IET Communications

Research Article

Computing in the air: an open airborne computing platform

ISSN 1751-8628
Received on 22nd May 2019
Revised 24th September 2019
Accepted on 26th November 2019
doi: 10.1049/iet-com.2019.0515
www.ietdl.org

Baoqian Wang[1,2], Junfei Xie[1] ✉, Songwei Li[3], Yan Wan[3], Yixin Gu[3], Shengli Fu[4], Kejie Lu[5]
[1]*Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182, USA*
[2]*Department of Electrical and Computer Engineering, University of California San Diego, San Diego, CA 92093, USA*
[3]*Department of Electrical Engineering, University of Texas at Arlington, Arlington, Texas 76019, USA*
[4]*Department of Electrical Engineering, University of North Texas, Denton, Texas 76201, USA*
[5]*Department of Computer Science and Engineering, University of Puerto Rico at Mayagüez, Mayagüez 00681, Puerto Rico*
✉ *E-mail: jxie4@sdsu.edu*

Abstract: In recent years, we have witnessed fast-growing unmanned aerial systems (UAS) based applications. To better facilitate these applications, many efforts have been made to enhance the capability of UAS from various aspects, including communications, control and networking, and so on. Nevertheless, most of these studies neglect the computation aspect. Recently, the UAS-enabled mobile edge computing (MEC) has attracted increasing research attention, which utilises UAS with onboard computing capability to provide on-demand computing services for mobile users. However, existing research on UAS-enabled MEC remains at the theory stage and how to design a UAS platform with advanced onboard computing capability has not been addressed. In this study, the authors aim to fill this research gap and design an open UAS-based airborne computing platform with advanced onboard computing capability. This platform was designed from three aspects: hardware, software, and applications. In particular, feasible computing hardware to perform UAS onboard computing is first considered and a prototype is then designed. To enhance the flexibility and programmability of the platform, two key virtualisation techniques are then investigated. Finally, they test the performance of their prototype by executing real UAS onboard computing tasks, the results of which verify the feasibility and potentials of the proposed airborne computing platform.

1 Introduction

Over the past few years, unmanned aerial systems (UAS) have become increasingly important. On the one hand, a single UAS or a group of them can support many commercial and civilian applications, such as forest-fire detection [1], reconnaissance [2], search and rescue [3], and 3D mapping [4]. On the other hand, UAS can be connected with many ground-based devices to facilitate more applications, such as data collection [5], and so on.

In the literature, many researchers have been working on the design of UAS platforms focusing on control [6], communications [7–9], networking [10], and so on. Nevertheless, we notice that the computation aspect of the UAS platforms has been largely neglected. For instance, most existing UAS platforms have limited computing capability and can perform only essential functionality, such as flight control, image/video capturing, and sensor data collection [11–14]. Consequently, computation-intensive tasks are often offloaded to the ground stations or to the cloud, which may lead to many issues. For instance, such a computing model may lead to significant transmission delays or failures, and thus cannot support many delay-sensitive applications. Moreover, for many high-bandwidth applications, such as real-time object detection and tracking, such a model requires large communication bandwidths, which may not be feasible in certain scenarios.

With the rapid development of Internet of Things (IoT) and the proliferation of mobile devices, the demand for computing resources is increasing dramatically among mobile devices. To address this challenge, mobile edge computing (MEC) [15] is deemed a promising solution, which allows computation-intensive and/or delay-sensitive tasks to run on resource-limited mobile devices, by moving computing resources to the edge of cellular networks. Recently, researchers propose to use UAS equipped with computing capability as computing servers to assist MEC. Such UAS-enabled MEC [16], once proposed, has attracted an increasing research attention, which allows on-demand computing services to be provided anytime and anywhere, even in the absence of communication infrastructure. This is especially useful in emergency scenarios where mobile users can leverage the onboard computing resources of the UAS to perform advanced data analysis for damage assessment [17].

Although researchers have realised the great potentials of UAS as a computing platform, existing studies on UAS-enabled MEC [17–23] are mainly based on theoretical analyses and simulations, and consider problems such as computation offloading and UAS trajectory design, and so on. The fundamental issue of how to design a UAS platform with advanced onboard computing capability has not been addressed. For instance, Jeong *et al.* [18] studied the computation offloading process from a static mobile device to a UAS with predetermined trajectory, and proposed an optimal bit allocation strategy for communication and computing to minimise the mobile energy consumption under maximum latency constraints. Extended from this study, Jeong *et al.* [17] further considered multiple mobile devices and UAS with unknown trajectory, and developed a strategy that jointly optimises bit allocation and UAS trajectory to minimise the total mobile energy consumption under both maximum latency constraints and UAS's energy budget constraint. As Jeong *et al.* [17, 18] assume all local data are offloaded to the UAS for computing, which may be impractical in reality due to the limitation of mobile devices' transmit power, Hua *et al.* [19] further generalised the approach and introduced a resource partitioning strategy, allowing part of the computation task to be executed locally for reduced communication energy consumption. Xiong *et al.* [20] studied a similar problem, but considered a different UAS movement pattern and the binary offloading paradigm, where the computation task is either completed locally or offloaded to the UAS. As real computational tasks are generated randomly, the stochastic computation offloading was explored in [21], where a stochastic queue model is used to simulate the arrival of computation tasks. To address the scenarios where mobile devices have limited battery energy for offloading or other computing tasks, Zhou *et al.* [22, 23] proposed a UAS-enabled wireless powered MEC system, where UAS also functions as an energy transmitter to provide power supply to mobile users through radio-frequency signals.

In this paper, we aim to fill the research gap and develop a new UAS-based airborne computing platform, which not only has advanced computing capability to allow data captured by UAS to be processed directly onboard of UAS, but also supports broadband wireless communication and networked UAS control to facilitate various network-based UAS research development, applications and services. Besides advancing UAS research, the proposed airborne computing platform also promotes the development of MEC. Compared with ground computing platforms, the UAS-based MEC is advantageous in that it allows computing services to be provided at anytime and anywhere, even in the absence of communication infrastructure, due to the 3D mobility and high manoeuvrability of UAS. Our airborne computing platform is open to the community and can be accessed through our project website [24].

Our *main contributions* include the following three parts: hardware design, software design, and applications.

(1) *Hardware design:* We first investigate how to design and implement the hardware of the UAS-based airborne computing platform. Specifically, we discuss the desired features for the airborne computing platform, especially according to the unique UAS structure and applications. We then conduct a comprehensive analysis and systematic study on ten state-of-the-art single-board computers regarding the computation performance, power consumption, size, and weight. Based on the thorough comparison, we choose NVIDIA Jetson TX2 as the computing unit for the platform. A prototype is then designed and implemented, which integrates Jetson TX2 for onboard computations, UAS, broadband communication system, and networked control system.
(2) *Software design:* With respect to the software of the platform, we aim to design an airborne computing platform with sufficient flexibility and programmability. A key technology to achieve this goal is virtualisation, because it can efficiently manage resources, can enable concurrent applications, and can enhance the security of the computing platform. In this paper, we investigate two key virtualisation techniques: (i) virtual machine (VM) using KVM [25] and (ii) container using Docker [26]. To understand the impact of virtualisation on UAS applications, we conduct extensive experiments to measure the performance of the two virtualisation techniques from the aspects crucial for UAS applications, including computing, networking, isolation, power consumption, and so on. The performance trade-offs are also discussed. These experiments verify the feasibility of virtualising UAS and demonstrate the potentials of virtualisation in enhancing UAS' onboard computing capability. The insights obtained from the comparison results between KVM and Docker also provide guidelines for the selection of appropriate virtualisation techniques for UAS applications.
(3) *Applications:* Using the prototype with virtualisation supports, we further test the performance of several real UAS applications that may benefit from the onboard computing capability. For instance, we test UAS image processing tasks of different complexities to evaluate the benefit of virtualisation. We also test the feasibility of performing real-time object detection onboard of UAS. To understand the potentials of distributed computing over a UAS network, we also implement an advanced coded distributed computing approach for efficient matrix multiplication, which is one of the most critical computing tasks for many artificial intelligent applications.

The rest of this paper is organised as follows. In Section 2, we discuss the hardware selection for UAS-based airborne computing, and we present the details for the design of a prototype. Next, in Section 3, we elaborate on the use of virtualisation to extend the functionality of the airborne computing platform and we present extensive experimental results on performance of the computing platform with the virtualisation capability. We then further investigate, in Section 4, the benefits of using the proposed airborne computing platform for real computation-intensive UAS applications. Finally, in Section 5, we conclude the paper with a brief summary and future works.

