From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices

Hua-Jun Hong
Department of Computer Science, National Tsing Hua University, Taiwan

***Abstract*—Fog computing extends cloud computing to end devices, in order to better support time dependent, location dependent, massive scale, and latency sensitive applications. In this paper, we propose a fog computing ecosystem, implement a real testbed, and evaluate it with diverse usage scenarios. More specifically, we study three usage scenarios and optimize our fog computing platform for them. These usage scenarios are: (i) content dissemination in challenged networks, (ii) crowd-sourced fog computing, and (iii) programmable Internet-of-Things analytics. We leverage and enhance open source projects to realize our fog computing platform. We then solve optimization problems in each usage scenario with novel algorithms. Sample results show that our proposed algorithms outperform baseline algorithms by at least 30.3%, 20.0%, and 89.4% in terms of the main performance metrics of the three usage scenarios, respectively. Several ongoing tasks aim to improve our fog computing platform for: (i) network resource provision, (ii) system dynamics adaption, and (iii) device availability prediction.**

I. Introduction

Fog computing becomes popular because cloud computing is not enough to deal with the large amount of data generated by increasing number of connected Internet-of-Things (IoT) devices. The number of devices was 15.41 billions in 2015, has reached 20.35 billions in 2017 and is expected to be 30.73 billions in 2020 [2]. Such a staggering number of devices leads to many issues, including network congestion, delay, and privacy concerns if the data is analyzed on the cloud. More specifically, transmitting the data to the cloud consumes many network resources, congests networks, and leads to long latency. Moreover, the data generated by the devices, such as home surveillance cameras are sensitive. Sending them to the cloud through the Internet leads to privacy issues.

To solve these issues, we utilize the concept of fog computing, which integrates *fog devices*, including end devices, edge network devices, and cloud servers. Utilizing these heterogeneous fog devices benefits: (i) time dependent, (ii) location dependent, (iii) massive scale, and (iv) latency sensitive applications, which are not suitable to be run in the cloud. Fig. 1(a) uses a robber as an illustrated example. The robber lives in a city with fog devices installed everywhere. The robber tends to commit a crime at midnight (time). The police department therefore sends a request to the fog for a sound detection application between 22:00 to 4:00. One day, the robber fires a shot on a street (location). The fog devices with microphones on street lights detect the unusual large volume. The sound detection application is then deployed on massive fog devices (scale) to immediately analyze the sound and notify the police as soon as possible (latency).

This work was partially supported by the Ministry of Science and Technology of Taiwan under grants 106-2221-E-007-101 and 105-2221-E-007-088. The authors also appreciate the generous supports from Lite-On Technology Corporation.

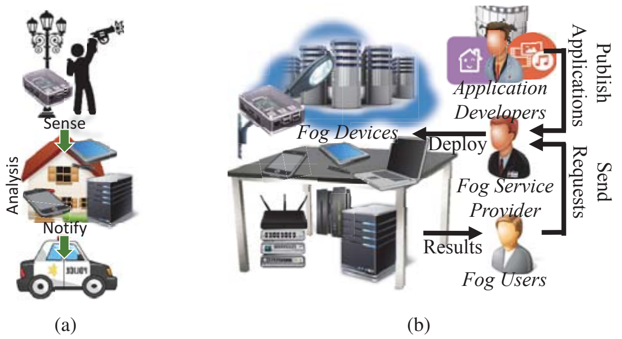


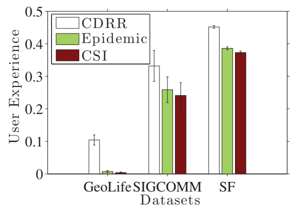
Fig. 1. Fog computing platform: (a) a sample usage scenario: gunshot detection and (b) envisioned ecosystem.

Fig. 1(b) shows the three parties in our envisioned fog computing ecosystem: a fog service provider, fog users, and application developers. In the robber example, the police department is the fog user who sends requests to the fog service provider for the sound detection application implemented by the application developers. The fog computing platform concurrently executes multiple applications. However, it runs into several challenges, which can be solved by various optimization problems. We consider three different usage scenarios in this paper.

