Elastic Urban Video Surveillance System Using Edge Computing

Jianyu Wang
University of Missouri-St. Louis
St. Louis, Missouri
jwgxc@mail.umsl.edu

Jianli Pan
University of Missouri-St. Louis
St. Louis, Missouri
pan@umsl.edu

Flavio Esposito
Saint Louis University
St. Louis, Missouri
espositof@slu.edu

ABSTRACT

During the past decade, the concepts and applications of Internet of Things (IoT) are pervasively propagated to the academia and industries. The widely distributed IoT devices contribute to building an effective smart urban surveillance system, which manages the regular operations and handles emergencies. The real time monitoring uploads massive amounts of data to the backbone network and requires prompt feedbacks. The recent rapid development of "Edge Computing" (also called "Fog Computing" or Mobile Edge Computing in different literature) aims at pushing the computation and storage resources from the remote data center to the edge of network for reducing the burden of backbone and the computing latency. In this paper, we design a three-tier edge computing system architecture to elastically adjust computing capacity and dynamically route data to proper edge servers for the real-time surveillance applications. A system prototype integrating Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) is implemented in an OpenStack based virtualization environment. Moreover, we introduce schemes of resource reallocation and workload balance in urgent situations. Experimental results of the prototype show the great potentials of using edge computing for future large-scale and distributed smart urban surveillance applications.

CCS CONCEPTS

• **Networks** → **Cloud computing**; **Location based services**;

KEYWORDS

Internet of Things, Smart City, Edge Computing, Real-time Surveillance

ACM Reference Format:
Jianyu Wang, Jianli Pan, and Flavio Esposito. 2017. Elastic Urban Video Surveillance System Using Edge Computing. In *Proceedings of SmartIoT'17, San Jose / Silicon Valley, CA, USA, October 14, 2017,* 6 pages.
https://doi.org/10.1145/3132479.3132490

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SmartIoT'17, October 14, 2017, San Jose / Silicon Valley, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5528-5/17/10…$15.00
https://doi.org/10.1145/3132479.3132490

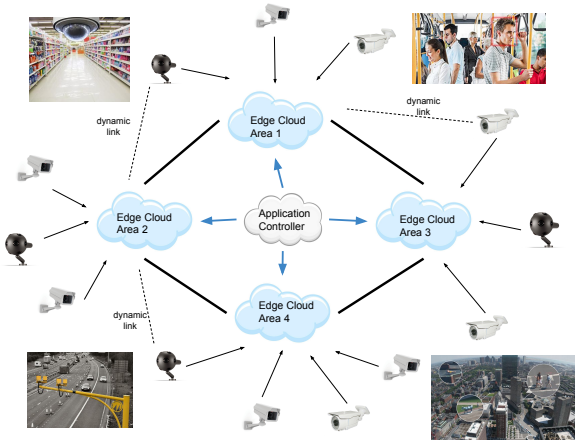

Figure 1: Surveillance applications using edge computing.

1 INTRODUCTION

Urbanization has become a fact for most developed countries and a fast rising trend for developing countries all over the world. According to the urban population data from the United Nations (U.N.), the number of cities that is greater than medium-size will increase from 488 in 2014 to 662 in 2030 [7]. The urban population agglomeration, especially in megacities like New York, Beijing, and Tokyo, are bringing challenges in fighting against ongoing emergencies such as criminal activities. Fortunately, with the advancement of Internet of Things (IoT) and its application in smart cities, a huge quantity of terminal devices could be deployed around the buildings, streets, highways and factories for security surveillance [3]. In the state-of-the-art surveillance system, the distributed monitors capture images and videos and send a large volume of data to the remote servers or data centers. These "big data" from the city surveillance system potentially brings problems including *backbone network congestion, lack of close-by computing power for real-time analysis and decision making, and increasing complexity of system management.* Such traditional architecture is not scalable when the number of the devices and the traffic are constantly increasing. It faces significant challenges for the future smart city applications that require low backbone bandwidth consumption and latency in tracing criminal, recognizing moving cars and other fast reaction tasks.