2 Hardware design for the airborne computing platform

In this section, we investigate the hardware design for UAS high-performance onboard computing. In particular, we first discuss the desired features for the onboard computing hardware. We then analyse ten state-of-the-art single-board computers and provide guidelines for choosing a suitable single-board computer as the computing unit. A prototype designed based on a selected single-board computer is then described.

2.1 Desired features

Since the onboard computing hardware is carried by a UAS, there are some unique considerations for the selection of a single-board computer. First, the computing hardware should be of light weight and compact size, due to the limited payload and space provided by the UAS. Second, because of the limited power capacity of the UAS-carried batteries, it is preferable to have an efficient power management system to reduce the power consumption. Third, in terms of computing, the hardware should have a powerful CPU, sufficient memory and storage to support most computing needs. Moreover, a powerful GPU is also necessary to enable real-time image processing and deep learning capabilities onboard of UAS. Last but not the least, the single-board computer should have extensive community support, such that developers and users can share experience and seek online support during their system development. In addition, it is also very important to have sufficient open access design documentation and configuration toolboxes.

2.2 Single-board computer selection

In the literature, several single-board computers are commonly used for UAS operations, including Raspberry PI [27], Odroid XU [28], Arduino Board [29], Cubieboard [30], Arndale Board [31], and so on. In our study, we do not consider them because they are not powerful enough to fulfill computation-intensive tasks.

Unlike the aforementioned computing devices, more powerful single-board computers have not been fully investigated for UAS. Recently, Shang and Shen [32] investigated the computing power of NVIDIA Jetson TX1 [33] with 4 CPU cores, 256 Maxwell CUDA GPU cores, and 4GB memory. Their studies show that NVIDIA Jetson TX1 is still not sufficient enough to achieve real-time 3D reconstruction and mapping using the simultaneous localisation and mapping algorithm.

To select a suitable single-board computer to enable UAS high-performance onboard computing, we consider those with computing capability comparable to Jetson TX1. In particular, ten state-of-the-art single-board computers, including NVIDIA Jetson TX2 [34], UDOO X86 ULTRA [35], Intel Aero Compute Board [36], LattePanda Alpha [37], Up Squared [38], NVIDIA Jetson Xavier [39], DJI Manifold [40], HiKey960 [41], Rock960 [42], and Jetson TX1 [33], are found and compared in detail from various aspects (see Table 1 for the comparison results).

As shown in Table 1, Jetson Xavier with 8 CPU cores and 512-core Volta GPU with Tensor cores has the highest computing power. Jetson Xavier also excels in memory capacity. However, it is the most power consuming and costly, and it does not natively support Wi-Fi communications. Jetson TX2 with 6 CPU cores, 256-core NVIDIA Pascal GPU, and 8GB memory is the second powerful single-board computer. It provides an out-of-the-box high-throughput wireless local area network (WLAN) interface, and is also the smallest in size. UDOO X86 ULTRA outperforms others in power consumption and operating system (OS) support; Intel Aero Compute Board is the lightest among those with known weight information; Rock960 is of the lowest cost; and Up Squared has the largest storage. All these single-board computers support virtualisation.

The above analysis provides us with guidelines to select proper single-board computers for UAS high-performance onboard computing. A trade-off should be achieved among different performance aspects based on the needs of specific applications. For instance, if flight time is more critical than real-time processing, UDOO X86 ULTRA that consumes less power or the lightweight Intel Aero Compute Board may be selected. If cost is of the major concern, Rock960 can be a good choice.

In this study, we select the NVIDIA Jetson TX2 as the computing hardware for the airborne computing platform. As shown in Table 1, its overall computing capability is above the average, especially considering the availability of a powerful GPU. Both the power consumption (7.5 W) and weight (85 g) are around the average. Another attractive factor is that there is an open-access online support community including FAQ and forum [43], which is very helpful for project developers.

While Jetson TX2 has a small size of 50 mm×87 mm, the development board provided by NVIDIA is very large (17 cm × 17cm). To address this issue, we design a new carrier board, as shown in Fig. 1*a*, based on the following technical specifications. The carrier board has a dimension of 88 mm×65 mm with weight

Table 1 Comparison of different single-board computers

	CPU	GPU	Memory	Connectivity	Dimension, mm	Power consumption	OS	Weight	Virtualisation support	Storage, GB	Price
Jetson TX2	Denver 2 (2 cores) 2 MB Cache, 2 GHz + ARM® A57 (4 cores) 2 MB Cache, 2GHz	256-core NVIDIA Pascal GPU	8 GB	1 Gigabit Ethernet, 802.11ac WLAN,Bluetooth	50×87	7.5W	Linux	85 g	yes	32	$400
UDOO X86 ULTRA	Intel® Pentium N3710 (4 cores) 2 MB Cache, 2.56 GHz	Intel® HD Graphics 16 units, 405–700 MHz	8 GB	1 Gigabit Ethernet, M.2 Key E slot for optional Wireless (WiFi+Bluetooth)	120×85	6 W	Windows, Linux, Android	117 g	yes	32	$267
Intel Aero Compute Board	Intel® Atom^{TM} x7-Z8750 (4 cores) 2 MB Cache, 2.56GHz	Intel® HD Graphics 16 units, 405-600 MHz	4 GB	Intel® Dual Band Wireless-AC 8260	$88 \times 63 \times 20$	7.5 W	Linux	30 g	yes	32	$399
Lattepanda Alpha	Intel® 7th Gen M3-7Y30 (2 cores) 4 MB Cache, 2.60 GHz	Intel® HD Graphics 615 300–900 MHz	8 GB	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth	$113 \times 80 \times 13.5$	NA	Windows, Linux	104 g	yes	64	$398
UP Squared	Intel® Apollo Lake (2-4 cores)	Intel® Gen 9 HD with 12 (Celeron) or 18 (Pentium) Execution Units	8 GB	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth	85.6×90	NA	Windows, Linux, Android	NA	yes	128	$399
Jetson Xavier	ARM V8.2 (8 cores) 8 MB L2+4 MB L3, 2.26 GHz	512-core Volta GPU with Tensor Cores	16 GB	1 Gigabit Ethernet	$100 \times 87 \times 16$	10–30 W	Linux	NA	yes	32	$1299
DJI Manifold	ARM Cortex-A15 (4 cores)	192-core NVIDIA CUDA GPU	2 GB	10/100/1000BASE-T Ethernet	$110 \times 110 \times 26$	5–15 W	Linux	197 g	yes	16	$499
HiKey 960	ARM Cortex-A73 (4 cores) +Cortex A53 (4 cores)	ARM Mali G71 MP8	4 GB	WiFi, Bluetooth 4.1	$85 \times 55 \times 9$	NA	Linux, AOSP	60 g	yes	32	$249

	CPU	GPU	Memory	Connectivity	Dimension, mm	Power consumption	OS	Weight	Virtualisation support	Storage, GB	Price
Rock 960	ARM Cortex-A72 (2 cores) Cortex A53 (4 cores)	ARM Mali T860 MP4	4 GB	WLAN 802.11 ac/a/b/g/n, Bluetooth 4.2	$85 \times 54 \times 11$	NA	Linux, AOSP	120 g	yes	32	$139
Jetson TX1	ARM Cortex-A57 (4 cores) 2 MB L2	256-core NVIDIA Maxwell GPU	4 GB	1 Gigabit Ethernet,802.11ac WLAN, Bluetooth	50×87	10 W	Linux	88 g	yes	16	$299

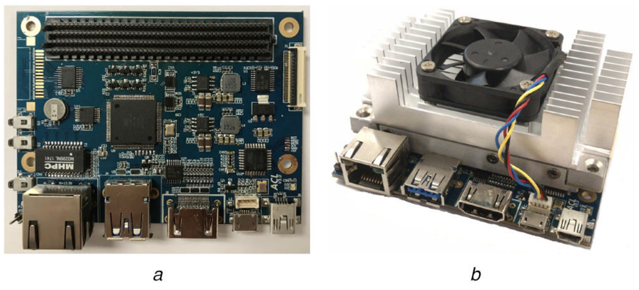


Fig. 1 *New Jetson TX2 carrier board*
(a) Without the processor, ***(b)*** With the processor

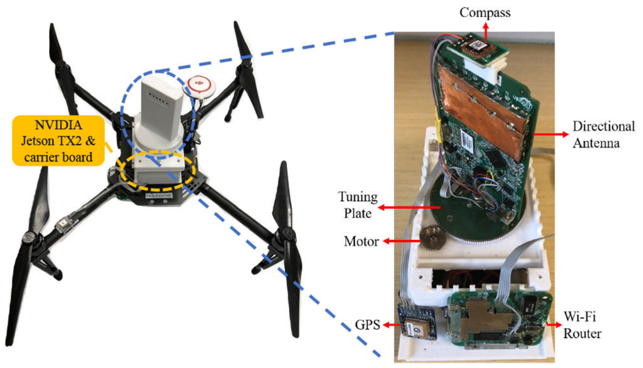


Fig. 2 *Prototype of the airborne computing platform*

of 53 g. The interfaces provided by the board include one HDMI, three UART, one CAN bus, one micro USB, one USB 2.0/3.0, one Ethernet port, four GPIO, and two camera ports. The carrier board with Jetson TX2 is shown in Fig. 1*b*.