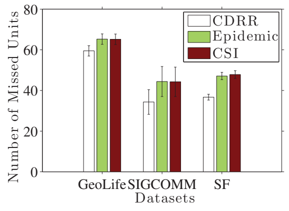
- **Content dissemination in challenged networks.** Under challenged networks, disseminating all content with highest quality is impossible because of scarce network resources. Therefore, maximizing the user experience of received contents is a crucial optimization issue to this usage scenario.
- **Crowd-sourced fog computing.** Computing data on fog devices takes unpredictable time because of uncertain behavior of the fog devices and applications, e.g., personal computers may be turned off anytime. Hence, accurately predicting the completion time is critical to this usage scenario.
- **Programmable IoT analytics.** Running IoT analytics, such as the example of robber requires not only computing resources but also spatial-temporal and sensor constraints. Hence, considering diverse requirements to

maximize the number of satisfied IoT analytics is important in this usage scenario.

In the next three sections, we present our preliminary work in these three usage scenarios.



(a)



(b)

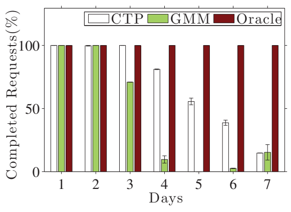
Fig. 2. Service quality improvement of our CDRR algorithm: (a) user experience and (b) missed units [13].

II. Content Dissemination in Challenged Networks: User Experience Maximization

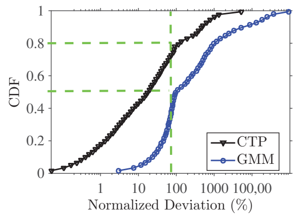
Content dissemination is mature all over the world except for areas under challenged networks. We focus on fog users living in challenged networks, who request for contents. We install local proxies, which have storage and WiFi interface as fog devices in popular locations, such as markets. The local proxies help to disseminate contents to fog users' mobile phones when they run into local proxies. We also utilize the mobile phones as fog devices to help disseminate contents within ad-hoc networks. The fog service provider has to optimally utilize the limited resources to disseminate contents and maximize user experience. To solve this problem, we propose an efficient heuristic algorithm called Contact Driven Round Robin (CDRR) with three intuitions: (i) disseminate content with higher user experience first, (ii) disseminate content to fog users with higher contact possibility first, and (iii) download content from fog users' fog devices (mobile phones) with lower contact possibility first.

To derive system models for the relationship between user experience and multimedia contents, we conduct a user study with 182 participants and 24 news reports, which are transcoded into 5 representations, including text, audio, low-, medium-, and high- resolution videos from lowest to highest representations. The user study shows that higher representation leads to higher user experience. The improvement of user experience compared to the next lower representation is diminishing. Based on our sample results, receiving the text results in 55% user experience improvement, while receiving the audio representation results in 13% improvement. We then conduct 3-day simulations with the system model derived from the user study and three real trajectory datasets, including GeoLife [27], SIGCOMM [19], and SF [20]. We implement our CDRR algorithm and two baselines, including Epidemic [25] and CSI [16] for comparisons.

Fig. 2(a) reports that our CDRR algorithm outperforms others by at least 20% in terms of user experience. This is because our CDRR algorithm considers multi-representation



(a)



(b)

Fig. 3. The CTP algorithm completes more requests with higher prediction accuracy: (a) completed request ratio and (b) normalized deviation [12].

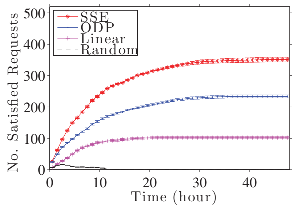
multimedia contents. It can be explained in Fig. 2(b). This figure reports the number of missed units (representations), which indicates that our CDRR algorithm receives at least 10% and 11% more units compared to Epidemic and CSI, respectively. The details can be found in [13].

