Policy-Based Function-Centric Computation Offloading for Real-Time Drone Video Analytics

Dmitrii Chemodanov†, Chengyi Qu†, Osunkoya Opeoluwa†, Songjie Wang, Prasad Calyam

Email: {dycbt4, cqy78, osoykp}.mail.missouri.edu, {wangso, calyamp}@missouri.edu; University of Missouri-Columbia, USA.

***Abstract*—Computer vision applications are increasingly used on mobile Internet-of-Things (IoT) devices such as drones. They provide real-time support in disaster/incident response or crowd protest management scenarios by e.g., counting human/vehicles, or recognizing faces/objects. However, deployment of such applications for real-time video analytics at geo-distributed areas presents new challenges in processing intensive media-rich data to meet users' Quality of Experience (QoE) expectations, due to limited computing power on the devices. In this paper, we present a novel policy-based decision computation offloading scheme that not only facilitates trade-offs in performance vs. cost, but also aids in offloading decision to either an Edge, Cloud or Function-Centric Computing resource architecture for real-time video analytics. To evaluate our offloading scheme, we decompose an existing computer vision pipeline for object/motion detection and object classification into a chain of container-based micro-service functions that communicate via a RESTful API. We evaluate the performance of our scheme on a realistic geo-distributed edge/core cloud testbed using different policies and computing architectures. Results show how our scheme utilizes state-of-the-art computation offloading techniques to Pareto-optimally trade-off performance (i.e., frames-per-second) vs. cost factors (using Amazon Web Services Lambda pricing) during real-time drone video analytics, and thus fosters effective environmental situational awareness.**

***Index Terms*—Cloud/fog Computing, function-centric computing, drone video analytics, computation offloading policies.**

I. Introduction

In the last few years, autonomous Unmanned Aerial Vehicles (UAVs), also known as drones, have been widely used in a large number of scenarios. The scenarios range from urban and rural area control for prevention of crime, rescue operations, traffic surveillance, and forest fire monitoring. Most commercially used drones are embedded with a high-resolution camera and used in surveillance systems to visualize and monitor target environments, e.g., for tracking purposes.

Recent advances in computer vision and drone-based technologies has enabled additional use cases, e.g. motion detection and object classification from drone-sourced video streams. However, visual data processing involves computation-intensive analysis of video streams, especially when video is of high-resolution and needs to be analyzed and matched against a large/distributed database of images. Obviously, such a processing thus requires high-performance computing (HPC) and adequate energy sources. It is not practical for a typical drone system to handle these critical computing/energy needs if visual data processing is performed on-board. Thus, there is a need for an efficient computation offloading to enable fast and accurate video analytics, e.g., for object tracking and classification tasks.

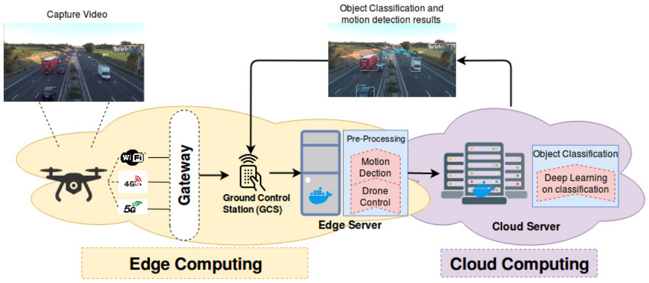


Fig. 1: Overview of the computation offloading scheme on drone-based video analytics

Figure 1 shows an overview of a common computation offloading scenario in a drone-based video surveillance system. In this air-to-ground scenario, a programmable drone is connected to a ground control station (GCS) at the infrastructure gateway through a wireless network e.g., based on Wi-Fi, 4G/LTE or 5G technologies. The GCS itself is connected to the edge/core cloud infrastructure that features HPC resources such as GPUs. Thus, the GCS can offload computations to either Edge or Cloud servers to analyze drone video streams. However, if inadequate HPC resources are available at the Edge server and a low-latency communication between GCS and the processing site is needed (e.g., for real-time drone control), new approaches need to be devised. Based on our literature survey [1], there is a lack of mechanisms to efficiently coordinate between the limited Edge computing resources and Cloud HPC resources during real-time video stream processing.