2.3 Prototype

With Jetson TX2 chosen as the computing unit, we then develop a prototype of the airborne computing platform [44, 45] that also incorporates a *quadcopter unit* for lifting and mobility, a *communication unit* for UAS-to-UAS (U2U), UAS-to-ground (U2G) and ground-to-UAS (G2U) communications, and a *control unit* for addressing communications, networking, UAS navigation and application needs (see Fig. 2).

2.3.1 Quadcopter unit: The quadcopter unit serves as a platform carrier to carry other units. Compared with fixed-wing UAS, quadcopters are easier to operate, allow vertical taking off and landing, and can hover in the air. Here we select DJI Matrice 100 [46] as the quadcopter unit, due to its nice properties in terms of payload, expandability, stability, and operability. For instance, the maximum weight allowed for the DJI Matrice 100 while taking off is 3.6 kg, which exceeds the total weight of the whole system of 3.13 kg. With a LiPo 6 s battery, our prototype can fly for around 18 min.

2.3.2 Communication unit: The communication unit supports the U2U and U2G/G2U communications. For U2U communication, we choose Ubiquiti Nanostation Loco M5 [47], a directional antenna, to enable long-range and broadband communication between two UAS. The transmission rate achieves up to 150 Mbps, allowing real-time video transmission. The maximum transmission distance is 10 km. For U2G/G2U communication, Huawei WS323 [48] is selected as the Wi-Fi router to enable communication between ground devices and the UAS, where ground devices connect to the router through the WLAN. Through this link, sensor data such as videos captured by the UAS can be transmitted to the ground for visualisation and analysis.

2.3.3 Control unit: The control unit consists of two sub-units: UAS flight control and directional antenna control. In particular, the UAS flight control sub-unit makes the UAS follow desired trajectories, while maintaining stability. It translates high-level control commands received from the remote pilot to motor pulse width modulation (PWM) signals, based on UAS state measurements captured by sensors such as GPS and inertial measurement unit. The directional antenna control sub-unit controls the heading direction of the directional antenna to maximise the performance of directional communication. This sub-unit is composed of a rotating motor, a tunable plate, a motor driver, and a compass. The tunable plate carries the directional antenna and the compass. To rotate the plate to a specified angle, the motor driver takes the control signal generated by the computing unit and translates it to PWM signals. The motor then drives the plate to rotate based on the PWM signals. Here we select MTI-3-8A7G6T Xsens and Adafruit TB6612 as the compass and the motor driver, respectively.

3 Software design for the airborne computing platform

In this section, we investigate two key virtualisation techniques, VM [49] using KVM and container using Docker [26], to improve the flexibility and programmability of the airborne computing platform. In particular, we first provide a brief overview of the current research status in the field of virtualisation. To understand the impact of virtualisation, we then conduct a series of experiments to study their performances from multiple aspects, including computing, networking, isolation, power consumption, and so on, as well as the trade-offs among them.

3.1 Background and related work

To support diverse computing tasks on the same UAS platform, one of the key technologies is virtualisation. First, virtualisation provides powerful resource management capabilities, so that it can efficiently enable an application with specific computing requirements, such as CPU, memory, storage, networking, and so on. Second, virtualisation can facilitate concurrent execution of multiple applications on the same UAS. Third, in terms of security, virtualisation can isolate unreliable and untrustworthy functionality, and improve the resilience of UAS to malicious attacks [50]. In addition to these advantages on a single UAS, virtualisation can help to exploit the distributed computing capabilities on multiple connected UAS, which can evolve towards future generation of the networked airborne computing.

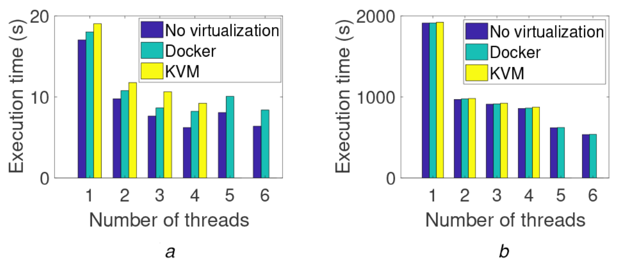


Fig. 3 *Execution time of the SC application implemented on CPU using increasing number of threads. The number of input points in the SC application is set to*
(a) 10,000, ***(b)*** 900,000

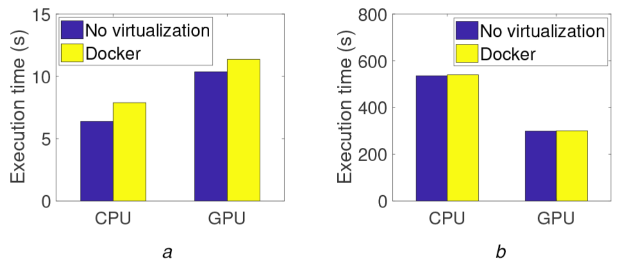


Fig. 4 *Execution time of the SC application implemented on GPU and CPU of six threads. The number of input points in the SC application is set to*
(a) 10,000, ***(b)*** 900,000

Virtualisation has been studied extensively in the literature. The server-based virtualisation has been mature and widely implemented in computing systems, especially the cloud [51–53]. Virtualisation for mobile devices is more relevant to this study, which has aroused increasing attention with the wide use of mobile devices and the fast evolution of ARM processors, but is still under development [54]. In the last few years, several studies have investigated the performance of virtualisation on different mobile devices, such as Raspberry PI 2 [55], Cubieboard2 [56], ARM Chromebook [57], Banana Pi [58], and Insignal Arndale board [59], and so on. However, since these studies were not directed to UAS, their performance analysis was limited to CPU, memory, and disk in a single device. The unique features of UAS such as small payload, power constraint, and real-time computing need, as well as the special characteristics of multi-UAS applications such as U2U communications and network connectivity were also not considered in these studies.

Virtualisation has also played a critical role in emerging computing paradigms including IoT, fog computing, and MEC. For instance, container-based virtualisation is applied in [60, 61] to enable data processing at IoT devices. A Docker-based fog computing framework over Raspberry PI is described in [62]. In [63], the integration of IoT and fog computing with virtualisation deployed in fog nodes is studied.

Despite the abundant works on virtualisation, virtualisation for UAS has been rarely studied. Among the limited studies we can find, paper [50] utilises virtualisation to enhance the resilience of UAS to malicious attacks, where Raspberry PI 2 is adopted as the onboard computing unit. Nutanix recently released a commercial UAS cloud platform, called Acropolis [64], which can hold multiple VMs. In [65], virtualisation is implemented on fog servers to provide computing services for UAS fire detection. Overall, a comprehensive investigation of virtualisation for UAS to enable high-performance onboard computing and advanced UAS applications is still lacking.

Virtualisation can extend the computing capabilities of UAS, at a cost of performance overheads, due to resource partition, isolation, and emulation. To understand the impact of virtualisation, we next investigate the performances of two key virtualisation techniques, KVM [25] and Docker [26], which are representatives of the hypervisor-based and container-based virtualisation techniques, respectively. Please refer to [66] for a brief introduction of the two virtualisation techniques, and the instructions to implement these techniques on Jetson TX2. In this study, both guest (VM or container) and host systems in Jetson TX2 implement Ubuntu 16.04 LTS with Linux kernel version 4.4 as the OS.

3.2 Computing performance

In this subsection, we investigate the impact of KVM and Docker on the CPU and GPU computing performances of the proposed airborne computing platform, which is crucial for the success of many time-critical UAS applications.