We envision a better surveillance system empowered by the IoT and edge computing technologies. In such a new system architecture, the large amount of video data do not have to be uploaded through the backbone network and processed remotely. Instead, these real-time video streams are analyzed locally by the nearby edge cloud servers. As illustrated in Fig. 1, the edge clouds deployed in local areas serve various surveillance use cases concurrently

under the management of an application controller. For example, a pedestrian flow detection and prediction application could be deployed to help reduce the occurrence rate of crowd accidents and crisis such as stampede in large festivals and events. The city law enforcement and emergency organizations could get alerts before illegal events, act quickly during incidents and tracking escaped criminals. In short, a smart and well-organized monitoring system with balanced computing resource empowered by the IoT and edge computing technologies would be a priceless assistance for city security.

The main contributions of this paper include:

(1) We propose and build a novel elastic real time surveillance system architecture on the top of a geographically distributed edge cloud platform. In the new system, computing and data processing tasks are carried out by the elastic dynamically launched Virtualized Network Functions (VNFs) on the edge servers instead of deploying expensive, static, and physical Closed Circuit Television (CCTV) servers. Benefits include cost reduction, scalability, flexibility and rich functional components. The edge servers resources can be dynamically allocated and adjusted based on the actual workload of the surveillance application to avoid waste and maximize the utilization efficiency.

(2) The system architecture coherently integrates Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) to guarantee the smooth operation on the hardware resource virtualization and the programmable virtual networks and entities configuration [8]. A group of Virtual Machines (VMs) or VNFs launched in the distributed edge cloud servers work together for a specific surveillance task, and they are configured, monitored, and managed effectively by the SDN controller.

(3) We implement a prototype of the proposed architecture and perform a series of case studies to demonstrate and evaluate the system performance of the elastic edge cloud management and dynamic resources allocation when the system tracks mobile objects that randomly move across different cloud areas.

(4) We demonstrate the feasibility and effectiveness of the proposed system architecture through a series of preliminary experiments.

The rest of this paper is organized as follows. Section 2 briefly introduces the current related work on city monitoring and edge computing. Section 3 is the system design and the model of the proposed elastic urban security surveillance system. Second 4 explains the key implementation processes and issues. Second 5 presents the experiments and analysis results. Finally, the conclusion follows in Section 6.

2 RELATED WORK

With the advancements in IoT and related technologies, more and more city scale surveillance solutions are developed and tested by academia.

The concept of real-time city is introduced by Rob Kitchin [6] drawing researchers' attention on how many digital devices are instrumented in the cities and how the mass of data is utilized to serve the city governance. Considering the data offloading from cameras to the data center, Zhou et al. [12] proposed a meshed network based on the routes of public buses, where the nearest bus stop acts as a node to upload video data. To fully take advantages of video analysis results, Shao et al. [11] constructed a solution to collaboratively monitor ongoing events according to the previous evidence and current information in a spatial-temporal manner.

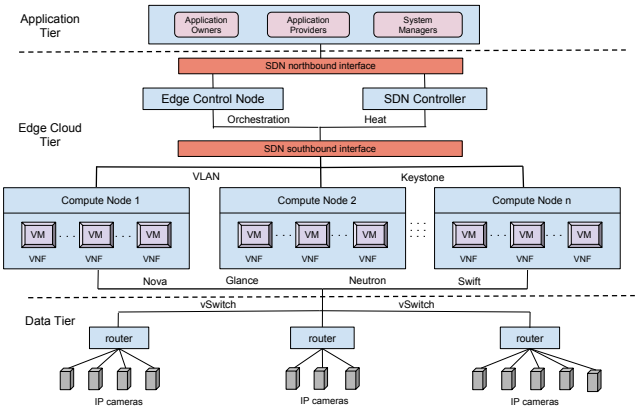

Figure 2: System Architecture.

Facing the challenges of low latency and massive data processing, edge cloud or edge computing shows great potentials for large scale video applications. Two seminal studies about edge computing efforts are: fog computing built on the deployment of self-organized devices like sensors on the network edge [4]; and cloudlet as micro data center distributed based on the specific applications to push computing power close to the customers, such as AR/VR video processing [10]. Moreover, The applications of edge computing are explored under the background of IoT and some use cases are studies to verify its superiority [9]. For example, A UAV surveillance system is designed to performs multi-target tracking by uploading real-time video to the nearest fog node [5].

