t=1}^{T} \sum_{1}^{k} C_i$ (9)
which should be controlled under budget B.

## V. QAMEC PROBLEM FORMULATION AND ALGORITHM DESIGN

Given the model defined in the previous section, we can formulate our problem as a multi-objective optimization problem. A multi-objective optimization problem can be formalized as following:

$$Min(Max)\, f(x) = (f_1(x), f_2(x), \dots, f_k(x))^T \quad (10)$$
$$s.t \; x \in \Omega$$

$\Omega$ is the feasible solutions set that is nonempty, $f(x)$ is a vector-valued function, k is the number of objectives.

Our goal is to select and allocate resources on the set of PoPs to optimize the QoS, which can be formulized as following:

Max U
Min Cost

$$s.t \sum_{n=1}^{k} s_{i,k}(t) = 1$$
$$C_{overall} \le B$$
$$\forall i, t, k,\ d_i(t)\, s_{i,k}(t) \le c_k$$
$$Q_i^* \ better\ than\ Q_i$$ ($Q_i^*$ represents the user specified QoS) (11)

Our optimization process is to find the best service supply plan which is characterized by the service supply model of S. The size of S is determined by the number of PoPs. The PoPs number is over 300, so we have a large problem space, which means that using exhaustive search to find the Pareto solution isn't practical. Here we use evolution algorithm to find the approximate Pareto solution in a reasonable time. Evolution algorithms can give near optimal solution in a reduced processing time. The Evolutionary Algorithms (EA) are one of the most known bio-inspired algorithms which can deal with NP-hard problems. The EA are based on the natural evolution theory. The main idea is that the adapted species apparition is a consequence of two principal phenomena: (1) The natural selection (the most adapted individuals will survive and reproduce), (2) numerous variation can happen on the genetic material of species [32].

The NSGA-II [33] is a Pareto based multi-objective EA. This genetic algorithm is a fast elitist approach, without parameters, which manipulates a population of solutions, using an explicit mechanism to preserve diversity. At the beginning, we generate N individuals by assigning the service supply to the nearest PoPs, which forms the initial population $P_0$. We sort population according to Pareto dominance. In a naive approach, in order to identify solutions of the first nondominated front in a population of size $N_i$. The non-dominated individuals are assigned to the first front ($F_1$) and have a rank 1. Then, others fronts are assigned recursively with ignoring individuals which have been assigned previously. By fast-Non-Dominated-Sorting (Algorithm 1 [33] ), the time complexity has reduced to $O(MN^2)$.

Along with convergence to the Pareto-optimal set, a good spread of solutions in the obtained set of solutions is required. The basic idea of crowding distance assignment is to address the drawback of assigning the parameters of sharing function manually. The crowding-distance computation procedure of all solutions in a nondominated set I is listed in Algorithm 2 [33].

We define an individual as a real value that concatenated together in the model of supply S that represents a service supply plan which is constrained by Formula (5) and (7). A population is a set of individuals that represent a solution. Every iteration of the algorithm is called generation. At the beginning. Individuals of the N iteration are called parents and those of the (N+1) $^{th}$ generation are called children. The mainloop of NSGA-II is explained in Algorithm 3 [33].

**QaMec-invoked Algorithm 1.** ***fast-Non-Dominated-Sorting***

**Input: P, candidate solutions**
**for each p ∈ P:**
  $S_p \leftarrow \emptyset$
  $n_p \leftarrow 0$
  **for each q ∈ Q:**
    **if p dominates q then**
      $S_p = S_p \cup \{q\}$
    **else:**
      $n_p \leftarrow n_p + 1$
    **if** $n_p = 0$ ***than***:
      $p_{rank} = 1$
      $F_1 = F_1 \cup \{p\}$
**i=1**
**while** $F_i \neq \emptyset$:
  $Q \leftarrow \emptyset$
  **for each p ∈** $F_i$:
    **for each q ∈** $S_p$:
      $n_q \leftarrow n_q - 1$
      **if** $n_q = 0$ ***then***
        $q_{rank} \leftarrow i + 1$
        $Q \leftarrow Q \cup \{q\}$
**i ←** $i + 1$
$F_i \leftarrow Q$

**QaMec-invoked Algorithm 2.** Crowding-distance-assignment

**Input: I , candidate solutions**
**I = |I|**
**for each i:**
  **I[i]distance← 0**
**For each objective m:**
  **I = sort(I,m)**
  $I[1]_{distance} = I[l]_{distance} = infinite$
  **For i = 2 to (l − 1):**
    $I[l]_{distance} = I[l]_{distance} + (I[i+1] \cdot m - I[i-1] \cdot m)/(f_m^{max} - f_m^{min})$

