Poster Abstract: Characterizing Computational Workloads in UAV Applications

Jayson Boubin, Shiqi Zhang, Venkata Mandadapu and Christopher Stewart
Department of Computer Science and Engineering, The Ohio State University

***Abstract*—Unmanned aerial vehicles (UAV) enable novel but demanding computational workloads that exceed processing capacity of their onboard resources. Mobile and edge devices can support demanding workloads, but they increase network communication and power usage. These resource constraints block potentially transformative UAV applications that execute too slowly or use too much power. This poster presents early efforts to characterize computational demands of emerging UAV applications. We are building a *selfie-drone benchmark*. Our benchmark will capture processing metrics, e.g., CPU usage, cache misses, power usage, etc. It will also enable characterization across a wide range of local and edge setups. Our benchmark uses a micro services design, making it easy to move workload execution across multiple contexts. Early results show that our benchmark functions well, i.e., accurately detects faces and safely uses flight controls. Further, edge devices matter. A smart phone uses 4X less power than a laptop when executing our benchmark.**

I. Introduction

Sales for unmanned aerial vehicles (UAV) are growing rapidly. As shown in Figure 1, sales in the United States likely exceeded $1.3B in 2017 [1]. From 2015 to 2017, sales nearly doubled market predictions [2]. Commercial applications are powering the surge. In the media and entertainment industry, UAV applications like automated photography could provide $8.8B in value. In agriculture, applications like crop scouting could provide $32.4B. For city infrastructure projects, UAV applications could provide $45B [3].

Modern UAVs fly with cameras, GPS sensors, processors and storage onboard. These resources, along with the act of flying, consume energy stored in UAV batteries. Data sensed from these resources could be processed onboard the UAV, but the computational demands can be significant. Consider following UAV applications:

1. Autonomous Selfie Drone: An end user or sensed condition triggers takeoff. The UAV repeatedly captures images of its surroundings, looking for targets (faces). Once a target is found, the UAV positions itself to take high quality snapshots and then it lands.

2. Crop Scouting [4]: The UAV uses positioning sensors to explore its surroundings. When targets (weeds) are discovered, the UAV takes geo-tagged photos. The UAV lands when the mission completes or batteries deplete.

3. Surveillance [5]: Like crop scouting, surveillance first requires exploring surroundings. However, when a target is discovered, the UAV may seek to follow the target, beacon for help, release a payload or otherwise actuate the world.

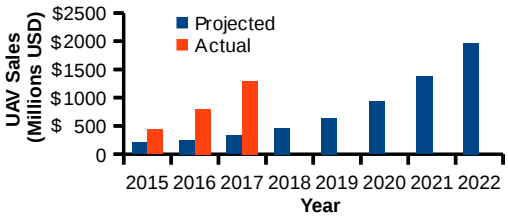

Fig. 1. Projected and actual UAV sales by year.

Computational workloads for these applications are demanding. They all requiring processing multiple images per second to discover targets. Crop scouting requires positioning and scene understanding. Surveillance requires computing complicated flight paths instantaneously. Processors onboard modern UAV are too slow. Further, modern UAV already perform complex computations, e.g., Kalman filters and object avoidance, to ensure aerial stability and safe flight [5].

Mobile and edge systems can connect to UAV through WiFi or wide-area networks. They provide additional compute capacity for these workloads, but they increase latency. Popular platforms, like DJI Go or Litchi, use such edge architecture to offload computation. Our research will study resource provisioning for UAV connected to edge devices. How many processors, what type and where are needed to support the envisioned workloads? What role will energy-aware optimizations, both onboard and at the edge, play [6], [7]? Can platforms provide first-class support to adapt resource usage under changing energy budgets [8], [9]?

This poster presents early efforts to characterize computational demands of emerging UAV applications in edge devices. We are building a selfie-drone benchmark. Our benchmark will capture processing metrics on the edge device, e.g., CPU usage, cache misses, power usage, etc. It will also enable characterization across a wide range of local and edge processing configurations. Our benchmark uses a micro-services design, making it easy to move workload execution across the edge.

II. Design

As shown in Figure 2, our benchmark models UAV applications using 5 logical layers. The UAV itself, onboard processors and storage and edge devices comprise the hardware layer. The runtime layer manages execution atop processors. This layer maps workload components (Java containers) to

Layer	Function	Key Technology
hardware	execute workloads, sense & affect the physical world	Flight, compute & edge resources
runtime manager	map workloads to hetero-geneous & dynamic devices	java, containers, sdn, coap
device/ middleware api	provide interfaces to control UAV sensors and actuators	dji android api, rapl, opencV, etc.
application	workflow for sensing, pattern detection & actuation	java, python, scratch
use-case customization	customize sensed patterns to use cases	json, python, ML models

Fig. 2. Benchmark design.