3 SYSTEM DESIGN

Utilizing edge computing and SDN technologies, our surveillance system design aims at elastically monitoring regular and urgent security events in urban areas with low latency and backbone bandwidth consumption, where the cameras are deployed to form a dense and connected network. Fig. 2 illustrates the system architecture and relations between the functional components. The architecture consists of three tiers from top to bottom: application tier, edge computing tier, and data tier.

The applications tier contains three components. The application owners provide the detailed specification of required resources for data processing and storage of various tasks. The application providers collect information and plan resources allocation and configurations. Meanwhile, the system managers monitor the running status of the applications and deal with the useful feedbacks from the below tiers to determine the following task arrangements.

The edge computing tier include edge cloud nodes and SDN components. Specifically, the SDN northbound interface communicates with upper tier while the southbound interface enables connections between SDN controllers and virtual computing entities (VNFs). The edge cloud consists of a controller node and a number of compute nodes. The VMs or VNFs are loaded on the compute nodes to

provide computing power for the surveillance applications. Moreover, SDN controller configures and manages the VMsâĂŹ flow tables based on OpenFlow protocol to intelligently route the data packets among the physical compute severs.

In the data tier, the terminal monitors collect and dynamically upload real-time video source data to the nearest available compute nodes. Each device is assigned with IP address and carries two attributes to its upper-level server: video resolution and location coordinates. From a global perspective, the density of monitors determines the geographical distribution of the compute nodes since the computation and storage resources on the edge servers should satisfy the requirements of different applications in a specific area.

The working processes of the system can be briefly summarized into the following key steps:

(1) Collecting application requests: For a specific surveillance scenario, the requirements vary on different parameters including video resolution, processing frequency, task location, computing complexity and storage space. The system manager determines and sends the application requests through SDN northbound interfaces.

(2) Translating requirements: After receiving the detailed requests from the upper level, SDN controller translated them to the expression of system behaviors and the corresponding data flow control configurations.

(3) Orchestrating configurations: In addition to the network routing, the edge control node performs resource orchestration to satisfy the requirements of computing power for specific surveillance tasks. The configurations are expressed by templates in the form of text file as executable code to describe the resource assignments including IP addresses, bandwidth volumes, computer node flavors, security group and etc.

(4) Launching VM or VNF instances: The VNF instances are launched on the in distributed compute nodes to flexibly utilize the hardware resources. The parameters for VNFs are retrieved from the orchestration file to specify CPU capacity, memory space and floating IP addresses. Each VNF is presented as an independent computing entity and preloaded with environments for a specific application.

(5) Routing data flow: The network organization of compute nodes and routers follows the SDN routing strategies transmitted from the southbound interface. By implementing the required network topologies, the source data uploaded from the lower tier would be forwarded to the optimal edge nodes through virtual switches (vSwitch).

(6) Capturing video: The source video stream is captured and uploaded by the monitors deployed in each corner of city. Once an urgent event happens, their coordinates would be used to assign new data routing and offloading destinations.

By carrying the above six steps, the real-time surveillance system could concurrently support different monitoring tasks.

Within the architecture design, we have four key designs to enhance the efficiency of surveillance which distinguish our system from other existing solutions:

a. Virutalized computing service provisioning

We use the VMs or VNF instances as the basic units to provide computing services for video surveillance tasks. Video analyzing algorithms run on the VNF instances launched on the physical servers. With such a method, independent workspace can be set up by launching different virtual instances to avoid possible communication interference among different applications such as license plate recognition and face detection. Moreover, through hardware virtualization, a physical machine could be virtualized into a reusable pool of resources such as virtual CPU (vCPU), RAM memory space and bandwidth. During the regular surveillance, applications only take a part of resources. However, reserved computation would be activated to offer a higher performance in emergency mode. In addition, the GuestOS images that are pre-installed with software and execution environment for different kinds of surveillance applications are stored as the snapshots on the compute nodes. Launching from the prepared images reduces the time gaps between the starting and the ready-to-use status.

b. Flexible data flow control

Another key feature by implementing edge computing is flexible data flow control. First, resources can be assigned to the VNFs to exactly suit the specific latency and frequency requirements on data collection and video analysis . Second, when an emergency occurs, the related applications would be pushed for higher priority for resources. The system tries to allocate more computing power and bandwidth by spawning new VMs. Third, the matching relationship between a camera and its edge node is not fixed so that the camera could upload data to another edge server in order to obtain lower latency or computing resource if the original node is overloaded.