III. Crowd-sourced fog Computing: Completion Time Prediction

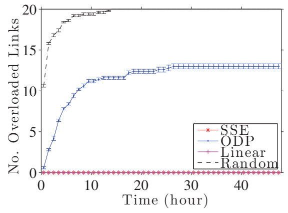
Crowd-sourced fog computing leverages massive fog devices to serve large scale applications, such as animation rendering. In this scenario, animation studios are the fog users who request for animation rendering services. The fog service provider has to give a price quota to animation studios once receiving the requests. The price is highly related to the rendering time. However, the rendering time depends on uncertain user behavior. For example, personal computers may be turned off anytime and each animation rendering job has different complexity, such as number of vertices, polygons, and frames. The uncertainty makes the prediction more difficult. We propose a Completion Time Prediction (CTP) algorithm, which integrates multiple machine learning approaches, such as Support Vector Machine (SVM), Random Forest, and Gradient Boosting Tree.

We conduct extensive simulations using animation rendering and available resource datasets. The animation rendering dataset collects rendering time and features of animations. The available resource dataset collects the fog device availabilities, including CPU and RAM over time. We then implement our CTP algorithm and the state-of-the-art GMM (Gaussian Mixture Model) [10] for comparisons. We also implement Oracle, which results in perfect completion time prediction by peeking into the dataset. We conduct a 7-day simulation with random requests of 30-min poisson arrival rate and the classic Earliest Start Schedule (ESS) scheduler.

Fig. 3(a) reports the completed requests ratio. Our CTP algorithm outperforms GMM algorithm by 30.3% in terms of the completed request ratio. More specifically, our CTP algorithm completes 2,455 more requests, which contains 579,573 frames. This is equivalent to 1,352,337 dollars based on price of a cloud rendering service [4]. That is, our CTP algorithm helps the fog service provider earn 1.3 million dollars more, every week. This can be explained by Fig. 3(b), which shows that our CTP algorithm results in much lower normalized deviation compared to GMM. By comparing the prediction results within 100% normalized deviation, our CTP algorithm outperforms the GMM algorithm by 30%. We report the details of this usage scenario in [12].



(a)



(b)

Fig. 4. Service quality improvement of our SSE algorithm: (a) the number of satisfied requests and (b) the number of overloaded links [14].

IV. Programmable IoT Analytics: Satisfied Requests Maximization

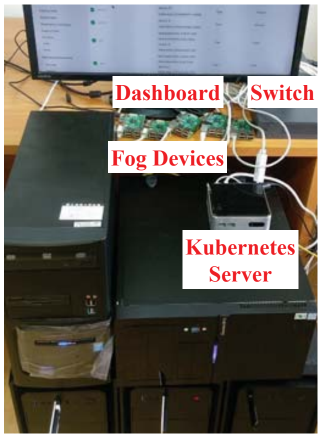
Different IoT analytics, such as plate recognizer and air quality monitor need different sensors and resources. However, running IoT analytics is challenging because of dynamic resource requirements. Hence, we derive system models to accurately predict the required resources and design a heuristic algorithm for deploying analytics. Our algorithm, called SSE is based on three intuitions: (i) **S**carcest resource first, (ii) **S**hortest path first, and (iii) **E**arly feature extraction to maximize the number of satisfied requests.

To derive our system model for the relationship between required resources and sensing frequencies, we conduct measurement studies with three IoT analytics. In particular, we implement an air quality monitor, a sound classifier, and an object recognizer. We derive system models resulting in 0.94 R-square value on average for different kinds of resources, including CPU, RAM, and networks. We use the derived system models to conduct extensive simulations. We implement our SSE algorithm and three baselines, including ODP [9], Linear, and Random for comparisons. We conduct a 48-hour simulation; the requests come with 1-min poisson arrival rate and 10-min poisson departure rate. This results in increasingly more requests over time.

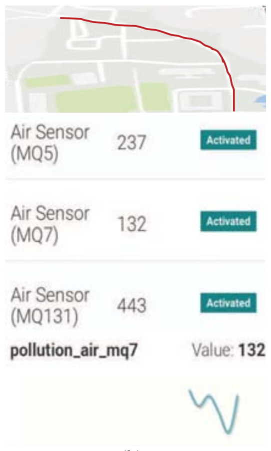
Fig. 4(a) reports number of satisfied requests, which indicates that our SSE algorithm outperforms ODP by 89.4%. Moreover, the Random algorithm cannot satisfy any request after running for 14 hours. This is because ODP and Random do not carefully capture resource constraints, such as network resources of each link as shown in Fig. 4(b). This figure shows that our SSE algorithm and Linear do not overload any links while ODP and Random overload 13 and 21 links, respectively. Although the Linear algorithm does not overload links, it only greedily considers neighboring fog devices without the global view. Hence, our SSE algorithm outperforms the Linear algorithm by up to 171%. The details are in [15].