In this paper, we present a novel *policy-based computational offloading scheme* that facilitates trade-offs in performance vs. cost factors (based on users' preferences) during real-time drone video analytics by utilizing Edge, Cloud and Function-Centric Computing architectures. The Function-Centric Computing (FCC) architecture features decomposing applications into microservice functions that can be deployed onto edge resources in conjunction with core cloud platforms.

†These authors contributed equally to this work.

This material is based upon work supported by the National Science Foundation under Award Number: CNS-1647182 and the Army Research Lab under Award Number: W911NF1820285. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Army Research Lab.

978-1-7281-1434-7/19/$31.00 © 2019 IEEE

The contributions of this paper can be summarized as follows:

- We describe a drone video analytics application that supports Function-Centric Computing by decomposing our video analytics pipeline into a chain of microservice functions, and deploying the functions onto a mixture of Edge/Cloud computing resources communicating via a RESTful API.
- We detail a novel offloading scheme that uses policy-based decision making mechanism and supports Function-Centric Computing strategy[1] to (Pareto) optimally trade-off performance vs. cost factors among the different computing architectures, with the purpose of maximizing performance while reducing costs, and thus to meet different users' QoE requirements.
- We deploy an edge/cloud testbed that spans multiple geographical locations and features HPC cloud resources by utilizing GENI [2] and CloudLab [3] infrastructures to evaluate different offloading strategies under realistic application settings.

The rest of the paper is organized as follows: Section II presents related work. In Section III, we discuss how our computer vision application case study can be decomposed into a service chain. In Section IV, we detail our novel policy-based function-centric computational offloading scheme for real-time drone video analytics. Section V describes our testbed-based evaluation methodology, performance metrics and results. Section VI concludes the paper.

II. Related Works

Drones Computation Offloading. Computation offloading is widely used to overcome insufficient local capabilities of resource-constrained (e.g., mobile) devices such as drones [4]. Computation offloading allows users to delegate jobs fully or partially to a server, minimizes local resource utilization (e.g., to save energy) and reduces time to obtain results. Thus, the authors in [5] propose a simple decision algorithm that offloads parts of the computation workload to a remote server. The offloading is performed if the execution cost of the computation operation is greater than the execution cost of the offloading mechanism. Although, authors in [5] outline a dynamic decision making scheme for the air-to-ground scenario (from drones to ground control stations), they do not capture interplay between multiple factors such as cost, performance and others under realistic cloud infrastructure settings [6]. Some of the recent approaches for drone video analytics propose borrowing computing resources opportunistically from other nearby drone clusters [7], [8]. However, due to constrained drone resources (i.e., energy, compute, human operations, etc.), offloading to the unconstrained infrastructure sites e.g., by using Edge or Cloud computing paradigms can be a more efficient way in common use cases.

In our work, we assume a scenario when drones lack the ability of processing functions with HPC demands and stream video directly to the ground control station via a wireless network connection. The ground control station subsequently is in charge of selecting an appropriate offloading strategy, i.e., Edge, Cloud or Function-Centric Computing.

Real-time Drone Video Analytics. Real-time video analytics is frequently applied for camera-equipped mobile devices. For example, the authors in [9] use Edge or Cloud computing for a face recognition by analyzing imagery data from mobile phones or Google Glass devices. To improve video analytics performance in terms of latency and reduce network bandwidth demands, the authors in [10] cache few video frames on a mobile device while sending triggered frames to a remote server for recognizing and labeling. Recently, a similar approach has been also applied for drone-based systems where the authors in [11] propose an adaptive video streaming and content-aware compression to satisfy real-time drone video analytics requirements. In the case when video analytics is offloaded from the ground control station, the authors in [12] show how Edge computing can greatly reduce drones' network bandwidth demands. Another proposed approach achieves a low-latency video analytics performance by placing drone ground control stations in heterogeneous networks [13].

In contrast, our approach considers a possible decomposition of a real-time drone video analytics pipeline into individual object motion detection and its consequent classification functions. Leveraging this decomposition outcome, we apply a novel policy-based function-centric computational offloading scheme that facilitates trade-offs in cost and performance factors for real-time drone video analytics.

III. Function Chain for Classification of Moving Objects

In this section, we introduce our computer vision application that is used for the real-time drone video analytics and can support our Function-Centric Computing approach in [1].