**QaMec-invoked Algorithm 3.** NSGA-II

**t ←0**
$P_0 \leftarrow$ *InitialPopulation*
***while t< MaxGen do***
  $R_t \leftarrow P_t \cup Q_t$
  $F \leftarrow$ *fast-Non-Dominated-Sorting*$(R_t)$
  *Creation of an empty population* $P_{t+1}$
  **i ← 0**
  ***while*** $|P_{t+1}| + |F_i| \le N$ ***do***
    $|P_{t+1}| \leftarrow |P_{t+1}| \cup F_i$
    *Crowding – Distance-Assignment*$(F_i)$
    $i \leftarrow$ **i + 1**
  ***end while***
  *sort*$(F_i)$
  $P_{t+1} \leftarrow P_{t+1} \cup F_i[1: N - |P_{t+1}|]$
  $Q_{t+1} \leftarrow$ *Generate new population with genetic operators from* $P_{t+1}$
  $t \leftarrow t + 1$
**end while**

## VI. QAMEC EXPERIMENTS AND ANALYSIS

In order to verify our proposed QaMeC: QoS-driven IoVs application optimizing deployment scheme in multimedia CDN edge clouds, we implement the main architecture, model and related algorithms, and carry out corresponding verification experiments. Our experimental environment is composed of two parts. One is a small scale OpenStack multi-cloud environment built by our lab. One is a larger scale CDN multi-cloud environment and data provided by the largest CDN operator in China.

### *A.* **Multi-cloud deployment of Web Servers**

To simulate the diversity of real cloud service market, we build three sets of OpenStack edge cloud environment with different service prices and configurations, according to the current market situation, as Figure 6 shown. These three OpenStack edge clouds are located three campuses of our university, with 1Gbps connection. We captured the monitoring data from OpenStack ceilometer. The Ceilometer project is a data collection service that provides the ability to normalize and transform data across all current OpenStack core components with work underway to support future OpenStack components.

Ceilometer is a component of the Telemetry project. Its data can be used to provide customer billing, resource tracking, and alarming capabilities across all OpenStack core components [46].

For the three clouds' web and application server, the detailed measured QoS metrics are showed in Table I. Availability refers to the usable level of services. Http latency refers to the response time of HTTP services. Throughput is the downloading speed provided by the server. SLA is the service level agreement that users need the service provider to guarantee. Latency refers to TCP response time. The QoS model can be obtained and constructed from these parameters and the corresponding weighting calculation. In comparison, the OpenStack Cloud1 is more economical, the OpenStack Cloud2 has better performance, and the OpenStack Cloud3 is more stable. The web and application server node configuration requirement of the experiments is listed in Table III. In Table III, server layer means different servers need different layers' server components combination. For example, web servers only need layer1-Apache server, application server need layer1-Tomcat server plus layer2-JVM plus layer3-SSH2 Framework, and database server only need layer1-MySQL server.

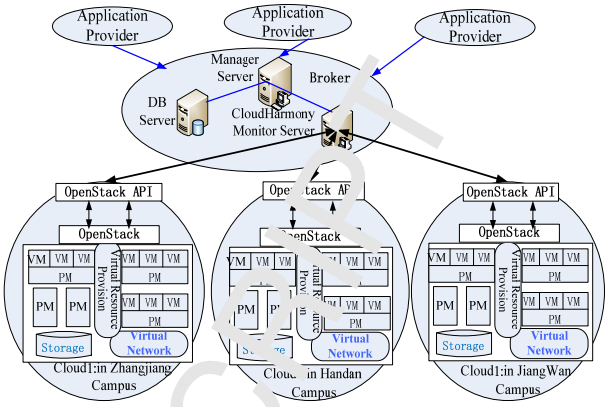



Figure 3. Multiple Edge OpenStack Clouds Experiment Topology

TABLE II. MULTI-OPENSTACK-CLOUDS EXPERIMENT ENVIRONMENT PARAMETER.

| Name | Availability | Http Latency | Throughput | SLA | Latency |
|---|---|---|---|---|---|
| OpenStack Cloud1 | 99.95% | 90ms | 62.17mb/s | 0.9 | 213ms |
| OpenStack Cloud2 | 98.95% | 50ms | 75.56mb/s | 0.9 | 189ms |
| OpenStack Cloud 3 | 99.98% | 70.33ms | 67.34mb/s | 0.9 | 209ms |