V. Real Implementation

Implementing the fog computing platform needs to solve several challenges. First, a single fog device, such as Raspberry Pi, may not be powerful enough to launch a complex application. Hence, we have to split applications into smaller *operators* to perform *graph processing*. The concept of graph processing is to represent an application into a graph and perform distributed computing on multiple devices. Second, many applications concurrently run on our platform. Therefore, resource provisioning is a crucial problem to guarantee QoS requirements of the applications. Third, some applications, such as gunshot detection need to be dynamically deployed on different locations at different time. Hence the mechanism of dynamic deployment is also an important issue.



(a)



Air Sensor (MQ5) 237
Air Sensor (MQ7) 132
Air Sensor (MQ131) 443
pollution_air_mq7 Value: 132

(b)

Fig. 5. (a) The testbed of our fog computing platform and (b) a screenshot of the sample air pollution application.

We extend three open source projects, including TensorFlow [5], Docker [1], and Kubernetes [3] to built our platform. TensorFlow is used to implement *multi-operator* applications and perform graph processing. Operators are virtualized by Docker with provisioned resources and managed by Kubernetes for dynamically deploying to fog devices. We implement our testbed with 10 fog devices, including five Raspberry Pis and five i7 personal computers, which is shown in Fig. 5(a). The fog devices are managed by our Kubernetes server. The server and fog devices are connected with a switch under a private network. To emulate different network conditions, we install Wonder Shaper [6] on the fog devices.

The three multi-operator applications are implemented using TensorFlow. Fig. 5(b) shows a sample IoT analytics application, which is an air quality monitor. The application has four quality sensors installed on a box made by Raspberry Pi. The box can be carried by transportation equipments, such as bikes. The application is split into five operators: one operator for collecting sensor data; others for calculating moving average values of four sensors, respectively. The application plots trajectories of transportation equipments and reports moving average values of four air quality sensors overtime. More implementation details are reported in [24].

VI. Related Work and Future Directions

In this section, we discuss characteristics and drawbacks of current fog computing platforms. Several studies [11, 23, 26] use the concept of fog computing [8] in IoT platforms without virtulization. To make the fog computing platform more general and have mobility support, other studies [7, 17, 18, 21] consider virtualization for resource provisioning and dynamic deployment. However, they do not consider graph processing to leverage the power of massive scale fog devices. Hence, Saurez et al. [22] split applications into smaller operators for graph processing, but the authors do not solve optimization problems on their fog computing platform for deploying operators. Cardellini et al. [9] then formulate a flexible operator deployment problem to solve the optimization issues. However, they do not carefully capture the constrained network resources and consider QoS guarantee in the formulation.

In summary, compared to our research, these studies do not consider a realistic ecosystem or implement a complete fog computing platform, which considers: (i) heterogeneous fog devices, (ii) graph processing, (iii) resource provisioning, (iv) optimization problems, and (v) real applications at the same time. To keep improving our fog computing platform, we are collaborating with a company to implement a larger scale fog computing platform. We are also improving our testbed in several directions, such as: (i) Software Defined Networks (SDN) for network resource provisioning, (ii) a migration mechanism for system dynamics, such as link failure, and (iii) a approach for available device prediction.

References

[1] Docker. https://www.docker.com.

[2] Internet of Things (IoT): number of connected devices worldwide from 2012 to 2020 (in billions). https://tinyurl.com/j3t9t2w.

[3] Kubernetes. http://kubernetes.io/.

[4] Shaderlight. http://limitlesscomputing.com/Shaderlight.

[5] Tensorflow. https://www.tensorflow.org.

[6] Wonder Shaper. http://lartc.org/wondershaper/.