c. Dynamic resource orchestration through northbound interfaces

We use the edge cloud control node as the orchestrator to deploy and deliver the surveillance applications on the edge cloud platform. Through the northbound interfaces, the orchestrator turns the high-level service-level agreements (SLAs) among the application owners, application providers, and the application managers into detailed resource allocation schemes and commits them into detailed application implementation. A typical example method is the Heat orchestrator in OpenStack [2]. More advanced orchestrating method such as an "intent" based method would be more effective in implementing the applications at the edge cloud. During the whole lifetime of the surveillance applications deployed in the edge cloud, the orchestrator will closely monitor the application resource status and application topology. The surveillance application can be scaled up or down by the orchestrator depending on the different modes the application needs to work in.

d. Elastic surveillance modes

Considering the various application purposes, our system can work elastically in two modes: normal monitoring mode and emergency surveillance mode. In the normal mode, multiple monitoring tasks work smoothly to complete periodic video processing at a relatively low frequency and event prediction function owns higher priority in the resource pool. However, when an emergent event occurs like tracing criminals, the frame rates of video capturing increase and image analysis for some particular objects turn to high-priority tasks. Meanwhile, the network bandwidth allocation would be reconfigured to boost the performance of event areas.

4 SYSTEM IMPLEMENTATION

In this section, we present the prototype implementation details and discuss its advantages in handling the emergency surveillance situations.

4.1 Edge Cloud Platform

We choose OpenStack as the basic edge cloud platform for the surveillance application. The video surveillance application is deployed and delivered on the edge cloud prototype which controls a cluster of compute, network and storage resources. The OpenStack based Infrastructure as a Service (IaaS) cloud computing platform is open source and rapid-developing, which consists of several key services including computing (Nova), networking (Neutron), identify (Keystone), orchestrating (Heat), imaging (Glance), object storing (Swift), and other extension projects. The edge cloud could be set up effectively with various scales and capabilities using OpenStack.

Our prototype implementation consists of five key OpenStack services, as illustrated in Fig. 2. Specifically, Keystone provides the authentication API to identify the users and VMs to access the system resources. The administrator also obtains rights to launch and modify the virtual network topology after the identity verification. Nova manages the computing resource pool through the virtualization technologies. VMs spawned by Nova support various kernel formats such as qcow2, iso, vdi, docker, etc. Neutron takes responsibility to build no bottleneck, flexible and dynamic network configuration. Swift provides the storage of object file system that can fast distribute data among the compute nodes, such as video file and operating system snapshots. Heat works as an orchestrator to specify and commit resource for the new applications by executing the configuration templates. From the perspective of system architecture, Nova and Neutron client services run on the compute nodes and the other service are administered by the control node. It is worth noting that a dedicated big-data architecture is needed from the application providers to meet the Service Level Agreements (SLAs) of the applications for more advanced surveillance systems with various data types.

4.2 Compute Unit

The Virtual Machines (VMs), as the VNF entities, launched on the edge server provide computing service for video processing, as the basic compute units. Before initiating and spawning a VM from an OS image, the system should assign the stack flavors including bearer network, communication protocols, security group, volumes, storage space and more configurations. Heat is an orchestration engine of OpenStack to gather the flavors and generate templates of a set of configurations. Through Heat-API component, the system administrator could create a stack from a template and launch VMs to be ready for user applications. In addition to stack configuring, image snapshotting is performed to provide ready-to-use compilation environments and effectively reduces VM launching time. To adjust the data flow of a specific VM instance, the Nova flavor could be assigned with bandwidth limits by the traffic shaping tool – tc to modify the inbound and outbound transmission.

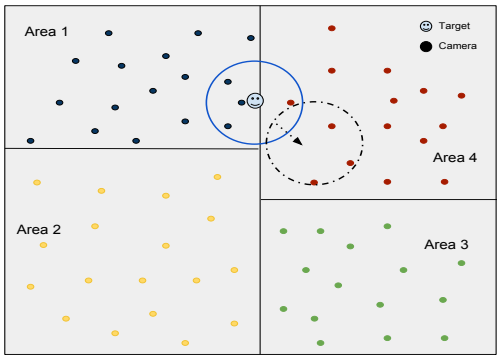

Figure 3: Tracking mobile targets.