A. Overview

Our application performs the initial objects motion detection (that can be also used for the drone real-time control) and their consequent classification. The result of this analytics is shown in Figure 2. To implement the function-centric pipeline, we decompose our application into microservices encapsulated with Docker containers and communicating via a RESTful API. The microservices are implemented using the python Flask package. The microservices expose OpenCV and TensorFlow functions that carry out the processing required for the application to complete execution. The application resides within a Docker container that allows it to be quickly and easily deployed on diverse nodes. Thus, we take advantage by deploying the same container across all the nodes in our infrastructure. This also allows us to independently execute specific application functions at desired locations to evaluate different computational offloading strategies later in Section V.

[1] Note that this strategy provides an additional offloading option to ensure a cost-effective and performance-optimized solution for certain policy requirements in drone video analytics.

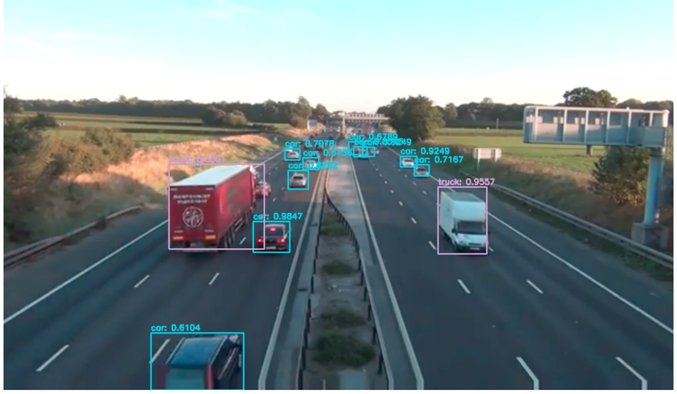


Fig. 2: Example of the moving object classification results on the ground surveillance video from the drone.

B. Implementation Details

Our overall drone video analytics pipeline is shown in Figure 3 and has the following RESTful API endpoints:

The classifier endpoint. This endpoint (*/objectClassifier*) features a YOLO v3 deep learning classifier running on TensorFlow that was pre-trained on COCO [14]. It receives a frame from an image as input and is able to quickly carry out object recognition on it and return bounding boxes and labels for all the objects that exist in the frame.

The frame processing endpoint. This endpoint (*/frameProcessing*) is responsible for extracting the sections of an image that changed between succeeding frames in a video using classic computer vision techniques. This endpoint combines the output of the classifier (which is able to identify all the objects in a frame but cannot tell those that moved between frames) with its output by correlating the areas they both identify. This allows it to only count objects whose positions have changed from preceding frames, but also allows for labeling them accordingly.

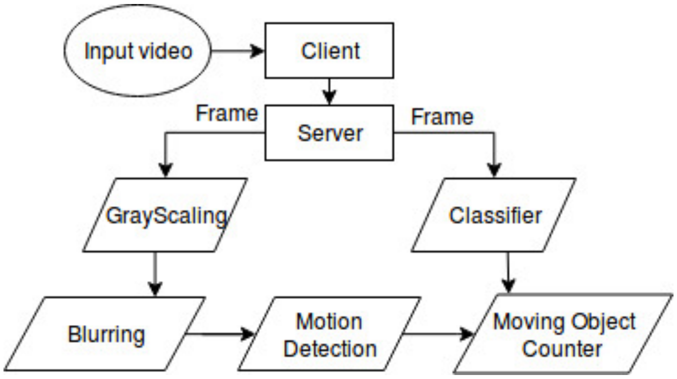


Fig. 3: Image processing pipeline to investigate computation offloading.

The counts endpoint. This endpoint (*/getCounts*) serves as an access point to view the outputs returned by the classifier and frame processing endpoints.

The function chain definition endpoint. This endpoint (*/setNextServer*) is used to dynamically determine which server is called to execute the next function. This allows for programmatic access to define where the classifier is executed since all infrastructure nodes support the same container image.

IV. Policy-Based Function-Centric Computational Offloading Scheme

In this section, we introduce our novel policy-based decision offloading scheme that allows us to not only facilitate trade-offs in performance vs. cost factors of real-time drone video analytics, but also aids in decisions pertaining to the pertinent data computation architecture, i.e., by selecting either edge, cloud or function-centric computing for data processing. Note that by *Edge computing* we assume data processing at the server that has a low-latency connection to the ground control station, and is sufficient to handle real-time drone control operations - usually with latency of $\leq$5 ms [15].

