Real-time UAV Resource Monitoring and Alerts with Automated Control Mechanism

Nisarg Parekh*, Rajashekhar Reddy Tella†, Lavanya Malakalapalli †, Praveen Tammana‡, Koteswararao Kondepu†
† Department of Computer Science and Engineering, IIT Dharwad, Dharwad, India
* Department of Smart Mobility, IIT Hyderabad, Hyderabad, India
‡ Department of Computer Science and Engineering, IIT Hyderabad, Hyderabad, India
Email: {200030058, k.kondepu}@iitdh.ac.in

***Abstract*—Autonomous navigation in Unmanned Aerial Vehicles (UAVs) is a crucial and rapidly advancing field with a wide range of applications. It utilizes various deep learning (DL) algorithms that enable UAVs to operate and navigate autonomously, reducing the need for direct human intervention. However, the use of DL algorithms demands significant computational resources, which can lead to resource constraints, potentially disrupting ongoing operations and compromising UAV safety. This work focuses on establishing a real-time monitoring, alerting, and automated control framework for UAV system resources. The proposed framework alerts the user/operator if system resource utilization exceeds the threshold. Furthermore, the automated controlling mechanism guarantees that resource consumption is adequately monitored and kept below predefined boundaries, lowering the danger of resource depletion and UAV damage. The suggested framework provides a powerful and efficient solution for optimal utilization of the available UAV system resources. By tackling the critical issue of system resource consumption, the suggested framework helps to achieve the larger goal of enabling the widespread adoption of autonomous navigation.**

***Index Terms*—UAV resources monitoring, Alerting Resources, Automated Control Mechanism, Prometheus, Grafana, Kafka**

I. Introduction

The increasing interest in autonomous navigation has paved the way for various applications, ranging from autonomous driving to drone delivery systems and drone swarming [1]. Unmanned Aerial Vehicles (UAVs), known as autonomous drones, have become invaluable tools in addressing critical humanitarian needs, such as delivering essential medicines to remote areas to aid in disaster relief operations [2] and traffic surveillance. In order to accomplish these tasks successfully, UAVs rely on access to satellite positioning system information and the ability to actively learn about their surroundings through image inference to avoid obstacles and navigate safely.

However, performing real-time or near-real-time inference with computationally intensive algorithms like You Only Look Once (YOLO) [3] necessitates significant resources. Mounting the required resources directly onto the UAV poses several challenges as there is a limitation on the payload capacity of the UAV. Firstly, these power-hungry algorithms consume precious battery power, diverting it from the primary task of sustaining the UAV's flight and secondly, the additional weight of the resources further reduces the UAV's flight duration, compromising its overall efficiency and effectiveness of the missions.In order to carry out these power-hungry applications on the UAV, real-time monitoring of resource consumption is a must, if ignored it might damage the UAV resources by over-utilizing them.

Both Monitoring and alerting have recently received more attention, as there is an expansion of information acquired from devices and processed at the other end [4]. Monitoring detects excessive resource utilization and allows for prompt notification to the appropriate team (i.e, UAV operator/User). This proactive approach guarantees that any anomalies (i.e., a rapid increase in power consumption) are addressed as soon as possible to avoid negative impacts on the UAV resources.

By monitoring the resources of UAVs in real-time, we can address the following objectives:

- *Efficient Resource Management*: Continuously track the resource usage of each drone so that we can maximize the utilization of resources and minimize downtime
- *Prevent Damage and Loss*: Detect potential system failures proactively and take corrective actions to avoid collisions and minimize the risk of damage to the drones and their surroundings

A work in [5] develops the monitoring and alerting framework for the UAV resources without employing the controlling mechanism to take towards the alerts. The authors in [6] used Grafana [7] and Zabbix to provide alerts on the failures that occur in wireless networks for internet service access. In this work, we investigate the notion of real-time resource monitoring, alerting, and automatically controlling resources of UAVs. Here, an alert-based control mechanism capable of successfully managing resource usage is created when it surpasses a predefined threshold; therefore preventing collisions and minimizing the risk of harm to drones and their surroundings