4.3 Network Configuration

The network structure is managed by Neutron service which controls the communication protocols, firewall rules, security groups, routing policies and floating ip pools. We construct the network of the surveillance system with VPN technology, where all the VMs for the same tasks are in the same VLAN even if located at different geographical areas. Each VM owns a virtual IP in the VLAN domain and can be associated with a public floating IP to be accessible from the Internet. The network of VMs is called provider network that offers computing service for user applications. Besides the provider network, the connected control node and compute nodes compose the management network, which performs software package installation, DNS, NTP and NAT in order to transmit data packets to the correct VMs and ports.

4.4 Elastic Resource Allocation

In the elastic surveillance application for smart cities, when emergency events happen, the computing resources are reallocated to handle the emergency. Due to the mobility of the objects being tracked, the data offloading and processing should be elastic to ensure that all pieces of evidence or clues are recorded and filtered within acceptable latency. For our surveillance system, a dynamic algorithm is proposed to adjust VM resources to deal with the tracking scenario, as illustrated in Fig. 3.

(1) VMs launching adjustment

In our algorithm, the city is presented as a two-dimension map, where each place has a coordinate. When the police need searching for or tracking a criminal P, the location can be estimated or determined, and denoted by (P_x, P_y). The real-time object tracking needs the collaboration of nearby monitors in the cycle coverage with the center point (P_x, P_y) and the radius R. Moreover, we add a margin R_p to the radius to compensate the service delay because of VM launching time. In this case, not only the launching delay, but also coordinate error is corrected. It is worth noting that the value R_p should be in direct proportion to the movement speed of the moving targets. In the emergency surveillance area, the involved n monitors are denoted by $C_{1:n} = \{C_1, C_2, ..., C_n\}$ with the dynamically changing value n. Correspondingly, the required computing amounts for monitors are $V_{1:n} = \{V_1, V_2, ..., V_n\}$. In addition, we hold a list of m edge nodes $E_{1:m} = \{E_1, E_2, ..., E_m\}$. Assuming that N_i monitors are located in the service space of the edge node E_i, the total required

amount of computing resource is: $W_{tot-i} = \sum_{k}^{N_i} (C_k \times V_k)$. The current available resource of E_i is W_{cur-i}, so that the system still need:

$$W_{req-i} = W_{tot-i} - W_{cur-i} = \sum_{k}^{N_i} (C_k \times V_k) - W_{cur-i} \quad (1)$$

Then assuming the computing power provided by one VM is V_{vm}, the number of VMs to be launched on the node E_i is:

$$N_{launch} = \frac{W_{req-i}}{V_{vm}} = \frac{\sum_{k}^{N_i} (C_k \times V_k) - W_{cur-i}}{V_{vm}} \quad (2)$$

In the case that $W_{tot-i} \leqslant W_{cur-i}$, the system owns enough power to meet the current requirements. N_{term} VMs could be terminated to release resources for other applications:

$$N_{term} = \frac{W_{cur-i} - W_{tot-i}}{V_{vm}} \quad (3)$$

Whenever the system launches new VMs or terminates VMs, such decision is determined by the tracking location and the number of related monitors: (P_x, P_y), $N_i \Rightarrow N_{launch}$ or N_{term}.

(2) Edge nodes work balance

Because of the limit of the available resource pool including vCPU, RAM and storage, the requests to launch new VMs on the edge node may not be satisfied. In this case, a part of video data should be uploaded to the nearby second-suitable edge node. Based on the node list $E_{1:m}$, we maintain a list of the communication delay between the monitor C_i and the edge nodes: $L_{i-1:m} = \{L_{i-1}, L_{i-2}, ..., L_{i-m}\}$. When the resource shortage occurs, some workload of the monitors initially bound to E_i would be transferred to another node with the lowest delay. In order to guarantee the effectiveness of the surveillance system, the monitors located away from tracking center (P_x, P_y) are chosen based on the order of their distances from far to near, denoted by the Euclidean distances:

$$D(P, C_i) = \sqrt{(P_x - C_{i-x})^2 + (P_y - C_{i-y})^2} \quad (4)$$

where (C_{i-x}, C_{i-y}) is the monitor coordinate.

If the required computing resource decreases because of the object mobility, the system cancels the workload transfers and the original nodes will continue serving the applications.

