Optimizing Power Efficiency in Low-Power IoT Devices Through Containerization

Nicholas Joseph Saliba
Institute of Information & Communication Technology
Malta College or Arts, Science & Technology
Corradino Hill
Paola PLA 9032

nicholas.saliba.f32090@mcast.edu.mt

***Abstract*—This research investigates the role of container orchestration in enhancing power efficiency on low-power IoT devices, using a Raspberry Pi as the experimental platform. The study compares the performance of compute-intensive scripts executed directly on the system with their orchestrated counterparts deployed through Docker Swarm. Key metrics such as power consumption, CPU utilization, and device temperature were recorded and compared. The findings indicate that orchestration provides noticeable improvements in resource management, resulting in reduced power usage and thermal output. These results align with existing studies in the field and support the hypothesis that container orchestration can optimize workload efficiency on constrained edge devices. The research offers a practical, replicable evaluation framework and lays the groundwork for future studies exploring scalable and energy-aware IoT deployments.**

***Index Terms*—IoT, Containers, Orchestration Platforms, Power Efficiency, Edge Computing**

I. Introduction

The growing adoption of Internet of Things (IoT) devices has brought about increasing attention to their energy consumption and operational efficiency, particularly in environments with limited power resources. As these devices are often deployed in edge settings or remote locations, ensuring optimal energy use without sacrificing performance is critical. One promising approach to improving resource efficiency in such systems is through the use of containerization and orchestration technologies, which provide lightweight, modular deployment options for managing distributed workloads. This research is driven by the need to enhance the power efficiency of low-power IoT devices, such as the Raspberry Pi, when executing compute-intensive tasks. While containerization has become a standard practice in cloud and enterprise environments, its impact on energy usage and performance in constrained edge systems remains underexplored. This project investigates whether orchestrated containerized deployments can reduce power consumption compared to traditional standalone execution methods.

A. Hypothesis

The research hypothesis states that power efficiency will improve significantly due to container orchestration platforms' ability to manage resources dynamically..

B. Research Aim and Purpose Statement

The aim of this research is to optimize power efficiency in low-power IoT devices through containerization, specifically by evaluating the effectiveness of orchestration platforms such as Docker Swarm. To examine this, the study addresses the following research question: How does containerization affect power consumption in IoT applications?

II. Literature Review

Deploying containers on Internet of Things (IoT) devices introduces a highly effective approach towards improving the devices' power efficiency. As IoT adoption accelerates across agriculture, healthcare, and smart infrastructure, optimizing the performance of resource-constrained edge devices becomes essential. Traditional deployment techniques often introduce inefficiencies due to lack of modularity, inflexible scaling, and the overhead of virtual machines. Containerization promises improved deployment workflows, dynamic resource allocation, and potential reductions in processing latency and power consumption. Containerisation is lightweight and can also be orchestrated for low power consumption and load-balancing, which makes it a highly viable solution for this scenario. This study builds upon the foundation of containerization's effect on IoT devices by conducting a thorough review of the body of existing literature and technological frameworks, as cited below. It also emphasises four respective themes being containerized IoT, Container orchestration, Power efficiency and low power constrained IoT devices, which have been covered below respectively.

Šimon et al. [1] present a comparative study of high availability container infrastructures focused on Docker, Kubernetes, and Proxmox. The primary aim of the research is to evaluate the resilience and performance of these container technologies under failure scenarios in Linux environments. The paper does not employ a specific public dataset but rather relies on empirical testing using ApacheBench for performance metrics including service recovery time, transfer rate, and failure rates. The solution uses direct benchmarking with container orchestrators, evaluating response under stress and failover scenarios. Docker Swarm outperforms Kubernetes and Proxmox in several scenarios, demonstrating faster recovery times and fewer failed requests. One limitation is that the experiments were conducted only on Linux platforms with no specific analysis of energy consumption or hardware variety. The authors recommend Docker Swarm for lightweight and resilient deployments and suggest future work on container orchestration efficiency in edge computing environments.

