# A Scalable Internet-of-Vehicles Service over Joint Clouds

Yong Zhang[1], Mingming Zhang[1], Tianyu Wo[1,2], Xuelian Lin[1,2], Renyu Yang[2,3], Jie Xu[3,2]
[1]*SKLSDE, Beihang University, Beijing, China*
[2]*BDBC, Beihang University, Beijing, China*
[3]*University of Leeds, Leeds, UK*
*Email: {zhangy11, zhangmm, woty, linxl}@act.buaa.edu.cn; {r.yang1, j.xu}@leeds.ac.uk*

***Abstract*—Since the Internet of Vehicles (IoV) technology has recently attracted huge research attentions, IoV service that can collect, process data and further provision services is increasingly becoming the mainstream. Considering the process efficiency, geo-distributed data is typically collected and exploited on different Clouds, making it significantly essential for IoV application to be deployed on multiple Clouds whilst system components still function well and jointly work. In this paper, we provide a scalable IoV system deployment in the joint Cloud environment where cloud vendors collaboratively cooperate as an alliance. In particular, system components are independently deployed in accordance with the data placement and resource capacities etc. Multiple replication mechanism is utilized to achieve the cross-cloud parallel processing, thereby effectively handling the scalability issues in the massive-scale vehicle data processing. Furthermore, we adopt the multi-source data fusion to facilitate the accuracy of IoV data analytics. We demonstrate the effectiveness of the proposed approaches through real-world use cases including fleet distribution management and passenger demands prediction.**

## 1. Introduction

The advanced Cloud computing techniques have reshaped numerous traditional industries. For instance, the vehicle and transportation industry are significantly influenced by the Cloud computing and mobile technology. Consequently, Internet-of-Vehicles (IoV) application is increasingly becoming the mainstream solution which leverages the Cloud datacenter to collect and process vehicle streaming data and further provide transportation-related services to users.

Such Cloud-based IoV systems usually consist of multiple components. Those components work together collaboratively to achieve a number of functionalities of a complex system. Traditional solutions to such scenario basically manage and process the data within a proprietary Cloud datacenter. For example, Cloud service provider such as Google [1], Microsoft [2], and AWS [3] proposed their own IoV-oriented Cloud services. However, in some real-world cases, different service components and the data store usually require to be deployed and managed in multiple Cloud platforms for the performance consideration (e.g., reduced latency and increased data transmission rate etc.).

There are three reasons for such deployment – Firstly, it is very common for multi-organizations to participate in the business and academic cooperations. It seems inappropriate to completely deploy a monolithic system within a single datacenter considering the frequently manifestations of system failures. Once a critical system component suffers from the power outage or late-timing failures, the holistic service chain might be affected, resulting in a service suspension [4] [5]. Also Organizations have their specialized requirements and priorities in terms of the provisioned quality of service(QoS) such as latency, throughput and reliability, etc.; Secondly, in the modern real world, environments may change rapidly and unexpectedly [6] [7]. Meanwhile, the data is generated everywhere, and the massive amount of data is collected and stored in geo-distributed Cloud datacenters. Due to the unaffordable network communication cost, the widely-distributed data cannot be easily moved across the Clouds. It is far more cost-effective to properly schedule computation tasks adjoin to the data; The third reason why such a large amount of data cannot be physically aggregated is the data confidentiality. In fact, not all organizations possess the same wealth of data, and the only means to access the data is to share collaboratively but restrictedly over the joint Cloud storage vendors.

In this context, the deployment of IoV systems over joint Clouds poses a number of challenges – The system must be deployed to ensure the existing modules to work smoothly whilst multiple constraints being satisfied. Additionally, to meet the growing data volume and latency requirements, the system should be highly scalable to timely handle the geo-distributed data at scale. Also, those distributed data is supposed to be flexibly managed for the effective access by data analysis applications. In this paper, we firstly introduce an evolutionary version of our previous CarStream [8] and its deployment from scalability perspectives over joint Clouds. To summarize, we make the following contributions:

- A scalable deployment scheme to ensure IoV services can best fit the circumstance over multiple Clouds with QoS requirements of multiple organizations satisfied.
- A cross-Cloud parallel processing approach based on geo-distributed data partitioning which can effectively promote the scalability of large-scale vehicle data

processing.