5 EXPERIMENTATION AND RESULTS

To validate and evaluate the proposed system architecture and prototype implementation, we conduct a series of experiments.

5.1 Experiment Testbed

The experiment testbed is built with four desktops, one router, one switch, a number of Raspberry Pi 3 Model Bs with camera modules v2. In the testbed, one desktop acts as the control node that manages the other three desktops as the computer nodes. The desktop configurations are Intel i7 quad-core processor 3.2GHz, 16GB RAM, 1TB storage and two NICs with Ubuntu 16.04 LTS operating system. The router is used to create VLAN for the communication between desktops and offers accesses via NAT method to divide the experimental network into two parts: management intranet for and provider extranet. Meanwhile, the desktops and router connect to the provider network via the switch to obtain internet service.

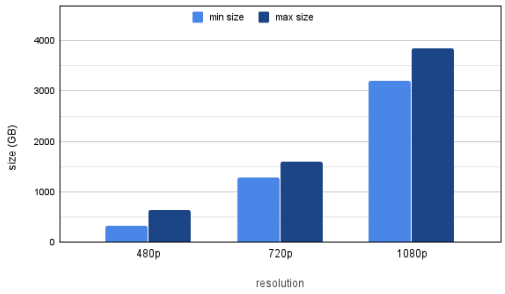

Figure 4: Data flow size per second.

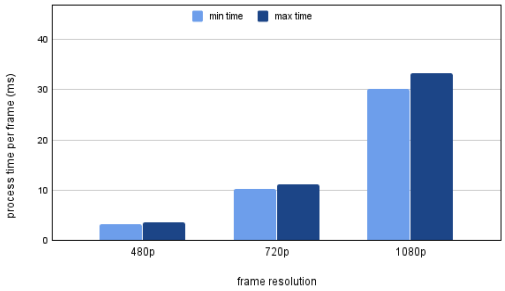

Figure 5: Processing speed for one video frame.

The video sources are provided by the Raspberry Pi and the camera modules installed outside the buildings, whose video solution is up to 1080P (1920x1080). With all the devices, we deployed OpenStack with Newton version, which is the currently second newest and relatively mature. On the each compute node, we launched three VMs with public floating IPs for Raspberry Pi accessing.

5.2 Tests and Results

(1) Data volume

The volume of video data uploaded to the edge servers are estimated using camera modules to record videos of different resolutions in the common city scenes. The real-time video is simulated so that the Pi camera uploads 16 frames of video per second to its VM server. Assuming that the number of monitors deployed in a typical big city like London is about 200,000, we estimate the data size created in such a city per second for different resolutions, as illustrated in Fig. 4.

The results show that thousand Gigabytes (GBs) of data for a city-level application would flood into the network under the future IoT and smart city environment. In this case, edge computing could effectively prevent the congestion of backbone network and process data at local areas.

(2) Data Processing and Transmit Latency

A face recognition application is implemented with OpenCV 3.2 [1] to test the processing speed of the video frames sent to the VM. The recognition contains two steps: (1) detect the faces in the image; (2) compare each face with the datasets already trained by the face images of our target persons. The face detection is performed by utilizing the Haar Cascade classifiers and the comparison by LBPH face recognizer. The VM for testing is configured with four vCPUs, 4G RAM and 100 GB storage, while the Pi camera could capture 16

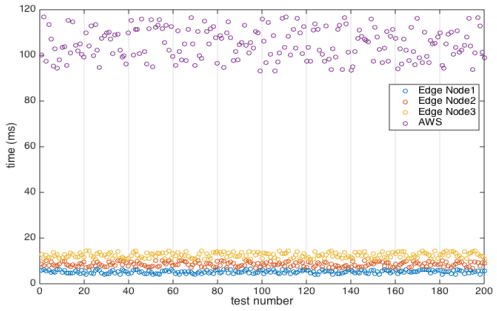

Figure 6: Data propagation round-trip time to edge servers and AWS.