Pearson and Plante [2] expand on this by proposing a secure deployment framework for containerized IoT applications using Docker, with Raspberry Pi devices serving as the low-power hardware platform. The study does not use a dataset but focuses on a practical testbed consisting of three Raspberry Pi 3 devices and a Linux-based server. The solution includes secure OTA updates through Docker registries and encrypted communication using SSH and TLS. Evaluation emphasizes modular container updates, network efficiency, and basic deployment scalability. Their research aims to simplify secure provisioning and updates for edge devices in a cost-effective manor, in which Docker's lightweight container model has excelled for decentralized, cost-sensitive IoT systems and comes highly recommended by the authors.

Alam et al. [3] aim to develop a modular, edge-fog-cloud distributed architecture using Docker for microservice orchestration. The framework distributes services among edge, fog, and cloud layers using 3 Raspberry Pi devices. While the paper does not focus on a formal dataset, it evaluates system latency, deployment reliability, and fault tolerance in a smart home scenario. Their Docker Swarm-based approach ensures automatic failover and dynamic service relocation. The results indicate that by using containerized microservices, computation power and latency is decreased in IoT deployments. Comparatively, Figueroa et al. [4] introduce a Kubernetes-based management system for IoT container orchestration. The goal of the paper is to automate deployment, improve availability, and simplify update management in smart homes. Their system combines Raspberry Pi edge devices with cloud-hosted components and employs MQTT for lightweight communication. System availability and auto-recovery behavior is monitored to evaluate the orchestration. A noted limitation is the high overhead introduced by Kubernetes, which may be unsuitable for extremely constrained edge nodes. The paper recommends hybrid architectures where Kubernetes manages cloud and fog tiers, complemented by lighter edge-level tools.

In another study proposed by Wang et al. In [5], the focus lies on deploying container orchestration in hybrid fog-cloud environments for real-time IoT applications. Their primary contribution is the integration of FogBus2 with K3s to build lightweight, distributed container clusters. The study does not rely on a dataset but uses simulated Raspberry Pi Zero-class virtual machines with constrained resources. Their solution includes three orchestration strategies: host networking, proxy servers, and environment variables. Evaluations show up to 29 percent improvement in response time and minimal orchestration overhead compared to full Kubernetes. Limitations include simulated hardware rather than physical Raspberry Pi testing and no direct measurement of energy use.

Kaiser et al. [6] propose a hybrid orchestration strategy combining containers with uni-kernels to improve power efficiency in edge computing. The research goal is to optimize system resource consumption without sacrificing performance, specifically on ARM-based Raspberry Pi 4 hardware. They deploy several containerized workloads using Docker, Podman, and Singularity, and compare these to unikernel deployments using Unikraft and OSv. The evaluation includes CPU and memory usage, and execution time for various workloads including computer vision and data analytics. Limitations include limited orchestration support for unikernels and complexity in unikernel compilation and integration. Low-power IoT devices such as Raspberry Pi and Raspberry Pi Zero have been widely adopted for their cost-effectiveness and adaptability in constrained environments, but their resource limitations necessitate highly optimized deployment strategies. Pearson and Plante [2] demonstrate that Docker-based containerization on Raspberry Pi enables modular, secure application deployment with minimal overhead, emphasizing OTA updates and encrypted communication as key features enhancing resource efficiency. Alam et al. [3] further show that distributing microservices across edge, fog, and cloud layers reduces reliance on cloud infrastructure and allows local processing, thereby conserving energy and lowering latency. Wang et al. [5] validate these findings in a simulated Raspberry Pi Zero environment, where lightweight orchestration using K3s and FogBus2 enables real-time workloads to run within tight CPU and memory constraints. Kaiser et al. [6] complement this by comparing traditional containers to unikernels on Raspberry Pi 4, reporting that unikernels use up to 36.62 percent less memory and 40 percent less CPU, making them a viable choice for enhancing energy efficiency in lightweight IoT applications.