II. System Model

Fig. 1 shows the architecture of UAV resources monitoring and automated controlling mechanism of resources. The framework mainly consists of an *UAV cluster* and an *Edge cluster*. The *UAV cluster* contains multiple UAVs that can be useful for achieving the desired tasks. The *Edge cluster* has the Prometheus server [8] which scrapes the system metrics from

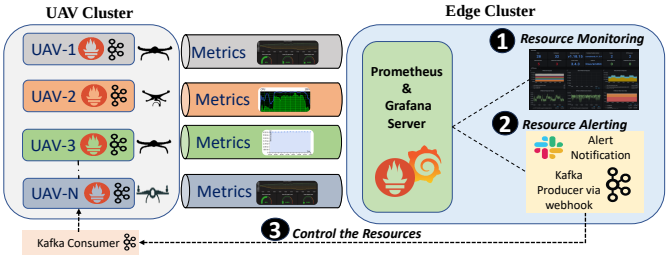

Fig. 1. System Architecture

the node exporter deployed on the UAVs at the *UAV cluster*. A detailed description of the clusters and their components is as follows:

UAV cluster: This cluster contains the UAVs that are mounted with Raspberry Pi (RPi) boards — to carry out various tasks like object detection, object tracking, video recording, etc. The following activities are performed at the UAV cluster: (i) node exporter is deployed in the RPi board to continuously expose the system resource usage metrics to the Prometheus server deployed at the Edge for monitoring; (ii) metrics from the Node exporter can be scraped based on predefined interval time; (iii) the requests from the applications running on the UAVs are processed, and corresponding actions are performed by the UAV cluster [5]; (iv) the Kafka consumer is integrated into the UAV cluster, thus the system can efficiently process alerts and automatically implement corrective measures to maintain resource usage within acceptable limits.

Edge Cluster: The Edge cluster contains the proposed monitoring framework with the following components: (i) **Prometheus**: It is an open-source tool that is used to monitor the system metrics of the defined target. It uses an HTTP pull model for time series collection, It can be used to monitor multiple targets. *Prometheus* provides comprehensive querying capabilities and quick alerting functionalities as crucial components for edge cluster monitoring; (ii) **Grafana**: Grafana is an open-source solution for monitoring and visualization. Prometheus scrapes the UAV cluster's resource metrics in the context of the edge cluster and it is added as a data source in Grafana. Grafana has in-built dashboards that can be used to view the parameters/system metrics in a more facile and straightforward way; (iii) **Alert Manager**: Grafana has an in-built Alert Manager that allows to customize alert rules and add notification channels. It sends the desired alerts to the configured channels when the resource being monitored is used more than the desired value. Notifications can be given across numerous channels such as Slack, Email, etc. ensuring rapid response and intervention as required; (iv) **Kafka**: Kafka is a distributed streaming technology that allows for the publication and subscription of record streams [9]. Kafka acts as a messaging system within the edge cluster, processing alerts created by the alert manager. It ensures dependable and flexible communication between system components.

The UAV clusters are deployed with the node exporters as mentioned above and expose the system metrics to the Prometheus which is running on the Edge cluster. The Prometheus scrapes the metrics periodically and sends them to the Grafana server for visualization and alerting. The Grafana is configured with alerting rules, predefined thresholds, and notification channels (Fig. 1). A Kafka topic is created on the Edge cluster which acts as a producer and the created topic is subscribed by the UAVs making them the consumer. A webhook (i.e., an HTTP callback for real-time communication between different services) is created between the Grafana-based alert manager and the Kafka producer that is running on the Edge cluster. Grafana sends an alert to the slack channel and webhook when any monitored resource usages exceed the threshold. Upon receiving the alert, the webhook on the Edge cluster writes it to the topic, which is read on the Kafka consumer side. After receiving the alerts, the Kafka consumer checks which resource is being over-utilized and takes a control action based on that. The Kafka consumer action may involve interrupting the program that might be consuming the highest CPU usage or the user can decide based on the importance of the programs (or jobs) that are running in the system. In addition, instead of complete termination of running jobs, the user could decide to pause certain tasks or reallocate or reschedule the jobs to later in time, depending on the availability of the system resources. All this automated controlling mechanism is shown in Fig. 2.