A. Function-Centric Offloading: Policies and Algorithm

In our investigation scenario, a drone continuously flies to capture video streams with a desired video resolution using either a pre-configured route or via remote control. At the same time, the drone streams the captured video to a ground control station via a wireless network for real-time video analysis as shown in Figure 1. We assume that the ground control station needs to offload computations as it lacks sufficient local HPC resources, such as GPU resources to classify moving objects by using deep neural networks. In the following, we first describe our supported policies, and then detail our computational offloading algorithm implementation.

Policies. The supported policies that we consider include: (i) the *real-time control* policy that requires a portion of the video analytics corresponding to the remote control of drones is performed at the edge with low-latency; (ii) the *cost/performance preference* policy allows users to specify whether they want to pay more for higher performance or not; and (iii) the *function-centric availability* policy ensures that a video analytics application is function-based and its individual functions can be executed at different locations.

Algorithm. To offload computations, the ground control station executes our policy-based function-centric scheme shown in Figure 4 and outlined in Algorithm 1.

We start by checking if HPC resources are available at the Edge (line 4). Thus, if Edge computing doesn't introduce any bottleneck for the video analytics, and if the drone real-time control $P.real\_time$ is not required, we use $P.preference$ to decide between offloading to the Edge or to the Cloud. This is because due to a lower latency we can expect slightly higher performance by analyzing video at the Edge (line 9), which is however more expensive than processing in the Cloud (line 6) based on the common (serverless) pricing models such as AWS Lambda [16].[2]

At the same time, if no HPC resources are available at the Edge (i.e., Edge computing is a bottleneck) and the

[2] Due to the mobility issues and limited power supply of drones in flight, we assume in our work that the serverless pricing model is more beneficial than its server-based counterpart for (short-term) video analytics tasks.

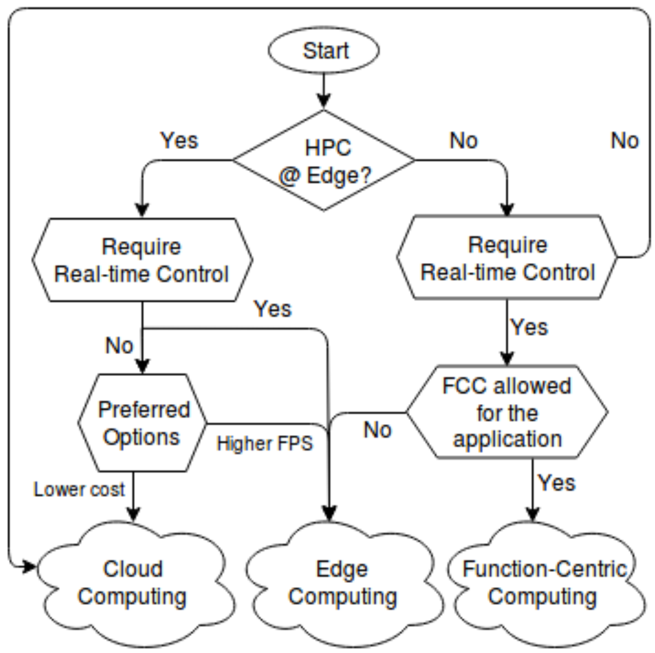

Start
HPC @ Edge?
Yes
No
No
Require Real-time Control
Require Real-time Control
Yes
No
Yes
Preferred Options
FCC allowed for the application
Higher FPS
No
Lower cost
Yes
Cloud Computing
Edge Computing
Function-Centric Computing

Fig. 4: Illustration showing our policy-based function-centric computation offloading scheme for real-time drone video analytics.