The reviewed literature highlights significant progress in optimizing container orchestration and energy-aware design for low-power IoT systems. Kaiser et al. [6] demonstrate that unikernels offer lower memory and CPU consumption compared to containers for lightweight tasks, although they introduce complexity in orchestration. Pearson and Plante [2], Alam et al. [3], and Wang et al. [5] show that Docker and K3s can be deployed efficiently on Raspberry Pi and other low-resource devices, enhancing security and performance. When comparing orchestration solutions, Docker Swarm offers simplicity and lightweight operation [3], whereas K3s provides better scaling and performance in hybrid deployments [5]. Kubernetes, while effective in large-scale applications [4], requires optimization before use in constrained edge environments. Despite these advances, the field still lacks a unified benchmarking framework for energy efficiency across orchestration tools. Many studies rely on testbed or simulated environments without standardized energy metrics. Extending their lightweight orchestration model to physical testbeds and integrating more power-aware scheduling strategies is recommended in [5]. Future research should aim to define consistent power benchmarks and explore the integration of orchestration layers with profiling tools for energy optimization.

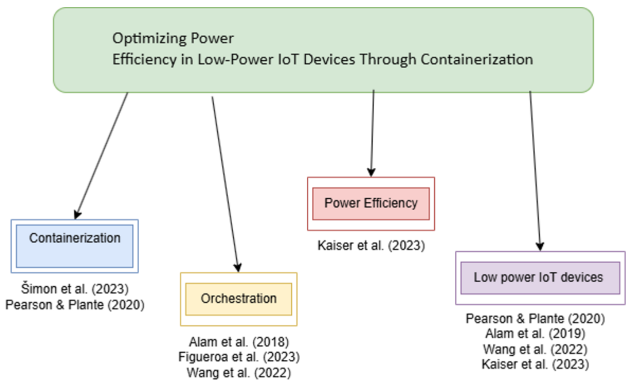

Fig. 1. Literature Map

III. Research Methodology

A. Pipeline

Phase 1 This phase involves initial investigation into low-power IoT environments and container-based technologies. Early literature is reviewed to determine the research gap in energy optimization through containerization and orchestration. From this, the research methodology is refined. Final outputs include a detailed research plan outlining hardware/software requirement, and a comparison framework to assess containerized vs. non-containerized scenarios.

Phase 2 This phase sets up two parallel environments: one containerized, the other non-containerized. Orchestration platforms are configured. Applications simulating IoT sensor workloads are deployed. Power/resource monitoring tools are configured alongside a power metering setup. This ensures repeatable experiments can be conducted under similar conditions for both environments.

Phase 3 Experiments are conducted to measure performance and energy metrics across both environments. Workloads are evaluated under idle, typical, and peak conditions. The non-containerized and containerized setups are tested separately using a power meter.

Phase 4 Logs and metrics from all experiments are systematically collected. These include resource usage over time, and power consumption. The data is organized based on each test scenario and visualized using infographics and charts to clearly compare the performance of containerized vs. non-containerized IoT environments.

Phase 5 This phase consists of the data collected in Phase 4 being evaluated for the purpose of answering the Research questions initially proposed. Analysis will take place to determine if the research objective has been satisfied, and any improvements or limitations will be identified accordingly.

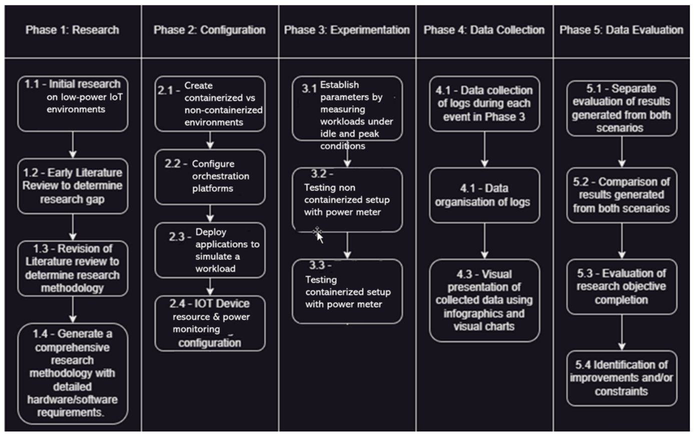
Phase 1: Research
Phase 2: Configuration
Phase 3: Experimentation
Phase 4: Data Collection
Phase 5: Data Evaluation
1.1 - Initial research on low-power IoT environments
1.2 - Early Literature Review to determine research gap
1.3 - Revision of Literature review to determine research methodology
1.4 - Generate a comprehensive research methodology with detailed hardware/software requirements.
2.1 - Create containerized vs non-containerized environments
2.2 - Configure orchestration platforms
2.3 - Deploy applications to simulate a workload
2.4 - IOT Device resource & power monitoring configuration
3.1 Establish parameters by measuring workloads under idle and peak conditions
3.2 - Testing non containerized setup with power meter
3.3 - Testing containerized setup with power meter
4.1 - Data collection of logs during each event in Phase 3
4.1 - Data organisation of logs
4.3 - Visual presentation of collected data using infographics and visual charts
5.1 - Separate evaluation of results generated from both scenarios
5.2 - Comparison of results generated from both scenarios
5.3 - Evaluation of research objective completion
5.4 Identification of improvements and/or constraints