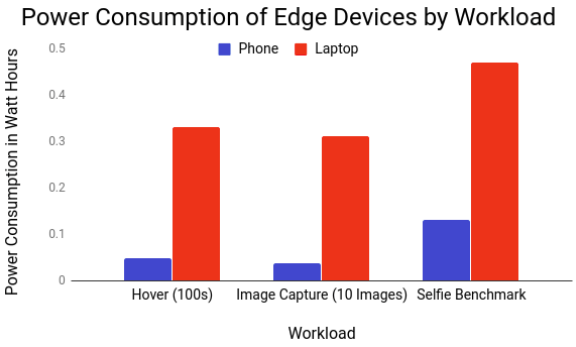

Fig. 3. Power usage under different execution contexts.

devices for execution. We use CoAP to exchange messages between components.

The API Layer is consists of interfaces to the technologies necessary for complex UAV applications. Each component has ”drivers” which expose functionality to the developer. For instance, drivers for a DJI drone expose the ability to take off, control the drone in flight, capture images, and more. Our drivers allow the developer to interact with computer vision software for real time image analysis, the operating system of the edge device, and performance information such as power consumption on the edge device and the drone. The Application layer is an abstraction above the API layer. It Allows for the development of benchmarks and other multi-faceted Autonomous UAV applications by combining calls to the API layer with control flow in Java.

Our selfie-drone benchmark application uses drivers to create a workload representative of common consumer UAV applications. It lifts the drone about 1 meter into the air and rotates it around it's yaw axis at eight 45 degree intervals. At each interval, it captures two images at different gimbal pitch positions. It uses OpenCV to detect human faces in the images. If none of the images contain a human face, the drone lands. If an image contains a human face, the drone will rotate and adjust the gimbal to center the face, and take a *selfie*. The benchmark will also log performance metrics and power consumption for later analysis.

III. Early Results

Using our benchmark and other simple workloads, we compared the power consumption of two edge devices interacting with the drone and other necessary APIs. The devices used were an LG Nexus 5 with a 2.26GHz ARM processor and a 2300mAh, 3.8v battery, and a Lenovo Thinkpad T470 laptop with a quad core, 3.5GHz x86 processor, and a 6400mAh, 11.25v battery.

To test the power consumption of these devices, we ran 2 simple workloads, as well as our selfie benchmark. The hover workload simply hovered the drone for 100 seconds. The image capture workload used the drone to capture 10 images and download them onto the edge device into RAM. The third workload was the selfie benchmark described above, ran in full, with no humans visible to the drone in any image.

Figure 3 shows power consumption for both devices in watt hours. Both simple workloads have a power consumption difference of 8x between the phone and laptop, whereas the selfie benchmark experienced a 4x increase. The simple workloads primarily idle the edge devices. The more complicated selfie workload requires the edge devices to run face recognition software and more complicated control flow. On simple workloads, the x86 laptop will obviously consume more power than the mobile device. As the computational complexity of the workload increases, this overhead can be eclipsed by the greater compute power of the x86 chip, as with selfie benchmark. In a power constrained environment, it may be ideal to sacrifice compute power in order to avoid the power overhead of a laptop. For applications that require many resources or low latency computation, the power savings from a phone may be eclipsed by it's limited resources. The ability analyze these tradeoffs and others will be useful for industry and academia when designing systems involving future intricate UAV applications.

Acknowledgements: Funded in part by NSF grants #1749501 and #1350941.

References

[1] J. Dunn, “Drones are growing rapidly, regardless of what the government does,” 2017.
[2] Statista.com, “Commercial drone revenue in North America,” 2018.
[3] D. Joshi, “Commercial unmanned aerial vehicle market analysis,” 2017.
[4] S. Khanal, R. P. Anex, B. K. Gelder, and C. Wolter, “Nitrogen balance in iowa and the implications of corn-stover harvesting,” *Agriculture, ecosystems & environment*, vol. 183, 2014.
[5] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, “Aerostack: An architecture and open-source software framework for aerial robotics,” in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, 2016.
[6] N. Morris, S. M. Renganathan, C. Stewart, R. Birke, and L. Chen, “Sprint ability: How well does your software exploit bursts in processing capacity?,” in *International Conference on Autonomic Computing*, 2016.
[7] C. Wang, B. Urgaonkar, A. Gupta, L. Y. Chen, R. Birke, and G. Kesidis, “Effective capacity modulation as an explicit control knob for public cloud profitability,” in *IEEE ICAC*, 2016.
[8] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, “Measuring and managing answer quality for online data-intensive services,” in *IEEE International Conference on Autonomic Computing (ICAC)*, 2015.
[9] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, “Obtaining and managing answer quality for online data-intensive services,” *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 2, no. 2, 2017.