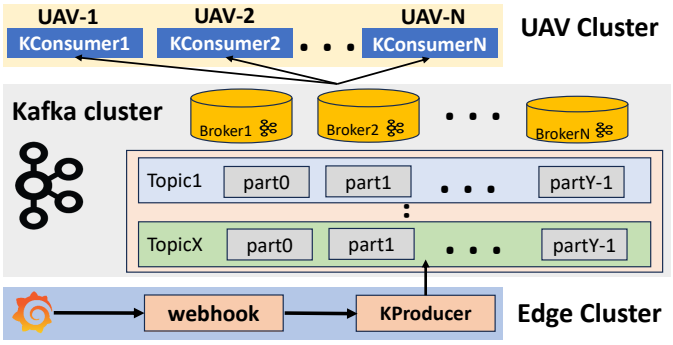

Fig. 2. Automated controlling mechanism using Kafka cluster

The following performance metrics are evaluated in the below experiments:

- *CPU Usage:* It is a metric representing the CPU utilization degree. It's an indicator of how much computational work the CPU is currently performing relative to its maximum capacity.
- *RAM Usage:* It refers to the amount of Random Access Memory (RAM) that is currently being utilized or consumed by a system, process, or application at a given moment.

III. Experimental Results

A. Resource Monitoring

The resource consumption of UAVs can vary depending on their operational mode and the algorithms they are running. As shown in Fig. 3, we can observe a progressive increase in CPU usage as the UAV carries out different tasks. In the experimental setup, the UAV was initially in an idle state, with CPU usage hovering around 3%. Subsequently,

Task#1, involving autonomous take-off and propeller activation, led to a significant spike in CPU usage, reaching 56%. Following that, Task#2 involved recording and transmitting video to an Edge cluster, increasing CPU usage to 78%. Finally, Task#3 involved onboard object detection using the YOLOv3 algorithm. This task pushed the CPU usage to its peak at 98%, triggering an alert. In response to this alert, all running processes were immediately aborted, bringing the CPU usage back to its initial idle state of 3%. The observed

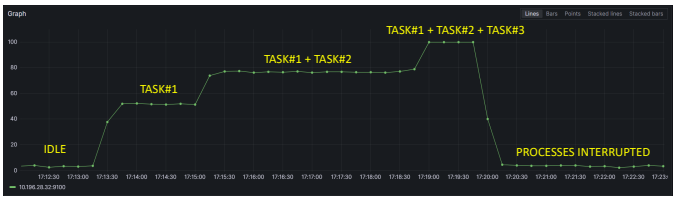

Fig. 3. Grafana Dashboard - Showing Resource Usage for Tasks Execution

substantial difference of approximately 50% in CPU usage when transitioning from IDLE to FLYING mode highlights the significant impact of algorithm execution on resource consumption. This information through monitoring enables us to optimally decide the thresholds for alerting and efficiently use the available UAV system resources.

B. Resources Alerting

Monitoring the resources can help the operator (i.e., the user) to know the status of system resources and take corresponding action. We created a sample alert rule with a threshold value of 80% CPU usage to trigger an alert when CPU usage exceeds the pre-defined threshold value. To simulate a scenario where the CPU usage surpasses the threshold, the propellers of a UAV are powered on and initiate the task of object detection on captured frames.