3.2.1 Experimental setup: To measure the computing performance of the airborne computing platform with virtualisation capability, we create a VM that virtualises CPU or GPU resources using KVM (or container using Docker) on the platform, and install the Rodinia Benchmark Suite [67] in the VM (or container). We then run the Stream Cluster (SC) application in the benchmark, which performs clustering for data streams. The execution time of the SC application indicates the computing performance. To reduce experimental uncertainty, each experiment is repeated for ten times and the average execution time of the SC application is presented. This procedure is also applied to each experiment conducted in following studies.

In the experiment on the CPU performance, as the benchmark supports CPU multi-threading, which allows applications to be executed by multiple CPU cores in parallel, we vary the number of threads to test the parallel CPU computing performance. Note that each thread uses one CPU core. As only four ARM A57 CPU cores can be virtualised using KVM, up to four threads are evaluated for KVM. Docker successfully virtualises all six CPU cores in Jetson TX2, and thus up to six threads are evaluated for Docker.

In the experiment on the GPU performance, as KVM does not support CUDA based GPU [68], we only evaluate the impact of Docker on the GPU performance, which adopts the PCI pass-through technique [69] to achieve GPU virtualisation. We also compare the GPU performance of the airborne computing platform with its best CPU performance, i.e. using six threads. In both experiments, we vary the size of the input data stream in the SC application to test the scalability of the computing platform.

3.2.2 Experimental results: Fig. 3 shows the CPU performance of the airborne computing platform before and after implementing KVM or Docker. As shown in the figure, both KVM and Docker introduce performance overheads, and KVM degrades the computing performance more. This is because KVM adopts more complicated procedures to allocate memory resources, in particular using second-level address translation [25], while Docker achieves this by directly utilising the Linux system utility, i.e. control groups (cgroups) [26]. Our experiments also show that running five or six threads on Denver CPU cores does not improve the efficiency when the size of the data stream is small (see Fig. 3*a*). This is due to the overheads for coordinating different CPU processors.

Fig. 4 compares the performance of GPU and that of the CPU in two scenarios, i.e. before and after implementing Docker. As we can see from the figure, the computing performance of GPU is slightly worse than that of the hex-core CPU when the problem size is small, but GPU significantly outperforms the CPU when the problem size is large.

3.3 Networking performance

In this subsection, we investigate the impact of KVM and Docker on the networking performance of the airborne computing platform, which is crucial for reliable and timely information sharing between UAS and ground mobile devices as well as among UAS.

3.3.1 Experimental setup: In this study, we conduct experiments to evaluate the networking performance for both the G2U/U2G and U2U communication links. To test the G2U/U2G communication link, we connect the airborne computing platform to a ThinkPad E540 laptop with the bandwidth of the Wi-Fi route set to 40 Mbps.

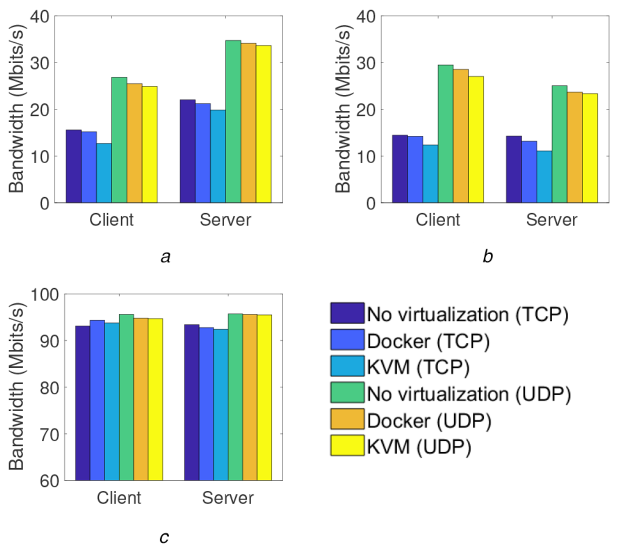


Fig. 5 *Bandwidth of the communication link*
(a) Between airborne computing platform and ThinkPad laptop connected using a Wi-Fi router, ***(b)*** Between two airborne computing platforms connected using a Wi-Fi router, ***(c)*** Between two airborne computing platforms connected using directional antennas

Table 2 Performance degradation of the well-behaved guest in different stress tests

	Docker, %	KVM, %
CPU	0.36	0.41
memory	5.03	6.0
disk I/O	2.56	2.9
fork bomb	6.24	1.28
network receiver	2.25	4.68
network sender	1.73	2.53

To test the U2U communication link, we consider two scenarios: (i) the omni-directional antenna based short-distance communications and (ii) directional antenna based long-distance communications. These two scenarios are tested by linking two computing platforms using the Wi-Fi router and the Ubiquiti Nanostation Loco M5, respectively.

To measure the networking performance, we install the Iperf benchmark [70] on airborne computing platforms and the Thinkpad laptop. This benchmark measures the throughput between two connected devices by sending data streams from one device (called client) to the other (called server). To obtain a comprehensive understanding of the networking performance, we vary the role of the airborne computing platform (client or server) and also vary the transmission protocol (TCP or UDP).

3.3.2 Experimental results: The networking performance of the airborne computing platform before and after implementing KVM or Docker under different networking configurations is shown in Fig. 5.

In all these experiments, Docker shows less impact on the networking performance than KVM. This is due to the simplicity of Docker in network virtualisation. In particular, unlike KVM that requires emulation of network devices in VMs to enable communications, Docker containers can directly build network connections by using the Linux system utilities, e.g. network namespace [26]. Another phenomenon observed in all experiments is that higher bandwidths are achieved when the UDP transmission protocol is adopted, as UDP sends packets continuously without acknowledgements.

Now let us analyse each subfigure. Fig. 5*a* shows that the bandwidth of the G2U/U2G communications increases when the airborne computing platform acts as a server. This is mainly caused by the use of different network devices in Jetson TX2 and ThinkPad laptop. In cases when two identical airborne computing platforms are connected to simulate the U2U communications, the bandwidth measured at the server side is smaller than that measured at the client side (see Figs. 5*b* and *c*). This is because VM or container requires port forwarding to receive packets, which introduces some overheads [71, 72]. The comparison between Fig. 5*c* and the other two subfigures suggests that the high bandwidth is achieved by directional antennas, demonstrating their advantage over omni-directional antennas. Of interest, in Fig. 5*c*, when the TCP transmission protocol is adopted, the bandwidth measured at the client side increases after virtualisation. This may be caused by the bridge network in KVM and Docker, which buffers packets sent from the guest to the host network interface and in turn helps alleviate traffic congestion and increases the packet transmission rate.

3.4 Isolation performance

Virtualisation can enhance the security of UAS applications, by isolating unreliable functionality. It can also enable concurrent execution of programs with different system requirements on the same airborne computing platform, by running these programs in different VMs (or containers). The success of these applications relies on how well VMs (or containers) are isolated, which is investigated in this subsection.

3.4.1 Experimental setup: To test the isolation performance of KVM and Docker, we follow similar experimental setups in [73]. In particular, we create two guests (VMs or containers) in the airborne computing platform, and assign each guest with two ARM A57 CPU cores exclusively. We then install the Isolation Benchmark Suite (IBS) [74] to evaluate the isolation performance, which works by measuring the impact of a misbehaved guest (runs a stress test) on a well-behaved one (runs a baseline application). The smaller the impact is, the better the two guests are isolated. In this study, we run the lower-upper Gauss–Seidel solver (LU) application in the well-behaved guest, which performs a synthetic computational fluid dynamics calculation for a cubic region [75]. We here set the cubic size to $64 \times 64 \times 64$. In the misbehaved guest, we run different stress tests available in IBS to test the performance of KVM and Docker in isolating different hardware resources.

To evaluate the impact of the misbehaved guest on the well-behaved one, we measure the performance degradation of the well-behaved guest using the following equation:

$$\frac{T_s - T_n}{T_n} \times 100\% \tag{1}$$

where T_n and T_s represent the execution time of the LU application before and after running the stress test in the misbehaved guest, respectively.

3.4.2 Experimental results: Table 2 shows the isolation performance of KVM and Docker in different stress tests. In particular, Docker demonstrates less performance degradation than KVM in the CPU, memory, disk I/O, and network intensive stress tests, indicating relatively better performance in isolating these hardware resources. This is because Docker directly uses Linux namespaces to achieve isolation, but KVM utilises the hypervisor's trap-and-emulate mechanism that hangs the rest of the VMs when one traps to the hypervisor [25]. In the fork bomb test that generates large amount of processes to overwhelm the OS, the performance of Docker degrades significantly compared to KVM, as containers share the same OS kernel with the host system. Overall, both KVM and Docker perform well in isolating CPU resources, as different VMs or containers occupy different CPU cores. However, both are relatively weak in isolating the other hardware resources.