Fig. 2. Pipeline

B. Approach

An experiment related to the research objective will be conducted to determine availability variables in both scenarios and provide a cross comparison of the two. Containerized applications with functional IOT capabilities such as data logging and periodic synchronisation will be configured on each virtual worker node to simulate a containerized IOT Device. Both configurations will be handling hashes to simulate a typical working load. A power meter will be used to monitor and collect the data. The data generated from these experiments is quantitative in nature, as it deals with values such as Power (W), CPU Usage and CPU Temperature, which can all be objectively quantified. Various testing will be conducted to determine power efficiency differences between containerized and non-containerized low-power IoT environments. The study will focus on resource-constrained devices as virtual IoT nodes. Two environments will be compared: one containerized using orchestration platforms, and the other running natively without container overhead. Each configuration will be deployed with python hashing applications simulating and processing tasks typically found in low-power IoT systems. Power consumption will be estimated using either external power meters or internal monitoring where supported. Both setups will operate under controlled idle, active, and peak load conditions to simulate real-world IoT activity. The data collected will be quantitative, focusing on power consumption (W) and efficiency (percent) metrics, and CPU Temp (degrees). These metrics will allow objective evaluation of which configuration is more energy efficient under equivalent workloads.

C. Ethical considerations

All materials used digitally are available to the public and were all collected from free online repositories or public use APIs, hence, no GDPR or copyright infringement is applicable. All materials used physically are owned by the researcher and have the same implications.

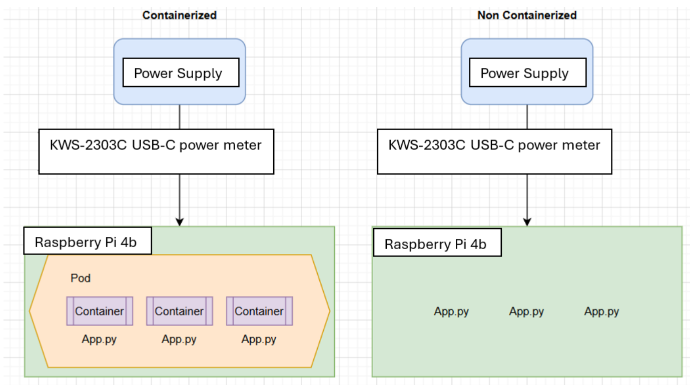

Fig. 3. Diagram

IV. Findings & Discussion of Results

This study investigated the power efficiency of container orchestration on low-power IoT devices by comparing standalone execution of Python scripts to their orchestrated deployment using Docker Swarm on a Raspberry Pi. The results showed a consistent difference in power usage, CPU load, and thermal output between the two configurations.

On average, the standalone setup consumed 7.00 W, utilized 98.94% of the CPU, and maintained an operating temperature of 61.77°C. In contrast, the Docker Swarm deployment showed improved resource management, averaging 5.97 W in power consumption, 63.23% CPU usage, and a reduced temperature of 59.82°C. These results confirm that container orchestration provides a tangible benefit in power efficiency and thermal regulation, particularly for long-running or concurrently executed workloads.