Algorithm 1: Policy-based Function-Centric Offloading

```
Input: $Video$:= video data to analyze; $P$:= set of policies; $Edge$:= IP:Port;
       $Cloud$:= IP:Port; $Edge_{HPC}$:= 1 if Edge HPC is available, 0 otherwise
Output: $URI$:= offloading RESTful API URI; $Data$:= JSON data to offload
1  begin
2      $URI \leftarrow$ http://
3      $Data \leftarrow Video$
4      if $Edge_{HPC} == 1$ then
5          if $P.real\_time == 0$ and $P.preference == cost$ then
6              $URI \leftarrow URI \cup Cloud$
7          end
8          else
9              $URI \leftarrow URI \cup Edge$
10         end
11         $URI \leftarrow URI \cup$ /fullVideoProcessing
12     end
13     else
14         if $P.real\_time == 0$ then
15             $URI \leftarrow URI \cup Cloud \cup$ /fullVideoProcessing
16         end
17         else if $P.fcc\_availability == 1$ then
18             $URI \leftarrow URI \cup Edge \cup$ /functionCentricProcessing
19             $Data \leftarrow Data \cup \{$"NextFunction" : $Cloud\}$
20         end
21         else
22             $URI \leftarrow URI \cup Edge \cup$ /fullVideoProcessing
23         end
24     end
25 end
```

real-time control $P.real\_time$ is required, we check whether the application for video analytics can be performed using function-centric computing $P.fcc\_availability$ (line 17) to improve both its performance and the cost w.r.t. the resource limited Edge computing. This is due to the fact that function-based processing depends on both processing data size and the time it takes to process this data [16]. Hence, by interfacing Edge and Cloud computing in this case, we can sufficiently speed up the data processing at the Edge and reduce the total cost.

Remark. Algorithm 1 uses IP addresses and ports of the corresponding video analytics services running at the Edge and Cloud servers. However, in the case of having multiple Edge/Core clouds, our algorithm can be used in conjunction with other existing schemes to optimize either *Edge* [17], *Cloud* [18] or both (e.g., simultaneously via use of service function chaining [19]) server selections.

V. Performance Evaluation

In this section, we evaluate the different computational offloading strategies experimentally and show how our proposed scheme allows to (Pareto-)optimally trade-off performance vs. cost factors of the real-time drone video analytics. Specifically, we evaluate benefits of our policy-based function-centric computation offloading scheme for real-time drone video analytics to classify moving objects as discussed in Section III. To this aim, we outline our geo-distributed edge/core cloud testbed setup that uses GENI [2] and CloudLab [3] infastructures and features function-centric computing capabilities.

Experimental Setup. In our experiments, we use the ground surveillance video of VGA resolution (640 x 480) to analyze and classify moving objects (e.g., cars, trucks, etc.) observed by the drone as shown in Figure 1. The example of a processed video frame is shown in Figure 2, and both our application and the video itself are publicly available at [20].

Our geo-distributed edge/core cloud testbed setup includes 1 virtual machine (VM) and 1 server reserved in the GENI rack at the Missouri InstaGENI site as shown in Figure 5, 1 VM reserved in the GENI rack at the Wisconsin InstaGENI site and finally 1 server with HPC capabilities reserved at the CloudLab Wisconsin site.

Both VMs have 1 core CPU, 1 GB RAM and emulate drone ground control stations with limited computation capabilities that accept video streams from drones and decide on their computational offloading strategy. Our server at Missouri Instageni has 12 cores Intel Xeon CPU and 16 GB RAM and acts as an Edge server without HPC capabilities for the ground control station at the Missouri site. Our server at CloudLab Wisconsin has 2 X 20 cores Intel Xeon Silver CPUs, 192 GB of EEC RAM and 12 GB NVidia Tesla P100 GPU and acts both as a Cloud server for the ground control station at the Missouri site and as an Edge server with HPC for the ground control station at the Wisconsin site. Finally, both servers feature Docker containers and allow us to execute a particular video analytics function without running an entire video analytics pipeline described in Section III.