TABLE III. WEB AND APPLICATION SERVER NODE CONFIGURATION REQUIREMENTS

| | vCpu | Memory | Disk | Server Layer 1 | Server Layer 2 | Server Layer 3 |
|---|---|---|---|---|---|---|
| **Server0** | 2 | 4GB | 500 GB | Apache | | |
| **Server1** | 2 | 4GB | 500 GB | Apache | | |
| **Server2** | 4 | 8GB | 500 GB | Tomcat | JVM | SSH2 Framework |
| **Server3** | 4 | 8GB | 500 GB | Tomcat | JVM | SSH2 Framework |
| **Server4** | 4 | 8GB | 500 GB | Tomcat | JVM | SSH2 Framework |
| **Server5** | 4 | 8GB | 1TB | MySql | | |
| **Server6** | 4 | 8GB | 1TB | MySql | | |

In this experiment, we focus on single-cloud deployment of web servers through multi-cloud selection. Given the QoS requirement, we can find that the user is more concerned about the availability of the service. We input this data into the Broker system, and the broker returned the optimal service provider, which is OpenStack Cloud3. It gives the comparison of system parameters and user requirement (Table IV) and the comparison of three deployment plans (Fig. 4). The detailed comparison and result is showed in TableV. Through a series of experiment, we can verify that the Broker System is able to provide an efficient solution which is more suitable for user's demand.

TABLE IV. QOS REQUIREMENT

| QoS Parameter | Acceptable Range | Weight |
|---|---|---|
| **Availability** | > 90% | 0.6 |
| **Latency (Http) (ms)** | < 100 | 0.1 |
| **SLA** | > 0.7 | 0.1 |
| **Throughput (Mb/s)** | > 40 | 0.1 |
| **Budget ($)** | < 200 | 0.1 |

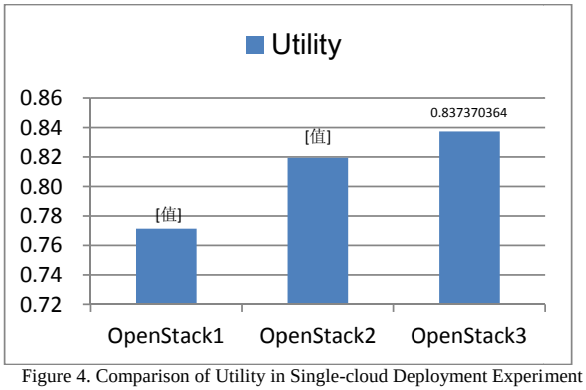



Figure 4. Comparison of Utility in Single-cloud Deployment Experiment

TABLE V. QoS Recommended result in Single-cloud Deployment Experiment

| QoS Parameter | User Requirement | Recommended Service Provider: OpenStack3 |
|---|---|---|
| **Availability** | > 90% | 99.98% |
| **Latency (Http) ( ms )** | < 100 | 90 |
| **SLA** | > 0.7 | 0.9 |
| **Throughput ( Mb/s )** | > 40 | 67.84 |
| **Budget** | < 200 | 26 |

*B.* **Multi-edge-cloud deployment of CDN**

In a larger scale environment, the log data is retrieved from Wangsu-the largest CDN operator in China. We have QoS data collected from the monitors deployed in each PoP nationwide. They send packets to each other and measures QoS metrics which is throughput (download speed) and responsiveness. Q=(throughput, responsiveness).

Throughput is the download capability of a cloud service providing the contents to clients:

$$a = \frac{s - s_{min}}{s_{max} - s_{min}}$$

$s$ denotes the current download speed, $s_{min}$ and $s_{max}$ denote the minimum and maximum of download speed. The closer $a$ is to 1, the higher the availability is.

Responsiveness is the ability of a cloud service provider to respond to the users' requests in a given time:

$$\tau = 1 - \frac{f_{i=1}^{n}(t_i)}{t_{max}}$$

$0 \le \tau \le 1$ denotes responsiveness. $t_i$ denotes the interval between the $i^{th}$ request and the completion time. n is the number of requests in a given time. $t_{max}$ is the max response time that is acceptable($t_i \le t_{max}$). $f$ can be mean function or median function. The closer $\tau$ to 1, the higher the responsiveness is.

**1）CDN Full network monitoring data**

When we compute QoS model, we need get the related QoS data. First QoS data is the download speed data. Within each hour, we choose two PoP servers in each area, one as a monitor server sends a HTTP request to another destination server from the same ISP, to download a 2MB data package, and then we get the download speed. There are about 400 nodes in each round of detecting, with the request was sent once an hour between every two points, the size of daily log data is about 404 MB.