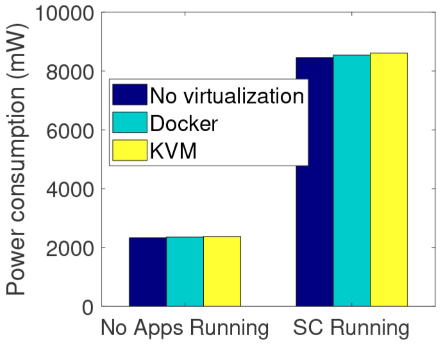
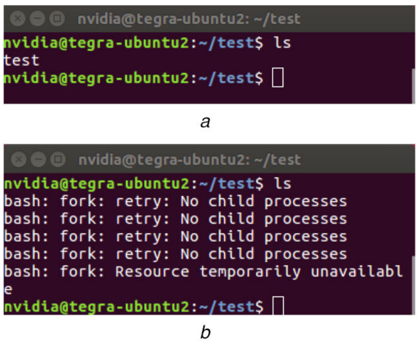


Fig. 6 *Power consumption of the airborne computing platform before and after running the SC application*

Table 3 Resource usage of a bare VM or container

	VM	Container
CPU	2.7%	0%
memory	476 MB	0.3 MB
storage	1.3 GB	103 MB



a

```
nvidia@tegra-ubuntu2: ~/test
nvidia@tegra-ubuntu2:~/test$ ls
test
nvidia@tegra-ubuntu2:~/test$
```

b

```
nvidia@tegra-ubuntu2: ~/test
nvidia@tegra-ubuntu2:~/test$ ls
bash: fork: retry: No child processes
bash: fork: retry: No child processes
bash: fork: retry: No child processes
bash: fork: retry: No child processes
bash: fork: Resource temporarily unavailabl
e
nvidia@tegra-ubuntu2:~/test$
```

Fig. 7 *Response of the host OS when the*
(a) VM is attacked, ***(b)*** Container is attacked

3.5 Power consumption

In this section, we study the impact of KVM and Docker on power consumption, which directly influences the flight endurance of the proposed airborne computing platform.

3.5.1 Experimental setup: To measure the power consumption of the airborne computing platform, we use the built-in three-channel INA 3221 monitors in Jetson TX2 [76]. Two experiments are then conducted to evaluate the impact of KVM and Docker on the power consumption of the airborne computing platform at different operating conditions. Particularly, in the first experiment, the airborne computing platform does not run any applications, and its power consumption is measured before and after implementing KVM or Docker. In the second experiment, the power consumption of the platform when running the SC application is measured.

3.5.2 Experimental results: The power consumption of the airborne computing platform before and after implementing KVM or Docker in the two experiments is shown in Fig. 6. As we can see, both KVM and Docker increase the power consumption slightly in the two experiments. KVM consumes more power than Docker, as it introduces more overhead. Also note that compared with the power consumed by running the SC application, the power consumed by virtualisation is negligible.

3.6 Discussions

In the above comparative studies, we evaluate the performances of KVM and Docker from the aspects of computing, networking, isolation, and power consumption that are of major concern to UAS applications. In this subsection, we briefly discuss other performance aspects that are also of interest.

3.6.1 Resource usage: Compared with VMs, containers consume fewer resources and thus can be quickly deployed. To demonstrate this feature, we conduct a simple experiment to measure the resource usage of a bare VM (or container) created by KVM (or Docker). No applications run in the VM or container. Table 3 summarises the CPU, memory and storage usage of a bare VM or container measured using the *sysstat* and *free* Linux commands [77].

3.6.2 Live migration: The live migration allows a running VM (or container) to be migrated from one computing platform to another, without interruptions during the migration process. Both Docker and KVM support live migration on server-based devices [49, 78]. However, live migration on mobile devices has been rarely studied and KVM currently does not support live migration on ARM-based devices. In addition, no solution is currently available for the Docker container-based live migration on Jetson TX2. We expect that this can be realised using checkpoints and restore utility [79], which we will leave to the future work.

3.6.3 OS support: On ARM-based Linux single-board computers, KVM supports unmodified guest OSs [25], such as Ubuntu and openSUSE. However, Docker only supports ARM-based images [80], such as arm64v8/ubuntu, windows/nanoserver, and windows/iotcore.

3.6.4 Security: KVM is more resilient to malicious attacks than Docker. As VMs have their own OSs and kernels, the collapse of one VM does not influence others. However, containers share the kernel with the host OS. Therefore, once one container is attacked, other containers or even the whole system may collapse. For verification, we conduct a simple test. Specifically, we run the fork bomb, a denial-of-service attack, in the VM and container, respectively. As shown in Fig. 7, the host OS works properly when the VM is under attack, but collapses when the container is attacked.

In summary, Docker achieves better performance than KVM in most aspects relevant to UAS applications, including computing, networking, isolation of CPU, memory, disk I/O and network resources, power consumption, and resource usage. Docker successfully virtualises all CPU cores and GPU in Jetson TX2, but KVM can only virtualise the four ARM A57 CPU cores. On the other hand, KVM provides higher security. To enable more advanced UAS applications, the strengths of KVM and Docker need to be integrated. Live migration on ARM-based devices also needs to be realised for both KVM and Docker.

4 Performance of the airborne computing platform

In this section, we investigate the performance of the proposed airborne computing platform in supporting real UAS applications. In particular, we first use OpenDroneMap for UAS image processing to illustrate the benefits of virtualisation. We then further investigate the performance of two advanced UAS onboard computing tasks, real-time object detection and coded distributed computing.

4.1 OpenDroneMap

OpenDroneMap is an open-source UAS image processing software [81]. In this study, we first use the image resizing and 3D model reconstruction functions in OpenDroneMap that require different amount of computing resources to investigate the impact of virtualisation on the computing performance. Fig. 8 shows the average execution time of the two functions to process a UAS image with size of 3.9 MB in different virtualisation environments [UAS images are downloaded through this link: https://github.com/OpenDroneMap/odm_data_copr.git.]. The results demonstrate the advantage of Docker over KVM, especially in computing complicated UAS onboard computing tasks. The 3D geographical model reconstructed from 41 2D UAS images is shown in Fig. 9.

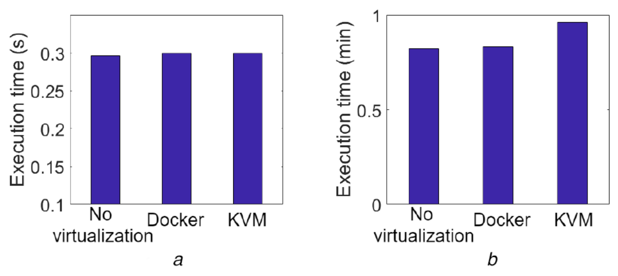


Fig. 8 *Execution time of the*
(a) Image resizing, ***(b)*** 3D model reconstruction functions in OpenDroneMap to process a UAS image in different virtualisation environments

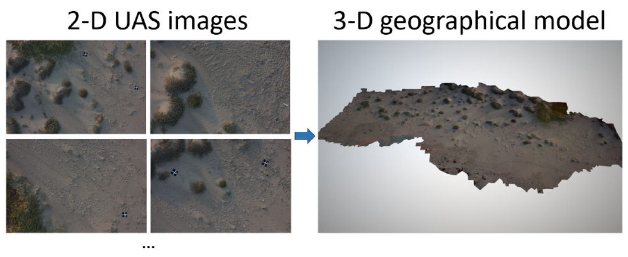


Fig. 9 *3D geographical model generated from 41 UAS images using the 3D model reconstruction function in OpenDroneMap*

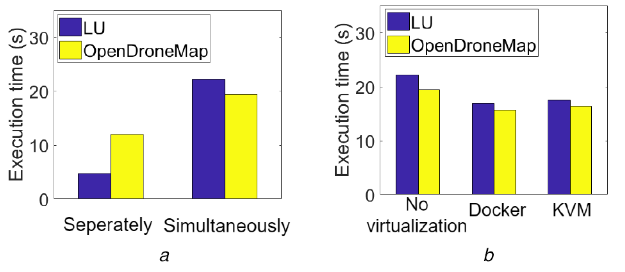


Fig. 10 *Execution time of two applications*
(a) When running separately and when running simultaneously in Jetson TX2 without virtualisation, ***(b)*** When running simultaneously in Jetson TX2 of three different virtualisation setups