Comparison Methods and Metrics. We compare performance of our moving object classifier using 3 types of computational offloading strategies such as: common cloud computing (*Cloud HPC*), novel edge computing with and without HPC capabilities (*Edge/Edge HPC*) and our function-centric computing (*FCC*) [1]. As our application is implemented using Docker containers, we can execute both the application functions at one place i.e., use either edge or cloud

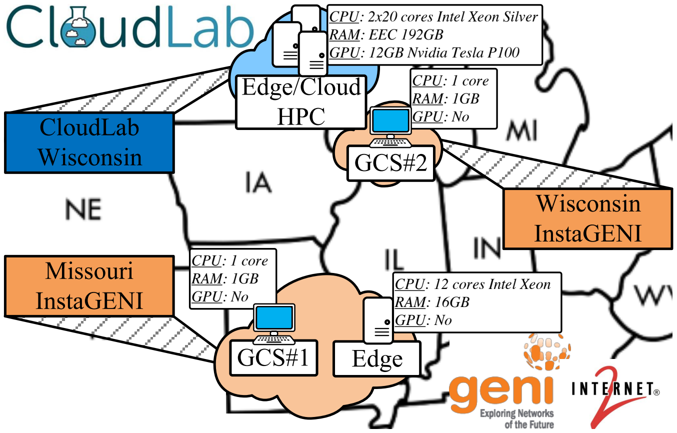


Fig. 5: The geo-distributed edge/core cloud testbed for drones ground control stations (GCSs) provisioned in GENI and CloudLab platforms that spans Missouri InstaGENI, Wisconsin InstaGENI and CloudLab Wisconsin sites.

server, or each function at different servers i.e., use both edge and cloud servers. Our ground control station first decides on the offloading strategy and then sends RESTful API calls to corresponding server location(s). We remark that - to test the scenario of checking whether the Edge server is capable of HPC, we move our ground control station to the Wisconsin site that is geographically close to our cloud server.

We measure the performance of each of these computational offloading strategies by using a 'processed Frames-Per-Second' (FPS) metric (the higher the better). Moreover, we use the 'round-trip-time' (RTT) metric (the lower the better) to access the ability of the chosen offloading strategy to perform real-time control of the drone e.g., to follow road traffic as well as its cost as described below. Finally, to assess the frame processing cost (the lower the better) of each offloading strategy such as edge computing, cloud computing and FCC, we use AWS Lambda pricing model [16] that accounts for product of processing data size and the time of its processing, as shown in Equations 1, 2 and 3, respectively.

$$C_{edge} = \frac{w_i \cdot 6.25125 \cdot 10^{-6}}{128 \cdot 1024} \cdot (\frac{1}{fps_i} - rtt_{edge}) \tag{1}$$

$$C_{cloud} = \frac{w_i \cdot 16.67 \cdot 10^{-6}}{1024 \cdot 1024} \cdot (\frac{1}{fps_i} - rtt_{cloud}) \tag{2}$$

$$\begin{aligned} C_{fcc} = & \frac{w_i \cdot 6.25125 \cdot 10^{-6}}{128 \cdot 1024} \cdot (\frac{1}{fps_i} - rtt_{edge}) + \\ & + \frac{w_i \cdot 16.67 \cdot 10^{-6}}{1024 \cdot 1024} \cdot (\frac{1}{fps_i} - rtt_{edge} - rtt_{cloud}) \end{aligned} \tag{3}$$

where w_i is a size of frame i in KB, fps_i is a frame rate at which frame i was processed and $rtt_{edge/cloud}$ is the RTT between the ground control station and the corresponding edge/cloud server.

Remark. We refer to an offloading strategy to be *Pareto-optimal* if there is no other alternative offloading strategy that makes any one factor (e.g., cost or performance) better off without making at least one factor worse off.

(i) Edge/Cloud HPC are Pareto-optimal choice w.r.t. cost and performance factors if no real-time control is needed. While observing Figure 6b, we can notice how $Cloud$ computing cannot be used in situations when we use drone video analytics for its real-time control due a high RTT ($\geq$ 20ms). However, when we are not concerned about the drone real-time control, we can see from Figure 7b how both $Cloud$ and $Edge$ HPC are choice #1 w.r.t. cost and performance factors. This is due to the fact that $Edge$ HPC provides slightly higher performance than $Cloud$ computing due to an order of magnitude lower RTT, but computing at the $Edge$ costs more (see Equations 1 and 2). Thus, both $Cloud$ and $Edge$ HPC are Pareto-optimal choice w.r.t. cost and performance factors.