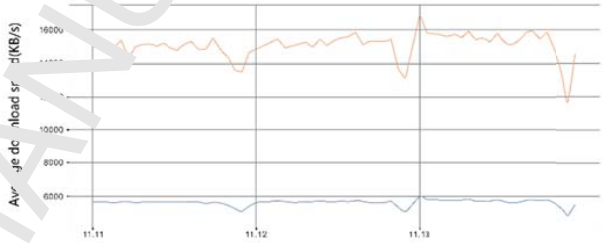



Figure 5. Temporal change of average download speed of Shanghai.

Second QoS data is the response time data. Like the HTTP request above, one monitor server sends a Ping request to another destination server and record the totally response time between the monitor server and the destination server. The detection frequency is 10 minutes 1 times, the detection packet size is 8Bytes, and the amount of log data of one day is 10 million 330 thousand lines, the total amount is 2.1GB.

Since we have more than 400 server nodes among a full network probe, the result is a 400*400 matrix, which is too big for our calculation. So we focus on the data which two server nodes are adjacent or very closely, and leave out the remote areas which network status usually worse.

We can see the temporal change of data from Nov. 11 to Nov. 13 in Fig 5. The top line represents the monitors in the same region and the bottom represents the monitors from nationwide. The download speed has a drop from 19:00 to 22:00 in the evening about 3 hours. The figure shows the periodic variation of download speed.

**2）CDN Multiple Edge Clouds Cross Region Distribution**

In Wangsu CDN, The whole of China is divided into eight regions according to geographical location, each region contains some provinces. Table VI shows the division.

TABLE VI. China region division

| REGION | PROVINCE |
|---|---|
| HUADONG | Shandong, Jiangsu, Anhui, Zhejiang, Fujian, Shanghai |

| | |
|---|---|
| HUANAN | Guangdong, Guangxi, Hainan |
| HUAZHONG | Hubei, Hunan, Henan, Jiangxi |
| HUABEI | Beijing, Tianjin, Heibei, Shanxi, Neimenggu |
| XIBEI | Ningxia, Xinjiang, Qinghai, Shaanxi, Gansu |
| XINAN | Sichuan, Yunnan, Guizhou, Xizang, Chongqing |
| DONGBEI | Liaoning, Jilin, Heilongjiang |
| GANGAOTAI | Hong Kong, Macau, Taiwan |

Generally, the scheduling policy is that one client usually visits one CDN PoP node in the same region, because the same region always has better network connection. Just like Shanghai Jiangsu and Zhejiang are parts of "Huadong region", which means they belong to Eastern China.

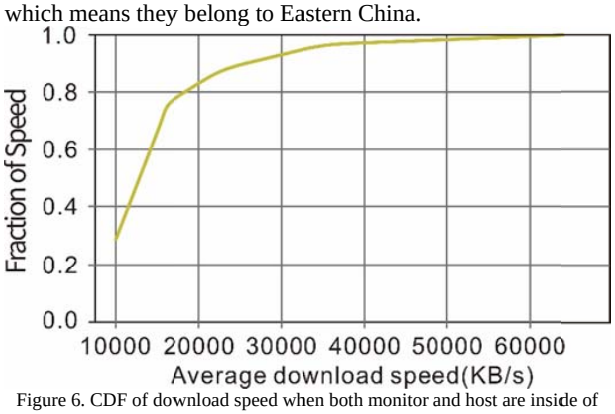



Figure 6. CDF of download speed when both monitor and host are inside of Huadong region

From Fig.6, we can observe that the monitors to the host within the same region have higher download speed up to 60000KB/s, which is much higher than those monitors to the host outside of the region, only about 8000KB/s, shown in Fig.7. Normally, the distance between a monitor and a host closer, the network status like download speed and response time better. We can see that the speed of Figure 6 is 7 times higher than that of Figure 7. That's the reason why the CDN edge cloud providers usually deploy and schedule their service according to the near service rule.

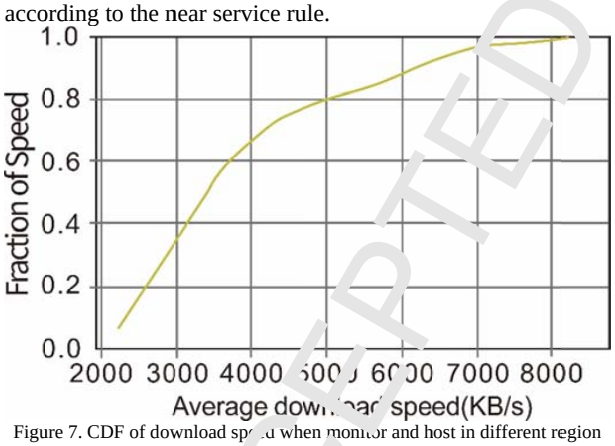



Figure 7. CDF of download speed when monitor and host in different region

But there are some exceptional situations, due to network incident, the node outside the monitor's region may have better network status than some nodes inside.