- A multi-source data fusion that allows for the increased accuracy and efficiency of machine learning (ML) applications.

## 2. Background and Problem

### 2.1. CarStream System Overview

Numerous IoV systems such as Connected Vehicles [2], Google Cloud Platform [1], CARASSO [3] and CarStream are used to schedule tens of thousands of vehicles daily and are critical for commercial, personal and research pursuits.

Among these systems, CarStream is a representative Cloud-based IoV management system we design and implement to serve for a real-world chauffeured car service. The proposed approaches had been widely-adopted into UCAR [9]. Regarding the business mode within UCAR, passengers book trips through a car requesting platform and drivers take orders from passengers through mobile phones. For each driver, his business-related operations (e.g., taking orders, changing service status) conducted by a mobile phone will be uploaded to the server. Besides, to ensure the safety and quality of the trip services and support timely fleet scheduling and long-time operational management, they deploy an Onboard Diagnostic (OBD) connector and a data-sampling device for vehicles. Specifically, the OBD collects parameter values of vehicle status such as speed, engine RPM, and vehicle error code at a pre-defined interval (e.g., 5 seconds) when the vehicle is running, and the collected data are transmitted back to the backend server through an integrated wireless-communication module.

In general, CarStream collects, stores, and analyzes these uploaded data in terms of vehicle and driver management as a back-end service. As shown in Figure 1, the architecture of CarStream mainly comprises four modules: (1) data receiving/dispatching layer; (2) data bus layer; (3) processing layer; and (4) data management layer. At the first stage, CarStream was deployed in an individual Cloud. Now it has been deployed over joint Cloud environment and over 30,000 vehicles distributed among 60 different cities in China are connected to this system. As a result, it needs to process nearly 1 billion data instances per day.

### 2.2. Joint Cloud Environment

Cloud computing has been advanced for over a decade and can elastically provision flexible resources and automatic maintenance, thereby tremendously reducing the operational costs. However, with the resources and data increasingly being geo-distributed, a given Cloud provider can hardly deal with such a widely-distributed data storage and a massive number of user requests unless more infrastructures are invested and constructed. It is inconceivable for any service providers to build a monolithic system at scale to manage across-region resources including storage and computational tasks. To this end, the JointCloud architecture [10] is proposed to tackle this issue inspired by the flight alliance mechanism. Since different Cloud vendors

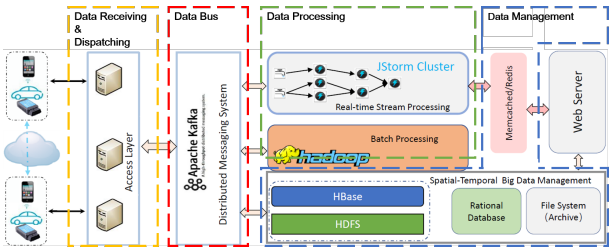



Figure 1. An overview of CarStream.

are independently connected, it is critical to activate the cooperation through a joint collaboration mechanism. In this context, a cross-Cloud cooperation architecture aims for the integrated Internet service customization by empowering the cooperation among multiple Cloud vendors to provide cross-Cloud services [11]. JointCloud takes the first step towards providing an evolving ecosystem so that all Cloud providers can serve the global computation cost-effectively with high availability and QoS guaranteed.

### 2.3. Basic Requirements and Principles

**System Module Decoupling.** Decoupling functional modules of a large-scale system into different components plays a significantly important role in the rapid progress of software systems. This can not only simplify the construction and maintenance of such systems at scale, but also allow for the system evolution of existing components for the scalability consideration. The system scalability will be extremely challenging particularly when the business scale is continuously boosting. To this end, the microservice architecture [12] is prevalent and urgently desirable to minimize the negative impact of monolithic application on the system development and evolution.

**Data Partitioning and Adjoin Processing.** To accelerate the data processing, the data is partitioned into different data shardings [13] and multiple replicas of the same compute task are responsible for separate data shards to enable the parallelized processing.