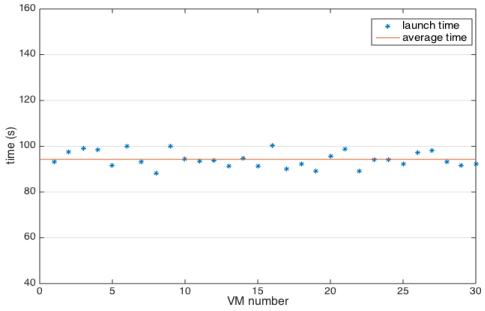

Figure 7: The distribution of VMs launching time.

frames per second. Fig. 5 shows the execution time to process one frame recorded in different resolutions. Because of the resolution difference, the process time increases rapidly from average 3.5 ms to 32 ms with the increment of data size. However, such a performance gap can be reduced by implementing more advanced algorithms.

Besides the processing time, the edge node deployed on the university campus near the cameras could also save the propagation time compared to the central cloud method. The edge node1 is the nearest server to the experimental camera while the node3 is the farthest one. For comparison, we test the average communication delays to four Amazon AWS servers: US-East (Virginia), US-East(Ohio), US-West(California) and US-West(Oregon). The experimental results, as shown in Fig. 6, verifies that edge computing provides highly responsive cloud services for city surveillance with low end-to-end latency and low jitter.

(3) Elastic Response

Since the workload of the user applications varies over time and the analyzing requirements change in an emergency event, the computing resources should be elastically adjusted quickly and smoothly based on the task loads. We create an Operating System (OS) snapshot of the ready-to-use OpenCV environment, which loads the complete basic libraries and contribute libraries, and its size is 7.8 GB. Then thirty VMs are launched concurrently to meet the handle the urgent requests. Fig. 7 demonstrate that the VM launch times are in minutes level ranging from 1 to 2 minutes. The launching speed is acceptable for boosting the emergency overloaded requests in short time.

6 CONCLUSION

In this paper, we proposed an IoT based elastic surveillance system using Edge Computing paradigm to perform data processing near the IoT devices in the smart city applications. A prototype for evaluating practical environment was implemented on a lightweight OpenStack platform and VMs serves as the unit of computing. The achievements of the elastic administration and resource allocation are sustained by the geographically distributed computing nodes and the reliable control node. We also discussed the methods for adjusting system performance and balancing workload to meet the regular and emergent requirements. The experimental results demonstrated that the system is rapid, responsive, flexible, scalable and easily configurable.

ACKNOWLEDGMENTS

This work has been partially supported by a University of Missouri Research Board (UMRB) award and the National Science Foundation award CNS-1647084.

REFERENCES

[1] 2016. OpenCV. (2016). http://opencv.org/opencv-3-2.html [Online resource], available at: http://www.openstack.org/.

[2] 2016. Openstack: free and open-source software cloud computing platform. (2016). http://www.openstack.org/ [Online resource], available at: http://www.openstack.org/.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials* 17, 4 (Fourthquarter 2015), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/2342509.2342513

[5] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann. 2016. Dynamic Urban Surveillance Video Stream Processing Using Fog Computing. In *2016 IEEE Second International Conference on Multimedia Big Data (BigMM)*. 105–112. https://doi.org/10.1109/BigMM.2016.53

[6] Rob Kitchin. 2014. The real-time city? Big data and smart urbanism. *GeoJournal* 79, 1 (01 Feb 2014), 1–14. https://doi.org/10.1007/s10708-013-9516-8

[7] United Nations. 2014. 2014 Revision of World Urbanization Prospects. (2014). https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Highlights.pdf

[8] J. Pan, L. Ma, R. Ravindran, and P. TalebiFard. 2016. HomeCloud: An edge cloud framework and testbed for new application delivery. In *2016 23rd International Conference on Telecommunications (ICT)*. 1–6. https://doi.org/10.1109/ICT.2016.7500391

[9] M. Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (Jan 2017), 30–39. https://doi.org/10.1109/MC.2017.9

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (Oct 2009), 14–23. https://doi.org/10.1109/MPRV.2009.82

[11] Z. Shao, J. Cai, and Z. Wang. 2017. Smart Monitoring Cameras Driven Intelligent Processing to Big Surveillance Video Data. *IEEE Transactions on Big Data* PP, 99 (2017), 1–1. https://doi.org/10.1109/TBDATA.2017.2715815

[12] W. Zhou, D. Saha, and S. Rangarajan. 2015. A System Architecture to Aggregate Video Surveillance Data in Smart Cities. In *2015 IEEE Global Communications Conference (GLOBECOM)*. 1–7. https://doi.org/10.1109/GLOCOM.2015.7417602