TABLE VII. The best rank of nodes outside the monitor's region

| LOCATION | RANK OF NODE | TOTAL |
|---|---|---|
| **SHANGHAI** | 54 | 74 |
| **SHANDONG** | 4 | 54 |
| **SHANXI** | 6 | 17 |
| **YUNNAN** | 22 | 36 |
| **XINJIANG** | 4 | 15 |
| **CHONGQING** | 15 | 27 |
| **LIAONING** | 3 | 19 |
| **HENAN** | 7 | 32 |
| **NINGXIA** | 2 | 15 |
| **HUBEI** | 17 | 45 |
| **GUIZHOU** | 14 | 34 |
| **ZHEJIANG** | 59 | 84 |
| **JIANGSU** | 40 | 84 |
| **SICHUAN** | 29 | 38 |

For example, in TABLE VII, we choose Shanghai as the monitor node. The region of Shanghai has 74 nodes totally, and the outside host nodes' best rank is 54, which means there are some nodes outside the region have better network connection than 22 of the nodes inside the region. From this observation, under some conditions, one server can serve the clients of other large regions.

CDN is an active application deployment scene supported by multiple edge clouds. The goal of CDN servers is to provide high availability and high performance by placing cache servers near end users in edge network. So it's important to deploy multiple cloud services in the PoPs of CDN and monitor the performance of CDN services to ensure the QoS received by the end user.

TABLE VIII. QoS requirement in multi-clouds small-object CDN experiment

| QoS Parameter | Acceptable Range | Weight |
|---|---|---|
| **Throughput** | > 100KB | 0.2 |
| **Responsiveness** | < 50 ms | 0.8 |
| **Budget** | < 200000RMB | - |

TABLE IX. QoS requirement in multi-clouds Larger-object CDN experiment

| QoS Parameter | Acceptable Range | Weight |
|---|---|---|
| **Throughput** | > 1.5MB | 0.8 |
| **Responsiveness** | < 100 ms | 0.2 |
| **Budget** | < 200000RMB | - |

In general, there are two types of PoP caches. The first kind of caches stores small objects such as web pages, pictures which are hundreds of KBs, and another kind of cache stores larger objects like videos which can be several hundred MBs. These two types of CDN cache server has very different QoS requirements. The users of the first kind of CDN server are more concerned about latency, while the users of the second one are more concerned about throughput, as shown in Table VIII and Table IX respectively. We design the small-object-oriented Web application multi-PoP deployment experiment and larger-object-oriented streaming application multi-PoP deployment experiment to verify our scheme. To emulate a real demand distribution, we assume the service demand of the first

application is from Shanghai and the service demand of second application is nationwide. The results are shown in Figure 8, Figure 9. Each point in the graph represents a deployment plan. From both the graphs we can observe that the higher the cost of the plan, the higher utility is for the users. The curves we plot with the points can be seen as the approximate Pareto-optimal fronts outputs by the NSGA-II. In Figure 8, we can observe the trade-off between utility and QoS in multi-cloud small-object CDN experiment in Shanghai region. As cost increases, utility continues to rise. When cost reaches $1.2*10^5$, utility is close to 1. The points shown in Figure 8 are the recommended utility points, and users can select the similar utility points by approaching them based on their cost budget.

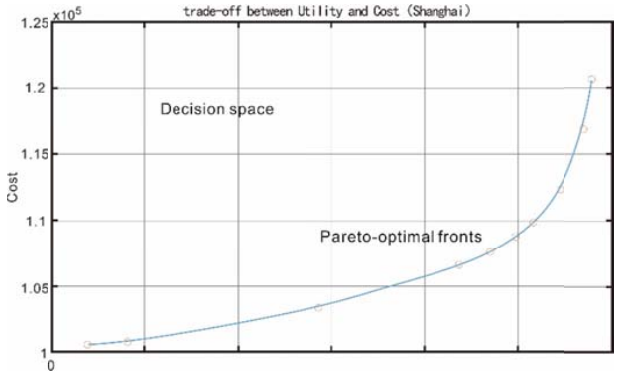



Figure 8. The trade-off between Utility and QoS in Multi-cloud small-object CDN experiment

Similar with Figure8, Figure 9 illustrates the trade-off between utility and QoS in multi-cloud larger-object CDN experiment in the national region. When cost reaches $1.0355*10^5$, utility is close to 0.83, a near-maximum value. Similarly, users can choose the recommended annotation utility points according to their own cost budget.