**Collaborative Data Sharing over Joint Clouds.** For example, the passenger demands prediction [14] would be improved if multi-source data such as weather data, trajectory data and map data were holistically involved. Therefore, it is necessary to deploy such an information querying interface under some customized constraints for Cloud storage.

### 2.4. Research Questions

The aforementioned discussion raises the following research problems:

- **[Q1]** How to optimize the process of choosing the best Cloud datacenter for hosting microservices under multiple constraints including the resource capacity, throughput, service response latency or other system costs?
- **[Q2]** How to characterize the geo-distributed data and effectively schedule the data processing component by launching and placing the multiple replicas?

- **[Q3]** How to exploit multi-source data over joint Cloud environment to improve the effectiveness of the data analysis within the IoV system?

## 3. System Component Deployment

This work involves three organizations. While we (BUAA) design and implement CarStream to do innovative research about IoV, UCAR hopes it can support their business efficiently and stably. Besides, OBD devices are produced by a company named Launch [15] and therefore the uploaded data will collected by them first. In this section, we display the criteria for the component placement and orchestration and how we deploy the system in practice.

### 3.1. Criteria for Component Placement

There are a number of system constraints when deploying different system components with differentiated requirements into the joint cloud environments.

- **Resource Capacity.** The resource capacity of Cloud providers are heterogeneous in many aspects. The computing machines that constitute the data center cluster are typically constructed from commodity hardware owing to significantly reduced merchandising and operational costs. The configuration diversities among different machines lead to the huge discrepancies in a cluster. These diversities can be characterized using dimensions such as micro-architecture, machine chipset version, CPU and memory capacities etc. Cloud providers need to be prepared to manage highly heterogeneous workloads that are served on the top of shared infrastructure.
- **Monetary Costs amd Pricing.** The pricing of Cloud vendors is dynamically fluctuated to maximize the revenue among the competitors. In order to strictly guaranteeing the system's performance and customer's SLA, it is highly desirable to strike a balance between the performance and the monetary costs. In fact, the diverse capacity of the provisioned resource and flexible charging scheme of joint Cloud providers enable the scalable and cost-effective deployment of numerous service components.
- **Geographical Location.** Long-distance will undoubtedly increase the response latency when invoking remote process calls or data queries from a cloud database service due to the uncertain network transmission. Therefore, the cross-realm and trans-ogranizational deployment scheme and the resultant replication management give rise to the geographical location and invoking frequency.
- **Data Dependence.** In general, IoV systems are time-critical and sensitive to most recently generated data. Namely, the value of data will sharply decrease with the time elapse. In this scenario, the order and dependencies of the real-time data streaming across multiple clouds have to be carefully designed. The latency-sensitive online services require to be prioritized during the orchestration.

### 3.2. A Practical Deployment Solution

We deploy the system in joint cloud environment. As shown in Figure 2, four cloud environments are involved in the deployment. The grey part represents the components deployed on Azure [16] and responsible by Launch. This part receives data and dispatches them to core components of the IoV system. The green, blue and orange parts represent three variants of CarStream, placing emphasis on different purposes and are deployed in IDC of UCAR, Azure and IVIC [17] respectively. The blue part contains some light components that support stream processing, and serves key operation (e.g., vehicle tracking, trip segmentation) of UCAR. The green part includes stream-processing subsystem and heavy storage system. Tasks involving sensitive data and some relatively insignificant business processes are conducted here. The orange part contains all components of Carstream and store all historical data. Lots of innovative IoV research is evaluated, such as online applications, offline data analysis, and the mature algorithms and models.

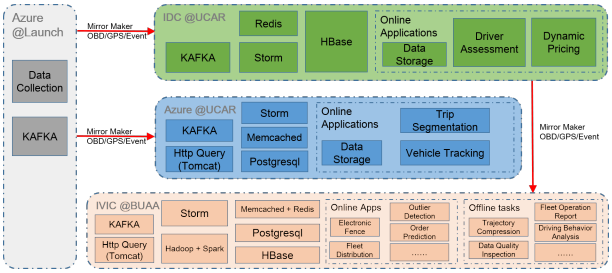



Figure 2. Multi-organization Oriented Deployment Scheme

Corresponding to above criteria, we can see the most crucial business process of this system is deploy on Azure (i.e., grey and bule parts) due to the high QoS of this cloud platform. However, since the high charge of Azure, it's not necessary to put the whole system on it. Therefore, other works are conducted in cloud environments based on a datacenter of UCAR (i.e., green part) and IVIC (i.e., orange part). Also, what run on IVIC will not affect the online business of UCAR, and UCAR can query most data from local replication directly, avoiding remote invoking from IVIC. Furthermore, it should be noted that the collected data are first delivered to UCAR to ensure the temporal value of real-time data.