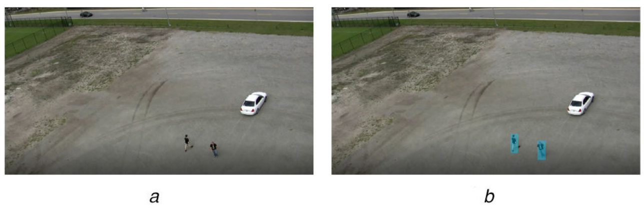


Fig. 11 *UAS image*
(a) Before applying the DNN based object detection, ***(b)*** After applying the DNN based object detection

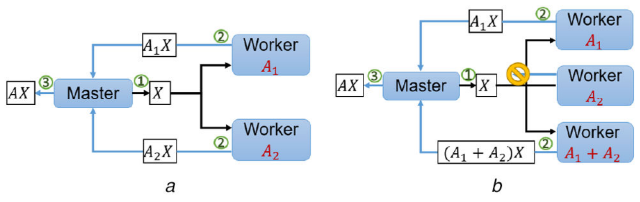


Fig. 12 *Illustration of the*
(a) Uncoded computation to perform matrix multiplication with $n = 2$, ***(b)*** Coded computation to perform matrix multiplication with $n = 2$. The numbers marked in green describe the computation procedures

We next use OpenDroneMap to demonstrate the benefits of virtualisation in facilitating resource management for UAS. Generally, virtualisation provides developers with the convenience to run multiple applications simultaneously without considering resource sharing and context switches among processes, which will increase the execution time of the applications. To illustrate this fact, we first conduct experiments to show the consequence of running two applications simultaneously when virtualisation is not applied. Fig. 10*a* shows the execution time of the LU application (cubic size is set to $36 \times 36 \times 36$) and the image resizing function in OpenDroneMap (processes 41 UAS images) when they run separately on the airborne computing platform compared to the case when they run simultaneously. As we can see from this figure, when the two applications run simultaneously, the execution time of both applications increases significantly, and the LU application even takes more time than the OpenDroneMap application.

We then implement virtualisation on the airborne computing platform, and run the two applications simultaneously but in different guests. For better overall performance, the guest that runs the LU application is allocated with one CPU core, and the one that runs the more computation-intensive OpenDroneMap application is allocated with three CPU cores. As shown in Fig. 10*b*, virtualisation helps improve the overall computing performance significantly through resource allocation and isolation, despite the associated overhead.

4.2 Real-time object detection

Real-time object detection is crucial for many UAS applications including search and rescue, traffic monitoring, infrastructure inspection, and reconnaissance. As this type of task is computationally demanding, it is typically executed at ground stations or the cloud. In this study, we show that real-time object detection can be achieved onboard of UAS even with virtualisation.

Consider the scenario where UAS is dispatched to detect and track humans in a search and rescue mission. To achieve this, we implement a deep neural network (DNN) model [82] on the airborne computing platform, which is built on GPU and uses NVIDIA TensorRT and cuDNN. This model has been pre-trained for human detection. We use ten images captured from a UAS action video [83] to evaluate the average execution time of this model to process a UAS image. In particular, the average recognition times of the airborne computing platform without virtualisation and with Docker container-based virtualisation are around 0.129 and 0.148 s per image of size 850 kB, respectively. This demonstrates the feasibility of performing real-time object detection onboard of UAS even with virtualisation. Note that it takes around 0.253 and 0.267 s to transmit a single image of size 850 kB from the airborne computing platform without virtualisation and with Docker-based virtualisation, respectively, to the ground (Thinkpad laptop) through omni-directional antenna- and UDP-based communication, according to the results shown in Fig. 5*a*. Fig. 11 illustrates the accuracy of the DNN model in recognising humans on a UAS image.

4.3 Distributed computing using codes

The onboard computing performance of UAS can be further enhanced by allowing tasks to be distributed over multiple platforms. Traditional distributed systems that allocate non-overlapping tasks to different computing nodes are sensitive to system noises such as stragglers and communication bottlenecks [84] and thus may not be suitable for UAS of high mobility and uncertain trajectories. Recently, coded computation was developed in [84] to address this issue. In this study, we investigate the performances of both uncoded and coded computations on the proposed airborne computing platform. Before we show the experimental results, let us first use an example to describe the key ideas of the two approaches.

Consider the matrix multiplication problem described in [84], which aims to multiply a large input matrix $\boldsymbol{X}$ with another pre-stored matrix $\boldsymbol{A}$. To reduce the computation time, the traditional approach (see Fig. 12*a* for an illustration) distributes the task by storing sub-matrices $\boldsymbol{A}_i$ of $\boldsymbol{A}$ at different computing nodes called worker nodes, where $\boldsymbol{A} = [\boldsymbol{A}_1; \boldsymbol{A}_2; \ldots; \boldsymbol{A}_n]$ and n is the total number of sub-matrices. To compute $\boldsymbol{AX}$, a master node first sends $\boldsymbol{X}$ to all worker nodes. Each worker node then computes $\boldsymbol{A}_i\boldsymbol{X}$ and sends the result back to the master node. As the master node cannot recover

Table 4 Execution time of uncoded and coded matrix multiplication with $n = 2$

	No straggler exists	Straggler exists
uncoded computation, s	13.09	24.58
coded computation, s	13.69	13.91

$\boldsymbol{AX}$ until all results are received, the efficacy of this approach is bounded by the slowest worker node, i.e. the straggler. The coded computation (see Fig. 12*b* for an illustration) addresses this issue by introducing redundancy into the computation through erasure codes. For instance, consider the matrix multiplication problem with $n = 2$, the coded approach introduces an additional worker node that stores $\boldsymbol{A}_1 + \boldsymbol{A}_2$. Therefore, the master node can recover $\boldsymbol{AX}$ upon receiving the results from any two worker nodes. For instance, if $\boldsymbol{A}_1\boldsymbol{X}$ and $(\boldsymbol{A}_1 + \boldsymbol{A}_2)\boldsymbol{X}$ arrive at the master node first, $\boldsymbol{AX}$ can be recovered by $\boldsymbol{AX} = [\boldsymbol{A}_1\boldsymbol{X}; (\boldsymbol{A}_1 + \boldsymbol{A}_2)\boldsymbol{X} - \boldsymbol{A}_1\boldsymbol{X}]$.

To evaluate the performances of the uncoded and coded computations, we implement each approach to solve the matrix multiplication problem with $n = 2$. In particular, we create three containers as the worker nodes in one airborne computing platform, and one container as the master node in another platform. Each container is assigned with one CPU core. Containers on different airborne computing platforms are linked through the Loco M5 to simulate the directional-antenna based long-distance U2U communications. We then create two 800×800 random matrices, $\boldsymbol{A}$ and $\boldsymbol{X}$. Matrix $\boldsymbol{A}$ is divided equally into two sub-matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ of size 400×800. The execution time of the two approaches to compute $\boldsymbol{AX}$ is provided in Table 4, where two scenarios are evaluated. In the first scenario, only matrix multiplication is performed within each container and thus no straggler exists. In the second scenario, a CPU stress test is executed concurrently in one of the worker nodes to consume its computing resources. This worker node thus becomes a straggler. The results shown in Table 4 illustrate the robustness of the coded computation to system noises such as stragglers. In the future, we will extend this study to achieve optimal task allocation and resource management for networked airborne computing using coded computation.

5 Conclusion

In this paper, we developed a new UAS-based airborne computing platform to address the onboard computing limitations of existing UAS platforms so as to support UAS-enabled MEC and to enable more advanced UAS applications. This airborne computing platform was designed from three aspects: hardware, software, and applications. To design the hardware, we first investigated the desired features for the onboard computing hardware, and then conducted a comprehensive comparison study among state-of-the-art single-board computers to select a suitable one as the computing unit. A prototype was then designed and implemented, which not only contains the computing unit, but also hardware for UAS mobility, communications, and control. To design the software, we investigated two representative virtualisation techniques, VM using KVM and container using Docker, and evaluated their performances from various aspects. Through comprehensive experimental studies, we find that Docker outperforms KVM in most performance aspects, including computing, networking, isolation of most hardware resources, power consumption, and resource usage. Docker also successfully virtualises all CPU cores and GPU in Jetson TX2. On the other hand, KVM is more secure. Finally, we studied three real UAS applications, including UAS image processing, real-time object detection, and coded distributed computing, to demonstrate the performance, applicability and potentials of the proposed airborne computing platform.