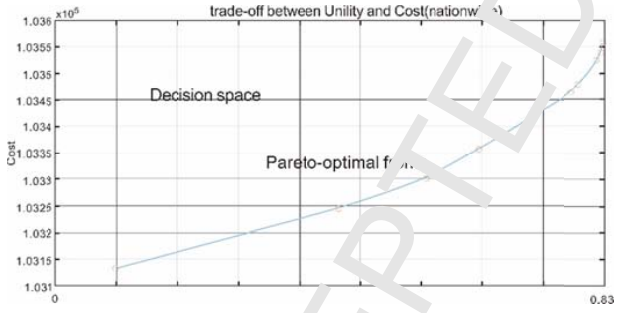



Figure 9. The trade-off between Utility and QoS in Multi-cloud larger-object CDN experiment

When a service provider need to make an efficient trade-off between utility and QoS in multi-cloud deployment, QaMeC utilize a NSGA-II searching algorithm to reduce the search space of tradeoff decision to a large extent. Now the service provider can leverage the QaMeC trade-off model to quickly choose a multi-cloud deployment plan based on cost budget and expected utility trade-off curve. Every point in the curve represents a deployment plan, so the service provider can efficiently choose a suitable deployment plan by specifying a point. We can see the figure 8 and 9 can fully finish the purpose of our model.

## VII. Conclusion and Future Work

Internet of Vehicles (IOV) as an integral part of the smart city is a distributed and integrated network system, which connects different people within automotives, different automotives in cities. The produced big data in automotives, especially multimedia data need rapid transmission and processing. Deploying IoVs applications to a centralized cloud for service delivery is infeasible, such as transporting all IoVs data to a centralized cloud for big data analysis processing, is infeasible because of the excessive latency and bandwidth limitation of the Internet. In this paper, we design, implement and evaluate QaMeC: a new IoVs application optimizing deployment solution based on QoS model in CDN multimedia edge clouds environment. We proposed a unified QoS model to eliminate the ambiguity. Then, with the data from the real CDN log data, we calculate the QoS model so as to give users a clearer evaluation of cloud service, hence improve user's confidence and loyalty. We also propose a service selection optimization algorithms in multiple edge clouds environment to improve the efficiency and accuracy of multi-clouds selection decision.

As a next step work, the QoS model can be improved to be more accurate by introducing more metrics and more data sources. Before service selection, several methods can be applied to reduce the problem space of application deployment methods. Furthermore, we also plan to improve the effectiveness of multi-cloud deployment optimization algorithm.

With the expansion of IoVs application service scale, such as many vehicle with sensors distributing in a lot of city locations, multiple edge clouds are at a same horizontal level, and some edge clouds with one centralized cloud forms an overly fat two-level flat structure. As a future work, we plan to design a scalable multi-tier edge clouds architecture to address the possible performance bottlenecks caused by this kind of two-level flat structure. The idea stems from CDN edge clouds, which itself supports multilevel cache node structures. However, the node configuration of CDN edge cloud is static, while multiple edge clouds node configurations are dynamic. According to the service size of different service providers, we need to build a hierarchical multiple edge cloud structure to meet their different requirements.

## Acknowledgment

The work of this paper is supported by National Natural Science Foundation of China under Grant No. 61728202-Research on Internet of Things Big Data Transmission and Processing Architecture based on Cloud-Fog Hybrid Computing Model, and Grant No. 61572137-Multiple Clouds based CDN as a Service Key Technology Research, Shanghai 2018 Innovation Action Plan- Hong Kong, Macao and Taiwan Science and technology cooperation project under Grant No.

18510760200- Research on smart city big data processing technology based on cloud-fog mixed mode.

## REFERENCES


[1] Yang F, Wang S, Li J, et al., An overview of Internet of Vehicles, China Communications, 2014, 11(10):1-15.

[2] Petcu, Dana, Consuming Resources and Services from Multiple Clouds, Journal of Grid Computing 12.2(2014):321-345./

[3] Petcu, D, et al., Towards Multi-Clouds engineering. Computer Communications Workshops IEEE, 2014:1-6.