## 4. Data Sharding

### 4.1. Partitioning for Geo-distributed Data

Stream-processing applications within CarStream usually have a strictly real-time requirement. However, the stream-processing component deployed in a single Cloud face increasing pressure to provision uninterrupted and reliable service. The pressure will be exacerbated by the soaring vehicle number and the resultant growing of the data volume since the the provider cannot easily deal with such issue when the CPU and memory resources are limited. For example, the buffered queue length of stream-processing subsystem during the peak hours even exceeds $3,300$ [8].

To solve this problem, we present a data sharding based method to accelerate the processing in parallel. As far as

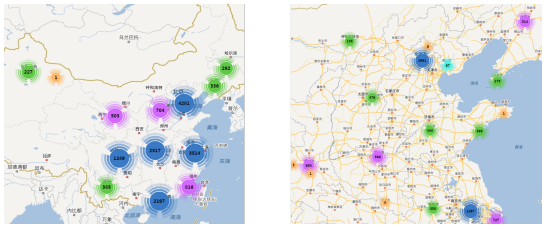


Figure 3. An example of multi-granularity fleet distribution monitoring.

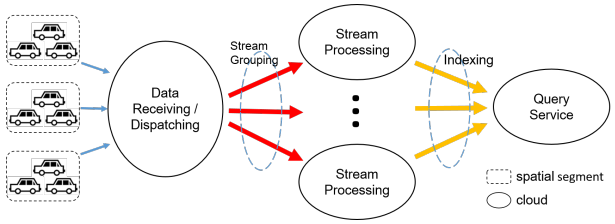



Figure 4. An illustration of parallel processing in joint cloud environment.

we know, it is observable that in some specific applications, stream data can be divided into different partitions according to some features (e.g., geographical location) and those partitions can be independently stored and processed in the subsequent procedures. Figure 3 illustrates a real-time snapshot of the geographical distribution of vehicles in China. Vehicles data from different spacial areas can be collected and computed independently. Therefore, streaming data belonging to different partitions can be dispatched to the replica of processing component in different Clouds. Figure 4 depicts the proposed stream-processing architecture. Components that are responsible for the data receiving and dispatching, stream-processing and the query are separately deployed into different Cloud infrastructures. It is noteworthy that multiple replicas of the stream-processing components are simultaneously deployed and executed in parallel in different location. There is no inter-communication among them due to the data partition and grouping strategy. In this context, each stream-processing component will calculate and maintain the intermediate results locally and those results are aggregated through the query component by utilizing a indexing to access and merge valuable data from different stream-processing sources. In this manner, the scalability bottleneck during the processing can be greatly mitigated.

### 4.2. Implementation

Following the aforementioned philosophy, we first partition the map of China according to multi-granularity grids so that the fleet data can be monitored and collected at different zoom levels (see Figure 3). Afterwards, we utilize the data uploaded by vehicles to compute the number of vehicles and the cluster centers of vehicles for every grid in real-time. The whole procedure is depicted in Figure 5,

To parallelize the data processing, we divide the geographic space of China into three non-overlapping segments in accordance with the data volume along the grid lines

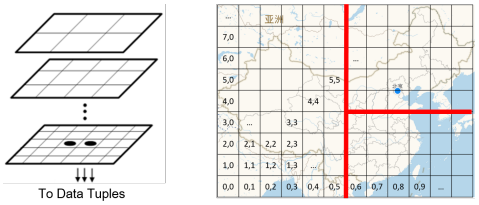



Figure 5. An illustration of grid partition.