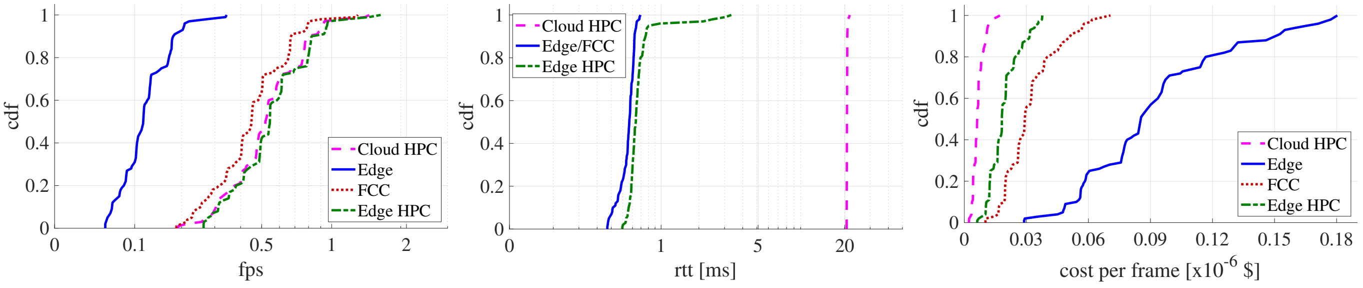
(ii) Function-centric computing is superior than Edge computing in terms of both performance and cost factors if no Edge HPC is locally available. We can observe that - when HPC is not available at the $Edge$ and real-time drone control is needed[3], function-centric computing (FCC) becomes the #1 choice when allowed by the video analytics application. Figure 7b shows how by interfacing $Edge$ with $Cloud$ computing, FCC can ~4x speed-up the former that has limited resources. As a result, FCC is also ~3x cheaper than $Edge$ computing despite the fact that it uses both $Edge$ and $Cloud$ resources (see Equation 3). 0 This can be also seen from Figures 6a and 6c.

(iii) Our policy-based function-centric computational offloading scheme Pareto-optimally trade-offs performance vs. cost factors of the real-time drone video analytics. While observing our policy-based offloading scheme performance (see Section IV), we can make several conclusions. First, if we are not concerned about processing costs, our scheme first selects $Edge$ HPC strategy (when available), then $Cloud$ or FCC (based on real-time control requirements). On the contrary, when we are concerned about processing costs, our scheme first selects $Cloud$ computing strategy (if it is feasible), then $Edge$ or FCC (based on $Edge$ HPC availability). Finally, when no real-time control is needed, our scheme always selects either the $Cloud$ or $Edge$ HPC strategy based on users' cost/performance preferences – note that both are Pareto-optimal w.r.t. these factors. Further, we can observe how our offloading scheme choices are in line with our experimental results shown in Figures 7a, 7b and 7c.

VI. Conclusion

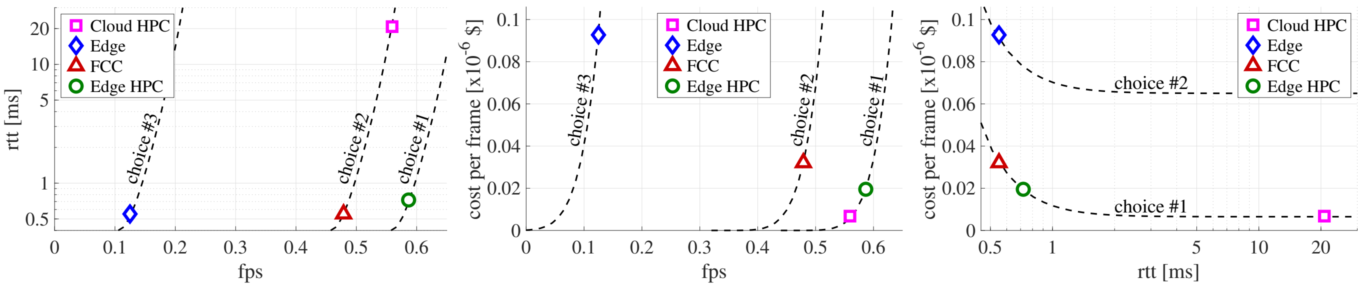
In this paper, we present a novel policy-based computational offloading scheme to address the challenges in processing video streaming data collected on drones. Our scheme features the choice of Edge, Cloud and Function-Centric Computing architectures. Using a realistic geo-distributed edge/core cloud testbed, we show how our proposed scheme enables users to decide which architecture is superior to Pareto-optimally trade-off performance vs. cost factors in real-time drone video analytics tasks.