[4] Cuadrado, F, et al. Research challenges for cross-cloud applications. Computer Communications Workshops IEEE, 2014:19-24.

[5] Wagle, Shyam S. Cloud Service Optimization Method for Multi-cloud Brokering. IEEE International Conference on Cloud Computing in Emerging Markets IEEE Computer Society, 2015:132-139.

[6] Aldawsari, Bandar, T. Baker, and D. England. Towards a Holistic Multi-cloud Brokerage System: Taxonomy, Survey, and Future Directions. IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing IEEE, 2015:1467-1472.

[7] Yabusaki, Hitoshi, et al. Wide area tentative scaling (WATS) for quick response in distributed cloud computing. Computer Communications Workshops IEEE, 2014:31-36.

[8] Toosi, Adel Nadjaran, R. N. Calheiros, and R. Buyya. Interconnected Cloud Computing Environments:Challenges, Taxonomy, and Survey. Acm Computing Surveys 47.1(2014):1-47.

[9] Casola, Valentina, et al. MUSA Deployer: Deployment of Multi-cloud Applications. IEEE, International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises IEEE, 2017:107-112.

[10] Duplyakin, Dmitry, et al. Rebalancing in a multi-cloud environment. ACM Workshop on Scientific Cloud Computing ACM, 2013:21-28.

[11] Del Castillo, J. A. L, K. Mallichan, and Y. Al-Hazmi. OpenStack Federation in Experimentation Multi-cloud Testbeds. IEEE, International Conference on Cloud Computing Technology and Science IEEE, 2013:51-56.

[12] Wu, Zhe, and Harsha V. Madhyastha. Understanding the latency benefits of multi-clouds web service deployments. ACM SIGCOMM Computer Communication Review 43.2 (2013): 13-20.

[13] Alexander K, Lee C, Kim E, et al. Enabling End-to-End Orchestration of Multi-Cloud Applications[J]. IEEE Access, 2017, PP(99):1-1.

[14] Alshammari, Mohammad M, et al. Disaster Recovery in Single-Cloud and Multi-Cloud Environments: Issues and Challenges. The IEEE International Conference on Engineering Technologies and Applied Sciences IEEE, 2017.

[15] Farokhi, Soodeh. Towards an SLA-Based Service Allocation in Multi-cloud Environments. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing IEEE, 2014:591-594.

[16] Yang, Jie, et al. Resource allocation policy based on trust in the multi-cloud environment. IEEE International Conference on Systems, Man and Cybernetics IEEE, 2017:3207-3211.

[17] Niu Y, Liu F, Fei X, et al. Handling flash deals with soft guarantee in hybrid cloud[C]// INFOCOM 2017 - IEEE Conference on Computer Communications, IEEE. IEEE, 2017:1-9.

[18] Niu Y, Luo B, Liu F, et al. When hybrid cloud meets flash crowd: Towards cost-effective service provisioning[C]// Computer Communications. IEEE, 2015:1044-1052.

[19] Liu F, Luo B, Niu Y. Cost-Effective Service Provisioning for Hybrid Cloud Applications[M]. Springer-Verlag New York, Inc. 2017.

[20] Tang H, Liu F, Shen G, et al. UniDrive: Synergize Multiple Consumer Cloud Storage Services[J]. Middleware '15 Proceedings of the 16th Annual Middleware Conference, 2015:137-148.

[21] A. Jonathan, M. Ryden, K. Oh, A. Chandra and J. Weissman, Nebula: Distributed Edge Cloud for Data Intensive Computing, in IEEE Transactions on Parallel & Distributed Systems, vol. 28, no. 11, pp. 3229-3242, 2017.

[22] Ola S, Imad E, Ayman K, Ali C. Edge computing enabling the Internet of Things[C]. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015: 603 – 609.

[23] Shi W, Schahram D. The promise of edge computing[J]. IEEE Journals & Magazines of Computer, 2016, 49(5):78 - 81.

[24] Papageorgiou, Apostolos, E. Poormohammady, and B. Cheng. Edge-Computing-Aware Deployment of Stream Processing Tasks Based on Topology-External Information: Model, Algorithms, and a Storm-Based Prototype. IEEE International Congress on Big Data IEEE, 2016.

[25] Sivasubramanian, S. Prasath. "QoS based web service selection. Ubicc Org (2010):190-199.

[26] Dastjerdi, Amir, and R. Buyya. A Taxonomy of QoS Management and Service Selection Methodologies for Cloud Computing. Cloud Computing. 2011.