(represented by red bold line in Figure 5). The location data of vehicles within each spatial segment are charged of the corresponding stream-processing component deployed within a specific Cloud provider. The Kafka [18] software stack is deployed as the data receiving/dispatching component in another Cloud provider. When receiving a data package uploading, the dispatching component first extracts the necessary data fields (vehicle_id, timestamps, longitude, latitude) before constructing a message. It then determines which spatial partition the message pertains to. Based on the result, the message will be added subsequently to the corresponding Kafka topic.

As for three stream-processing component, due to they are located on three stream-processing clouds, they only need to pull data from corresponding topic in the Kafka and process them based on the original processing logic and update the results in real-time. However, the scenario raises the *cross-part problem* when a vehicle moves from one spatial segment to another segment. To solve this problem, we augment a processing logic when dispatching the data. In particular, if we find out a vehicle is traversing spatial parts according to its uploading data, we will add this data package to both the Kafka topic of old part and the topic of new part. This will make sure that the stream-processing submodule that is responsible for the old one can recognize the movement of the vehicle. Furthermore, users and system administrators can leverage the query service to find out the real-time fleet distribution.

## 5. ML Application Case Study: Passenger Demands Prediciton through Multi-source Data

### 5.1. Cross-cloud Multi-source Data Fusion

Some applications need to use multi-source data on multiple clouds. For example, an important work of machine learning (ML) task is the feature engineering, and constructing a feature vector may use a wide range of data. For example, besides vehicle distribution and passenger demands in adjacent time which are extracted from UCAR fleet data, a model may also need to make use of the map and weather information to predict passenger demands more accurately. However, these data sources are usually maintained on different clouds and it is unlikely to aggregate the data together. In this section, we present how to realize the ML application by utilizing multi-source data in joint cloud environment.

Since it is unreasonable to directly put all original data together, the only way we can do is to extract feature

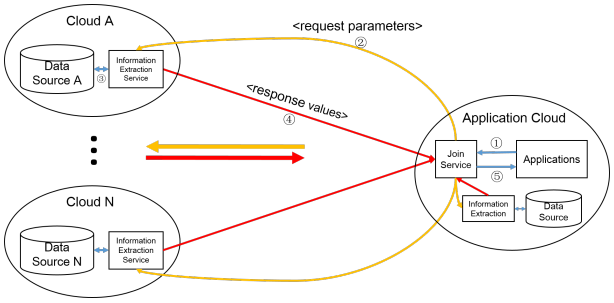
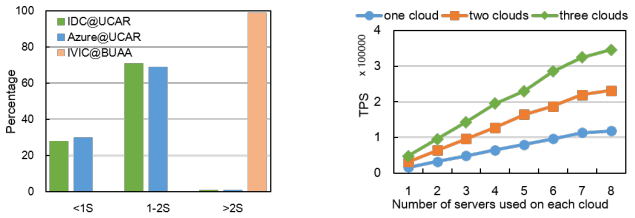



Figure 6. An illustration of cross-cloud multi-source data fusion.




(a) A statistic of the delay in data processing.

(b) The scalability evaluation of joint cloud deployment.

Figure 7. System performance.

information from cross-cloud multi-source data and merge them. As is shown in Figure 6, we deploy an information extraction component in every data cloud and a join component in application cloud where ML tasks run. For an object to be classified or regressed, it has some attributes. For instance, in advertising click-through-rate (CTR) prediction applications, the object is the possibility of clicking an advertisement by a user. The corresponding attributes can be $< user\_id, time, user\_location, advertisement\_id >$. On receiving an object from applications, the join component dispenses attributes of this object to the corresponding information extraction service components in different data clouds (e.g., user cloud, advertisement cloud, map cloud). Afterwards, it collects the respondent feature information (e.g., user profiles, advertising properties, POI information of the location) from each data cloud and merges them into a feature vector. Finally, The ML application will leverage these feature vectors for training or predicting. Similarly, other information can be easily incorporated into this procedure. For example, if the CTR prediction is permitted to access an event data cloud, the event information can be extracted according to user location and time (i.e., what happens when a user browse an advertisement at a location at a time).