[7] P. Bellavista and A. Zanni. Feasibility of fog computing deployment based on docker containerization over RaspberryPi. In *Proc. of ACM Conference on Distributed Computing and Networking (ICDCN)*, Hyderabad, India, January 2017.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proc. of ACM SIGCOMM workshop on Mobile Cloud Computing (MCC)*, Helsinki, Finland, August 2012.

[9] V. Cardellini, V. Grassi, F. Presti, and M. Nardelli. Optimal operator placement for distributed stream processing applications. In *Proc. of ACM International Conference on Distributed and Event-based Systems (DEBS)*, Irvine, CA, June 2016.

[10] T. Estrada, M. Taufer, and K. Reed. Modeling job lifespan delays in volunteer computing projects. In *Proc. of IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, Shanghai, China, May 2009.

[11] K. Giang, M. Blackstock, R. Lea, and C. Leung. Developing IoT applications in the fog: A distributed dataflow approach. In *Proc. of IEEE International Conference on Internet of Things (IoT)*, Seoul, South Korea, December 2015.

[12] H. Hong, J. Chuang, and C. Hsu. Animation rendering on multimedia fog computing platforms. In *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Luxembourg, December 2016.

[13] H. Hong, T. El-Ganainy, C. Hsu, K. Harras, and M. Hefeeda. Disseminating multi-layer multimedia content over challenged networks. *IEEE Transactions on Multimedia*, July 2017. Accept to appear.

[14] H. Hong, P. Tsai, A. Cheng, M. Uddin, N. Venkatasubramanian, and C. Hsu. Supporting internet-of-things analytics in a fog computing platform. In *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Hong Kong, December 2017.

[15] H. Hong, P. Tsai, A. Cheng, M. Uddin, N. Venkatasubramanian, and C. Hsu. Supporting internet-of-things analytics in a fog computing platform. Technical report, 2017. http://tinyurl.com/y9e44max.

[16] W. Hsu, D. Dutta, and A. Helmy. CSI: A paradigm for behavior-oriented profile-cast services in mobile networks. *Ad Hoc Networks*, 10(8):1586–1602, 2012.

[17] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee. A container-based edge cloud paas architecture based on Raspberry Pi clusters. In *Proc. of IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Vienna, Austria, August 2016.

[18] C. Pahl and B. Lee. Containers and clusters for edge cloud architectures–a technology review. In *Proc. of IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Rome, Italy, August 2015.

[19] A. Pietilainen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. MobiClique: Middleware for mobile social networking. In *Proc. of ACM workshop on Online social networks (WOSN)*, Barcelona, Spain, 2009.

[20] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *Proc. of IEEE Communication Systems and Networks and Workshops (COMSNETS)*, Bangalore, India, 2009.

[21] D. Pizzolli, G. Cossu, D. Santoro, L. Capra, C. Dupont, D. Charalampos, F. Pellegrini, F. Antonelli, and S. Cretti. Cloud4IoT: a heterogeneous, distributed and autonomic cloud platform for the IoT. In *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Luxembourg, December 2016.

[22] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwalder. Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In *Proc. of ACM International Conference on Distributed and Event-based Systems (DEBS)*, Irvine, CA, June 2016.

[23] S. Shin, S. Seo, S. Eom, J. Jung, and H. Lee. A Pub/Sub-Based fog computing architecture for Internet-of-Vehicles. In *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Luxembourg, December 2016.

[24] P. Tsai, H. Hong, A. Cheng, and C. Hsu. Distributed analytics in fog computing platforms using Tensorflow and Kubernetes. In *Proc. of IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Seoul, South Korea, September 2017.

[25] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000.

[26] D. Wu, I. Arkhipov, M. Kim, L. Talcott, C. Regan, A. McCann, and N. Venkatasubramanian. ADDSEN: adaptive data processing and dissemination for drone swarms in urban sensing. *IEEE Transactions on Computers*, 66(2):183–198, 2016.

[27] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma. Understanding mobility based on GPS data. In *Proc. of ACM international conference on Ubiquitous computing (UbiComp)*, Seoul, Korea, 2008.