[27] Wang, Congjie, Zhihui Lu, et al. Optimizing Multi-Cloud CDN Deployment and Scheduling Strategies Using Big Data Analysis. IEEE International Conference on Services Computing IEEE, 2017:273-280.

[28] ZhiHui Lv, Ning Wang, Jie Wu, Meikang Qiu:IoTDeM: An IoT Big Data-oriented MapReduce performance prediction extended model in multiple edge clouds. J. Parallel Distrib. Comput.(JPDC) 118(Part): 316-327 (2018).

[29] Chen M, Qian Y, Fortino G, et al. Cognitive Internet of Vehicles[J]. Computer Communications, 2018:Volume 120, Pages 58-70.

[30] Chen M, Hao Y, Hu L, et al. Edge-CoCaCo: Toward Joint Optimization of Computation, Caching, and Communication on Edge Cloud[J]. IEEE Wireless Communications, 2018, 25(3):21-27.

[31] Chen M, Hao Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-dense Network[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3):587-597.

[32] Benatia, Mohamed Amin, et al. Multi-Objective WSN Deployment Using Genetic Algorithms Under Cost, Coverage, and Connectivity Constraints." Wireless Personal Communications (2017):1-30.

[33] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2):182-197.

[34] Wei Zhang, Zhihui Lu, Ziyan Wu, Jie Wu, Huanying Zou, Shalin Huang:Toy-IoT-Oriented data-driven CDN performance evaluation model with deep learning. Journal of Systems Architecture - Embedded Systems Design 88: 13-22 (2018).

**First Author** Ziyan Wu is a master student at School of Computer Science, Fudan University. His research interests are cloud computing, multi-cloud deployment, and big data process architecture.

**Corresponding Author** ZhiHui Lu is an Associate Professor in School of Computer Science, Fudan University. He received a Ph.D computer science degree from Fudan University in 2004, and he is a member of the IEEE and China computer federation's service computing specialized committee. His research interests are cloud computing and service computing technology, big data architecture, edge computing, and software defined network.

**Third Author** Patrick C.K. Hung is a Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology. Patrick has been working with Boeing Research and Technology in Seattle, Washington on aviation services-related research projects. He owns a U.S. patent on Mobile Network Dynamic Workflow Exception Handling System with Boeing. His research interests include services computing,cloud computing, big data, business process and security.

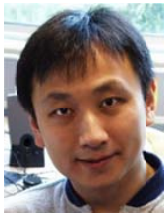
**Forth Author** Shih-Chia Huang is a Full Professor with the Department of Electronic Engineering at National Taipei University of Technology, Taiwan, and an International Associate Professor with the Faculty of Business and Information Technology at the University of Ontario Institute of Technology, Canada. He has been named a senior member of the Institute of Electrical and Electronic Engineers (IEEE). He is currently the Chair of the IEEE Taipei Section Broadcast Technology Society, and was a Review Panel Member of the Small Business Innovation Research (SBIR) program for the Department of Economic Development of Taipei City and New Taipei City, respectively. Professor Huang has published more than 80 journal and conference papers and holds more than 60 patents in the United States, Europe, Taiwan, and China.

**Fifth Author** Yu Tong a master student at School of Computer Science, Fudan University. His research interests are cloud computing, multi-cloud deployment, and big data process architecture.

**Sixth Author** Zhenfang Wang is a master student at School of Computer Science, Fudan University. His research interests are cloud computing, multi-cloud deployment. He has already graduated.

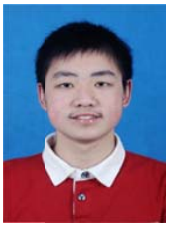
**First Author** Ziyan Wu

**Corresponding Author** ZhiHui Lu



**Third Author** Patrick C.K. Hung

**Forth Author** Shih-Chia Huang



**Fifth Author** Yu [illegible]ng

**Sixth Author** Zhenfang Wang

Highlight

In this paper, we propose QaMeC: a novel QoS-driven IoVs application service optimizing deployment scheme in CDN multimedia edge clouds environment.

The proposed service demand model and QoS model can provide a complete description of user requirements. It gives a quantitative description of the service request and delivery. The QoS data is retrieved from the PoPs log data of the real CDN operator network, which ensures the objectivity for the users, overcoming the drawbacks of the SLA-based solution.

The designed NSGA-II algorithm is applied to search for the best deployment plan for the users, to reduce the vast problem space of the combinatorial optimizing decision-making problem.

The implementation and experiments show that our QaMeC scheme can provide optimal and efficient service deployment solutions for a variety of applications with different QoS requirements in CDN multimedia edge clouds environment.