### 5.2. Implementation

We implement a passenger demands prediction method which uses fleet data, weather data and map data altogether. The data is stored in separate datacenters and we deploy the information extraction service component to each of them respectively. Herein, our aim is to predict the passenger demands of spatio-temporal areas of cities (e.g., how many people will hail a vehicle in a specific geographic area from 9:00 to 9:30). Simply speaking, the attributes of a spatio-temporal area to be predicted can be $< date, day\_of\_week, time\_slice, city\_id, region\_id >$. To construct a feature vector of a spatio-temporal area, prediction application first deliver the area information to the join service component. The join service sends spatial attributes of this area (i.e., city_id, region_id) to the Cloud storage for the map, sends temporal attributes of this area (i.e., date, time_slice) to Cloud storage for weather, and sends spatio-temporal attributes of this area (i.e., date, day_of_week, time_slice, city_id, region_id) to the Cloud storage for the fleet respectively. Then information extraction services within these clouds extract the corresponding information from the data sources they own according to the attributes they received, and return these information (i.e., weather condition, POI categories, fleet distribution and temporal-adjacent demands related to this spatio-temporal area) to the join service, where these information are finally merged to be a feature vector. If it is in the scenario of generating training data, a label also be attach to this feature vector to form a training sample, which will be subsequently stored in a sample database for reusing. Otherwise, a prediction model will import this feature vector and produce a prediction result of passenger demands.

## 6. Evaluation

### 6.1. Experiment Setup

We set up three groups of experiments: (1) to evaluate the data transmission performance of practical joint cloud deployment (see Figure 2), we use real-time vehicle tracking as the test application to measure the delay of data reaching each cloud. Specifically, we compare the timestamps between the data being generated and the data entering in-memory caching platform, before which they go through buffering platform and stream processing platform; (2) we use real-time fleet distribution monitoring as the test application to test the joint-cloud processing scalability by increasing the cross-cloud deployment scale of the processing subsystems. Specifically, on one cloud, we deploy the processing subsystem on one server and increase the number of servers gradually. Meanwhile, we test the performance in joint-cloud environment by increasing the number of clouds used. By injecting the historical dataset back into the system with high frequency, we evaluate the relationship between the throughput and the system scale. The processing subsystem is deployed on a virtualized cluster. Each server has 16GB memory, a 2-core 2.0 GHz CPU and 2*10 Gbps Emulex NIC; and (3) to measure the effectiveness of weather information and map information, we first test our model by only using fleet features ($F_b$), and then we add weather features ($F_w$) and map features ($F_m$) to this model.

### 6.2. Result Analysis

**End-to-end Delay Evaluation.** As Figure 7a shows, the average delay in IDC and Azure of UCAR is less than 2

TABLE 1. EXPERIMENT RESULTS OF DIFFERENT FEATURES

| Features | $F_b$ | $F_b + F_w$ | $F_b + F_m$ | $F_b + F_w + F_m$ |
|---|---|---|---|---|
| MAE | 3.38 | 2.54 | 2.69 | 2.23 |
| RMSE | 8.06 | 6.98 | 7.45 | 5.62 |

seconds, while that in IVIC of BUAA is more than 2 seconds since the data received by IVIC are reposted by IDC of UCAR. Sush delay is acceptable by both UCAR and BUAA.

**Throughput Evaluation of Multi-replica Processing.** The result reported in Figure 7b shows a near linear joint-cloud scalability of the processing subsystem. This performance can be attributed to three reasons: the distributed design of the processing platform, the naturally distributed characteristic of the vehicle data, and little communication between clouds responsible for processing.

**Effectiveness Evaluation of Multi-source Data.** The result reported in Table 6.2 shows the new features diminish the mean absolute error (MAE)[1] and root mean squared error (RMSE)[2] in prediction, and weather features are more effective than map features. Since the revenues of different features are different, we should balance the revenue and the cost of introducing a new data source in practice.