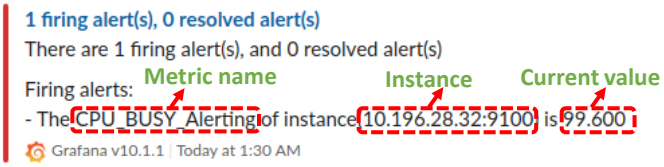

Fig. 4. An example of slack alerts generation with Grafana-based alert manager

The Fig. 4 depicts the Slack alert received from the alert manager, providing details such as the alert name, instance, and the actual value of the metric usage.

C. Automated Control Mechanism

In order to handle the automated controlling of the resources, the thresholds are defined in two different levels, namely (i) *warning* alert — where the alert is generated when the utilization exceeds 60%, which does not require any action to be taken — and (ii) *critical* alert — where the alert is generated when the utilization exceeds 80% and require a strict action to reduce the resource usage from the some of the running processes (i.e, jobs). Depending on the above-defined threshold alert values, the Kafka consumer (or the UAV) takes appropriate action to control the system resources.

Upon the consumer takes action, resource usage is expected to decrease, eventually falling back within the predefined threshold limits.

Simultaneously, the alert manager continues to monitor the resources and detects when the resource usage is back to normal within the pre-defined threshold values. Once this condition is reached, the alert manager publishes a *resolved message* to the Kafka topic. The Kafka consumer then consumes the resolved message. Based on its value, the user might decide to resume the paused jobs if possible or adjust its behavior back to normal operation.

Fig. 5 shows both the *warning*, *critical*, and the *resolved* alert received at the Kafka consumer.

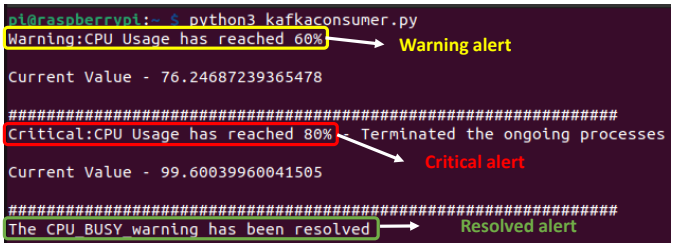

Fig. 5. Alert received at Kafka Consumer

IV. Conclusion and Future Work

The UAV resource monitoring and alert system offers a full resource management solution for UAV operations. The system not only detects anomalies but also provides a warning alert. It also includes an automated control mechanism to balance system resources. The proposed UAV framework is capable of monitoring several UAVs at the same time, which is particularly useful in scenarios incorporating swarms of drones or massive amounts of UAV deployments. The measured automated control response time is 10μs — the time taken to act upon the Kafka consumer receiving a resource control critical alert from the producer. The potential future work of this paper is to build a framework that uses a Machine Learning approach like GANs or Reinforcement Learning and provides the alerts beforehand based on the forecasted data.

References

[1] E. Frachtenberg, "Practical Drone Delivery," *Computer*, vol. 52, no. 12, pp. 53–57, 2019.

[2] B. P. A. Rohman, M. B. Andra *et al.*, "Multisensory Surveillance Drone for Survivor Detection and Geolocalization in Complex Post-Disaster Environment," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 9368–9371.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] K. Hitchcock, *Monitoring*. Berkeley, CA: Apress, 2022, pp. 203–240.

[5] Y. C. Makkena, R. R. Tella, N. Parekh *et al.*, "Experience: Implementation of Edge-Cloud for Autonomous Navigation Applications," in *in Proc. 15th COMSNETS*, 2023, pp. 579–587.

[6] A. R. H. Velasco, E. E. G. Malla, R. D. C. C. Herrera, and F. D. M. Arévalo, "Real-time monitoring and alerting system using zabbix and grafana software for wireless internet access service management." in *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, 2023, pp. 1–6.

[7] Grafana. [Online]. Available: https://github.com/grafana/grafana

[8] Prometheus. [Online]. Available: https://prometheus.io

[9] Apache Kafka. [Online]. Available: https://kafka.apache.org/