In the future, we will investigate the integration of KVM and Docker to maximise their strengths. We will also study the KVM and Docker based live migration and distributed computing techniques to enable networked airborne computing that shares resources among multiple UAS. We will also explore UAS mobility control and computation offloading strategies [85] to optimise the computing services to ground users.

6 Acknowledgments

The authors thank National Science Foundation under grants CI-1953048, 1730589/1730675/1730570/1730325 and CAREER-1714519, for the support of this work.

7 References

[1] Casbeer, D.W., Beard, R.W., McLain, T.W., *et al.*: 'Forest fire monitoring with multiple small uavs'. Proc. of the 2005 American Control Conf., Hilton Portland, Portland, Oregon, June 2005, pp. 3530–3535

[2] Kuiper, E., Nadjm-Tehrani, S.: 'Mobility models for UAV group reconnaissance applications'. Proc. of the 2006 Int. Conf. on Wireless and Mobile Communications, Vancouver, Canada, July 2006, pp. 33–33

[3] Tomic, T., Schmid, K., Lutz, P., *et al.*: 'Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue', *IEEE Robot. Autom. Mag.*, 2012, **19**, (3), pp. 46–56

[4] Nex, F., Remondino, F.: 'UAV for 3D mapping applications: a review', *Appl. Geom.*, 2014, **6**, (1), pp. 1–15

[5] Abdulla, A.E., Fadlullah, Z.M., Nishiyama, H., *et al.*: 'An optimal data collection technique for improved utility in uas-aided networks'. Proc. of the 2014 INFOCOM, Toronto, Canada, July 2014, pp. 736–744

[6] Satici, A.C., Poonawala, H., Spong, M.W.: 'Robust optimal control of quadrotor UAVs', *IEEE Access*, 2013, **1**, pp. 79–93

[7] Fotouhi, A., Ding, M., Hassan, M.: 'Flying drone base stations for macro hotspots', *IEEE Access*, 2018, **6**, pp. 19530–19539

[8] Li, K., Ni, W., Wang, X., *et al.*: 'Energy-efficient cooperative relaying for unmanned aerial vehicles', *IEEE Trans. Mob. Comput.*, 2016, **15**, (6), pp. 1377–1386

[9] Mozaffari, M., Saad, W., Bennis, M., *et al.*: 'Mobile internet of things: can UAVs provide an energy-efficient mobile architecture?'. Proc. of 2016 Global Communications Conf. (GLOBECOM), Washington, D.C., USA, December 2016, pp. 1–6

[10] Fadlullah, Z.M., Takaishi, D., Nishiyama, H., *et al.*: 'A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks', *IEEE Netw.*, 2016, **30**, (1), pp. 100–105

[11] Motlagh, N.H., Bagaa, M., Taleb, T.: 'UAV-based IoT platform: a crowd surveillance use case', *IEEE Commun. Mag.*, 2017, **55**, (2), pp. 128–134

[12] Qureshi, B., Koubaa, A., Sriti, M.F., *et al.*: 'Poster: dronemap-a cloud-based architecture for the internet-of-drones'. Proc. of the 2016 Int. Conf. on Embedded Wireless Systems and Networks, TU Graz, Austria, 2016

[13] Im.Cho, Y., Giyenko, A.: 'Intelligent UAV in smart cities using IoT'. Proc. of 2016 16th Int. Conf. on Control,Automation and Systems (ICCAS), Gyeongju, Korea, 2016, pp. 207–210

[14] Yoo, S.J., Park, J.h., Kim, S.h., *et al.*: 'Flying path optimization in UAV-assisted IoT sensor networks', *ICT Express*, 2016, **2**, (3), pp. 140–144

[15] Hu, Y.C., Patel, M., Sabella, D., *et al.*: 'Mobile edge computing - a key technology towards 5G', *ETSI White Paper*, 2015, **11**, (11), pp. 1–16

[16] Loke, S.W.: 'The internet of flying-things: Opportunities and challenges with airborne fog computing and mobile cloud in the clouds', 2015, arXiv preprint arXiv:150704492

[17] Jeong, S., Simeone, O., Kang, J.: 'Mobile edge computing via a UAV-mounted cloudlet: optimization of bit allocation and path planning', *IEEE Trans. Veh. Technol.*, 2018, **67**, (3), pp. 2049–2063

[18] Jeong, S., Simeone, O., Kang, J.: 'Mobile cloud computing with a UAV-mounted cloudlet: optimal bit allocation for communication and computation', *IET Commun.*, 2017, **11**, (7), pp. 969–974

[19] Hua, M., Wang, Y., Li, C., *et al.*: 'UAV-aided mobile edge computing systems with one by one access scheme', *IEEE Trans. Green Commun. Netw.*, 2019, **3**, (3), pp. 664–678

[20] Xiong, J., Guo, H., Liu, J.: 'Task offloading in UAV-aided edge computing: bit allocation and trajectory optimization', *IEEE Commun. Lett.*, 2019, **23**, (3), pp. 538–541

[21] Zhang, J., Zhou, L., Tang, Q., *et al.*: 'Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing', *IEEE Internet Things J.*, 2019, **6**, (2), pp. 3688–3699

[22] Zhou, F., Wu, Y., Sun, H., *et al.*: 'UAV-enabled mobile edge computing: offloading optimization and trajectory design'. Proc. of the 2018 IEEE Int. Conf. on Communications (ICC), Kansas City, MO, USA, May 2018, pp. 1–6

[23] Zhou, F., Wu, Y., Hu, R.Q., *et al.*: 'Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems', *IEEE J. Sel. Areas Commun.*, 2018, **36**, (9), pp. 1927–1941

[24] 'Airborne Computing Networks (Project Website)'. Available at https://www.uta.edu/utari/research/robotics/airborne/, accessed 12 September 2019

[25] Dall, C., Nieh, J.: 'Kvm/arm: the design and implementation of the linux arm hypervisor', *ACM SIGARCH Comput. Archit. News*, 2014, **42**, (1), pp. 333–348

[26] Merkel, D.: 'Docker: lightweight linux containers for consistent development and deployment', *Linux J.*, 2014, **2014**, (239), p. 2

[27] 'Raspberry Pi'. Available at https://www.raspberrypi.org/, accessed 12 September 2019

[28] 'Odroid Xu'. Available at https://www.hardkernel.com/ko/tag/odroid-xu/, accessed 12 September 2019

[29] 'Arduino'. Available at https://www.arduino.cc/, accessed 12 September 2019

[30] 'Cubieboard'. Available at http://docs.cubieboard.org/products/start, accessed 12 September 2019

[31] 'Arndale Board'. Available at http://www.arndaleboard.com/us/?menuType=product&mode=list&lcate=001&mcate=001, accessed 12 September 2019

[32] Shang, Z., Shen, Z.: 'Real-time 3D reconstruction on construction site using visual slam and UAV', arXiv preprint arXiv:171207122, 2017

[33] 'Jetson TX1 Module'. Available at https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/, accessed 12 September 2019

[34] 'Jetson TX2 Module'. Available at https://elinux.org/Jetson_TX2, accessed 12 September 2019

[35] 'UDOO X86'. Available at https://www.udoo.org/udoo-x86/, accessed 12 September 2019

[36] 'Intel Aero Compute Board'. Available at https://software.intel.com/en-us/aero/compute-board, accessed 12 September 2019

[37] 'LattePanda Alpha'. Available at https://www.kickstarter.com/projects/139108638/lattepanda-alpha-soul-of-a-macbook-in-a-pocket-siz, accessed 12 September 2019

[38] 'UP Squared'. Available at https://up-board.org/upsquared/specifications/, accessed 12 September 2019

[39] 'Jetson Xavier'. Available at https://developer.nvidia.com/embedded/buy/jetson-xavier-devkit, accessed 12 September 2019

[40] 'DJI Manifold'. Available at https://www.dji.com/manifold, accessed 12 September 2019

[41] 'HiKey960'. Available at https://www.96boards.org/product/hikey960/, accessed 12 September 2019

[42] 'Rock 960'. Available at https://www.96rocks.com/, accessed 12 September 2019

[43] 'Support Resources'. Available at https://developer.nvidia.com/embedded/community/support-resources, accessed 12 September, 2019

[44] Li, S., He, C., Liu, M., *et al.*: 'The design and implementation of aerial communication using directional antennas: learning control in unknown communication environment', *IET Control Theory Applic.*, 2019, **13**, (17), pp. 2906–2916

[45] Lu, K., Xie, J., Wan, Y., *et al.*: 'Toward UAV-based airborne computing', *IEEE Wirel. Commun.*, 2019, **26**, (6), pp. 172–179