## 7. Related Work

The Internet of vehicles has become an important development direction of automobile industry, and cloud data processing center is the technical core. Microsoft launched cloud service platform of Internet of things (IoT). One of the main services is IoV services [2], including on-board lifecycle management services, information services, vehicle driving auxiliary services, advanced navigation services, etc. Similarly, Google launched Connected Vehicle Platform [1], which are able to handle extremely rich driving the predefined data types, including vehicle location, vehicle condition, environmental information, and sensor data. Amazon Web Services also provides a cloud platform for IoT [19], the platform integrates Lambda, Kinesis, S3, Machine Learning, DynamoDB, CloudWatch and Elasticsearch, supporting stream processing, data analysis, platform monitoring, etc. In addition, traditional vehicle manufacturers are also actively developing IoV systems and providing in-car services [3], [20]. In contrast, Baidu [21] and Apple [22] provide on-board services in relatively easy ways by connecting mature mobile applications to vehicles directly.

## 8. Conclusion

In this paper, we present a scalable deployment and evolution of an Internet-of-Vehicles service over joint cloud. A multi-organization oriented deployment scheme, a cross-cloud parallel processing mechanism and a cross-cloud data fusion method is described in detail. We have experimented on real-world data and our study shows our exploratory approaches is effective in joint cloud scenarios. Regarding the future works, we plan to study more automatic and intelligent deployment mechanisms in joint cloud scenarios.

1. https://en.wikipedia.org/wiki/Mean_absolute_error
2. https://en.wikipedia.org/wiki/Root-mean-square_deviation

## Acknowledgments

This work is supported by the National Key Research and Development Program (2016YFB1000103) and the National Natural Science Foundation of China (61421003). This work is also supported by Beijing Big Data and Brain Computing (BDBC) innovation center.

## References

[1] Google, "Google cloud platform," https://cloud.google.com/solutions/designing-connected-vehicle-platform, 2015.

[2] Microsoft, "Connected vehicles," https://www.microsoft.com/en-us/internet-of-things/connected-vehicles, 2013.

[3] AWS, "Aws case study: Bmw," https://aws.amazon.com/cn/solutions/case-studies/bmw/, 2016.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE TDSC*, 2004.

[5] R. Yang, Y. Zhang, P. Garraghan *et al.*, "Reliable computing service in massive-scale systems through rapid low-cost failover," *IEEE TSC*, 2017.

[6] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, 2017.

[7] R. Yang and J. Xu, "Computing at massive scale: Scalability and dependability challenges," in *IEEE SOSE*, 2016.

[8] M. Zhang, T. Wo, T. Xie, X. Lin, and Y. Liu, "Carstream: an industrial system of big data processing for internet-of-vehicles," *VLDB*, 2017.

[9] "Ucar," https://www.crunchbase.com/organization/ucar, accessed: 2017-11-28.

[10] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperation architecture for integrated internet service customization," in *IEEE ICDCS*, 2017.

[11] X. Lu, H. Wang, J. Wang, J. Xu, and D. Li, "Internet-based virtual computing environment: Beyond the data center as a computer," *Future Generation Computer Systems*, 2013.

[12] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: migration to a cloud-native architecture," *IEEE Software*, 2016.

[13] R. Cattell, "Scalable sql and nosql data stores," *ACM SIGMOD*, 2011.

[14] H. Wei, Y. Wang, T. Wo, Y. Liu, and J. Xu, "Zest: a hybrid model on predicting passenger demand for chauffeured car service," in *ACM CIKM*, 2016.

[15] "Launch," http://www.cnlaunch.com/, accessed: 2017-11-28.

[16] "Microsoft azure," https://en.wikipedia.org/wiki/Microsoft_Azure, 11, accessed: 2017-11-28.

[17] "Ivic," http://ivic.act.buaa.edu.cn/, accessed: 2017-11-28.

[18] "Kafka," https://kafka.apache.org/, accessed: 2017-11-28.

[19] AWS, "Aws internet-of-things," https://aws.amazon.com/iot, 2016.

[20] I. Lunden, "Bmw signs on to develop in-car ai and iot services with ibm's watson," https://techcrunch.com/2016/12/15/bmw-signs-on-to-develop-in-car-ai-and-iot-services-with-ibms-watson/.

[21] Baidu, "Baidu internet-of-vehicles," http://che.baidu.com/, 2015.

[22] Apple, "Carplay," https://www.apple.com/ios/carplay/available-models/, 2015.