[3] Note that the real-time drone control usually depends on fast pre-processing steps such as motion detection (see Section III), and hence can be still processed at the $Edge$ at $\sim 100 - 1000$ FPS even without HPC.



(a) (b) (c)

Fig. 6: Cloud High-Performance Computing (HPC), Edge Computing, Function-Centric Computing (FCC) and Edge HPC Cumulative Distribution Function (CDF) results of (a) Frames-Per-Second (FPS), (b) packet Round-Trip-Time (RTT) and (c) processing cost per frame (based on Amazon Lambda Pricing [16]).



(a) (b) (c)

Fig. 7: Cloud High-Performance Computing (HPC), Edge Computing, Function-Centric Computing (FCC) and Edge HPC comparison results of (a) average Round-Trip-Time (RTT) and average Frames-Per-Second (FPS), (b) average processing cost per frame (based on Amazon Lambda Pricing [16]) and average FPS, and (c) average cost and average RTT.

Our future work involves consideration of additional factors e.g. energy consumption on the drone devices and network bandwidth fluctuations to expand our offloading scheme investigations. Energy consumption is a critical factor determining the flight time of drone devices, while network quality affects the latency in transmitting data between drone and the edge, and between the edge and the cloud. These factors need to be combined with the cost and performance factors to evaluate the overall trade-offs.

References

[1] R. Gargees, B. Morago, R. Pelapur, D. Chemodanov *et al.*, "Incident-supporting visual cloud computing utilizing software-defined networking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 182–197, 2017.

[2] J. S. C. Mark Berman and L. Landweber, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, 2014.

[3] "Cloudlab," https://www.cloudlab.us/, accessed: May, 2019.

[4] N. Hossein Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, Dec 2016.

[5] B. Kim, H. Min, J. Heo, and J. Jinman, "Dynamic computation offloading scheme for drone-based surveillance systems," *Sensors*, vol. 18, p. 2982, 09 2018.

[6] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, February 2017.

[7] R. Valentino, W.-S. Jung, and Y.-B. Ko, "A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (uav) clustering networks," *Sensors*, vol. 18, p. 3751, 11 2018.

[8] C. Grasso and G. Schembra, "A fleet of mec uavs to extend a 5g network slice for video monitoring with low-latency constraints," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, 2019.

[9] H. Trinh, P. Calyam, D. Chemodanov *et al.*, "Energy-aware mobile edge computing and routing for low-latency visual data processing," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2562–2577, Oct 2018.

[10] T. Y.-H. Chen *et al.*, "Glimpse: Continuous, real-time object recognition on mobile devices," *GetMobile: Mobile Comp. and Comm.*, vol. 20, no. 1, pp. 26–29, Jul. 2016.

[11] X. Wang, A. Chowdhery, and M. Chiang, "Skyeyes: Adaptive video streaming from uavs," in *Proceedings of the 3rd Workshop on Hot Topics in Wireless*, ser. HotWireless '16. New York, NY, USA: ACM, 2016, pp. 2–6. [Online]. Available: http://doi.acm.org/10.1145/2980115.2980119

[12] J. Wang *et al.*, "Bandwidth-efficient live video analytics for drones via edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct 2018, pp. 159–173.

[13] X. Sun and N. Ansari, "Latency aware drone base station placement in heterogeneous networks," in *IEEE GLOBECOM*, Dec 2017, pp. 1–6.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE CVPR*, 2016, pp. 779–788.

[15] K. K. Chintalapudi and L. Venkatraman, "On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15. 4 hardware," in *IEEE IPSN*, 2008, pp. 356–367.

[16] "Aws lambda," https://aws.amazon.com/lambda/, accessed: May, 2019.

[17] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[18] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, and K. Palaniappan, "On qoe-oriented cloud service orchestration for application providers," *IEEE Transactions on Services Computing*, 2018.

[19] D. Chemodanov, P. Calyam, and F. Esposito, "A near optimal reliable composition approach for geo-distributed latency sensitive service chains," in *IEEE INFOCOM*. IEEE, 2019.

[20] "Moving object classification application repository," https://github.com/Blowoffvalve/ImageProcessingWebServices, accessed: May, 2019.