[46] 'DJI Matrice 100'. Available at https://www.dji.com/matrice100, accessed 12 September 2019

[47] 'NanoStation Loco M5'. Available at https://store.ui.com/collections/wireless/products/nanolocom5, accessed 12 September 2019

[48] 'WS323 300Mbps Wireless Range Extender User Guide'. Available at https://www.manualslib.com/manual/547195/Huawei-Ws323.html#manual, accessed 12 September 2019

[49] Kivity, A., Kamay, Y., Laor, D., *et al.*: 'kvm: the linux virtual machine monitor'. Proc. of the Linux symp., Ottawa, Ontario, July 2007, vol. 1

[50] Yoon, M.K., Liu, B., Hovakimyan, N., *et al.*: 'Virtualdrone: virtual sensing, actuation, and communication for attack-resilient unmanned aerial systems'. Proc. of the 8th Int. Conf. on Cyber-Physical Systems, Pittsburgh, PA, April 2017, pp. 143–154

[51] Jain, N., Choudhary, S.: 'Overview of virtualization in cloud computing'. 2016 Symp. on Colossal Data Analysis and Networking (CDAN), Indore, India, March 2016, pp. 1–4

[52] Seo, K.T., Hwang, H.S., Moon, I.Y., *et al.*: 'Performance comparison analysis of linux container and virtual machine for building cloud', *Adv. Sci. Technol. Lett.*, 2014, **66**, (2), pp. 105–111

[53] Malhotra, L., Agarwal, D., Jaiswal, A.: 'Virtualization in cloud computing', *J. Inf. Technol. Softw. Eng.*, 2014, **4**, (136), p. 2

[54] Shuja, J., Gani, A., Bilal, K., *et al.*: 'A survey of mobile device virtualization: taxonomy and state of the art', *ACM Comput. Surv. (CSUR)*, 2016, **49**, (1), pp. 1–36

[55] Morabito, R.: 'A performance evaluation of container technologies on internet of things devices'. Proc. of the 2016 IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, April 2016, pp. 999–1000

[56] Patel, A., Daftedar, M., Shalan, M., *et al.*: 'Embedded hypervisor xvisor: a comparative analysis'. Proc. of the 2015 23rd Euromicro Int. Conf. on Parallel,Distributed and Network-Based Processing (PDP), Turku, Finland, March 2015, pp. 682–691

[57] Gu, F., Hu, F., Chen, H.: 'Real-time performance evaluation of linux arm virtualization'. Proc. of the 2nd Int. Conf. on Energy Science and Applied Technology (ESAT 2015), Wuhan, China, August 2015

[58] Toumassian, S., Werner, R., Sikora, A.: 'Performance measurements for hypervisors on embedded arm processors'. Proc. of 2016 Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, September 2016, pp. 851–858

[59] Raho, M., Spyridakis, A., Paolino, M., *et al.*: 'KVM, Xen and docker: a performance analysis for arm based nfv and cloud computing'. Proc. of the 2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Riga, Latvia, November 2015, pp. 1–8

[60] Morabito, R., Beijar, N.: 'Enabling data processing at the network edge through lightweight virtualization technologies'. Proc. of 2016 IEEE Int. Conf. on Sensing, Communication and Networking (SECON Workshops), London, UK, June 2016, pp. 1–6

[61] Morabito, R.: 'Virtualization on internet of things edge devices with container technologies: a performance evaluation', *IEEE Access*, 2017, **5**, pp. 8835–8850

[62] Bellavista, P., Zanni, A.: 'Feasibility of fog computing deployment based on docker containerization over raspberrypi'. Proc. of the 18th Int. Conf. on Distributed Computing and Networking, NY, USA, January 2017, p. 16

[63] Baccarelli, E., Naranjo, P.G.V., Scarpiniti, M., *et al.*: 'Fog of everything: energy-efficient networked computing architectures, research challenges, and a case study', *IEEE Access*, 2017, **5**, pp. 9882–9910

[64] 'Edge Computing for UAVs, UASs, and Drones'. Available at https://www.nutanix.com/go/edge-computing-for-drones.html, accessed 12 September 2019

[65] Kalatzis, N., Avgeris, M., Dechouniotis, D., *et al.*: 'Edge computing in IoT ecosystems for UAV-enabled early fire detection'. Proc. of 2018 IEEE Int. Conf. on Smart Computing (SMARTCOMP), Kuala Lumpur, Malaysia, July 2018, pp. 106–114

[66] Wang, B., Xie, J., Li, S., *et al.*: 'Enabling high-performance onboard computing with virtualization for unmanned aerial systems'. Proc. of 2018 Int. Conf. on Unmanned Aircraft Systems (ICUAS), Dallas, TX, June 2018, pp. 202–211

[67] Che, S., Boyer, M., Meng, J., *et al.*: 'Rodinia: a benchmark suite for heterogeneous computing'. Proc. of IEEE Int. Symp. on Workload Characterization, Austin, TX, October 2009, pp. 44–54

[68] Montella, R., Giunta, G., Laccetti, G., *et al.*: 'Virtualizing CUDA enabled GPGPUs on arm clusters'. Proc. of the 12th Int. Conf. on Parallel Processing and Applied Mathematics, Krakow, Poland, 2016, pp. 3–14

[69] Kang, D., Jun, T.J., Kim, D., *et al.*: 'Convgpu: GPU management middleware in container based virtualized environment'. Proc. of the 2017 IEEE Int. Conf. on Cluster Computing (CLUSTER), Hawaii, USA, September 2017, pp. 301–309

[70] Tirumala, A., Dunigan, T., Cottrell, L.: 'Measuring end-to-end bandwidth with IPERF using web100'. Proc. of Passive and Active Monitoring Workshop, San Diego, CA, April 2003

[71] Wei, Z., Xiaolin, G., Wei, H.R., *et al.*: 'TCP DDoS attack detection on the host in the kvm virtual machine environment'. Proc. of the 2012 IEEE/ACIS 11th Int. Conf. on Computer and Information Science (ICIS), Shanghai, China, May 2012, pp. 62–67

[72] Mabry, R., Ardonne, J., Weaver, J.N., *et al.*: 'Maritime autonomy in a box: building a quickly-deployable autonomy solution using the docker container environment'. Proc. of OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, September 2016, pp. 1–6

[73] Xavier, M.G., Neves, M.V., Rossi, F.D., *et al.*: 'Performance evaluation of container-based virtualization for high performance computing environments'. Proc. of the 2013 21st Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing (PDP), Belfast, UK, February 2013, pp. 233–240

[74] Matthews, J.N., Hu, W., Hapuarachchi, M., *et al.*: 'Quantifying the performance isolation properties of virtualization systems'. Proc. of the 2007 Workshop on Experimental Computer Science, San Diego, CA, June 2007, p. 6

[75] Bailey, D.H.: '*Nas parallel benchmarks*' (Springer, Boston, MA, US, 2011), pp. 1254–1259

[76] 'Jetson TX2 Thermal Design Guide'. Available at https://devtalk.nvidia.com/default/topic/1036126/measure-jetson-x2-ener gy-usage-during-a-given-task/, accessed 12 September 2019

[77] 'Sysstat'. Available at http://sebastien.godard.pagesperso-orange.fr/, accessed 12 September 2019

[78] Yu, C., Huan, F.: 'Live migration of docker containers through logging and replay'. Proc. of the Int. Conf. on Mechatronics and Industrial Informatics Advances in Computer Science Research, Zhuhai, China, October 2015

[79] 'Checkpoint and Restore'. Available at https://criu.org/Main_Page, accessed 12 September 2019

[80] 'Docker'. Available at https://www.docker.com/, accessed 12 September 2019

[81] 'OpenDroneMap'. Available at http://opendronemap.org/, accessed 12 September 2019

[82] 'Jetson inference'. Available at https://github.com/dusty-nv/jetson-inference, accessed 12 September 2019

[83] 'UCF Aerial Action Data Set'. Available at http://crcv.ucf.edu/data/UCF_Aerial_Action.php, accessed 12 September 2019

[84] Lee, K., Lam, M., Pedarsani, R., *et al.*: 'Speeding up distributed machine learning using codes', *IEEE Trans. Inf. Theory*, 2018, **64**, (3), pp. 1514–1529

[85] Zheng, J., Cai, Y., Wu, Y., *et al.*: 'Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach', *IEEE Trans. Mob. Comput.*, 2018, **18**, (4), pp. 771–786