The observed results suggest that the orchestrated environment effectively distributed and managed the script execution load, minimizing resource contention and unnecessary spikes in CPU cycles. This supports our research hypothesis that containerization and orchestration improve the availability and efficiency of IoT workloads.

B. Hypothesis Reflection

These findings also align with the conclusions of researchers such as Kaiser et al., who demonstrated that combining orchestration with lightweight virtualization technologies like containers and unikernels led to better energy profiles on constrained devices. Similarly, Wang et al. highlighted the role of orchestration in optimizing edge computing tasks for real-time efficiency. Compared to these studies, this research contributes a unique perspective by measuring actual power usage on physical hardware rather than relying on simulated data or theoretical models.

Time (s)	Power (W)	CPU Usage (%)	Temperature (°C)
0	6.095	74	59.9
2	6.03	59	58.9
4	5.71	59	59.9
6	6.373	73	60.8
8	5.81	59	60.3
10	5.92	59	59.4
12	6.088	74	59.4
14	5.783	59	58.9
16	5.7	60	59.9
18	6.118	73	59.9
20	5.807	60	60.3
22	5.72	59	60.3
24	6.401	71	60.3
26	6.283	63	60.8
28	5.777	59	61.3
30	5.878	67	59.4
32	5.986	66	59.4
34	5.747	59	59.4
36	6.146	62	60.8
38	6.034	71	59.9
40	5.697	60	62.3
42	5.841	60	60.8
44	6.021	73	60.3
46	5.729	59	59.4
48	5.911	59	60.8
50	6.391	74	59.4
52	6.093	59	50.9
54	6.008	60	59.4
56	6.22	50	60.3
58	5.786	60	60.8
60	5.935	60	60.8

TABLE I
ORCHESTRATED RESULTS OVER 1 MINUTE

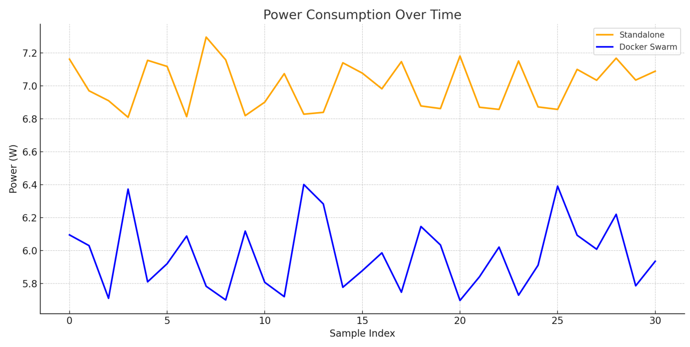

Fig. 4. Power Draw

In summary, the hypothesis was affirmed through empirical data, confirming that container orchestration contributes to improved power efficiency in low-power IoT environments. The comparison with prior research further reinforces the relevance and applicability of orchestration strategies in energy-sensitive deployments.

V. Conclusion

This research has demonstrated that using container orchestration, specifically through Docker Swarm, can yield measurable improvements in power efficiency when managing concurrent workloads on low-power IoT devices such as the Raspberry Pi. The orchestrated deployment consistently

Time (s)	Power (W)	CPU Usage (%)	Temperature (°C)
0	7.162	99	58.4
2	6.97	100	58.9
4	6.91	96	60.3
6	6.809	98	59.9
8	7.155	100	60.3
10	7.118	100	60.3
12	6.813	100	60.3
14	7.296	99	60.3
16	7.158	100	60.3
18	6.819	100	60.3
20	6.901	98	60.3
22	7.074	99	60.3
24	6.828	99	60.8
26	6.839	97	61.3
28	7.14	99	62.8
30	7.077	99	61.3
32	6.982	99	61.8
34	7.147	99	62.3
36	6.878	99	63.3
38	6.862	99	62.3
40	7.181	99	62.3
42	6.87	99	63.7
44	6.857	99	62.8
46	7.151	99	63.7
48	6.872	100	63.3
50	6.857	97	64.2
52	7.1	99	63.3
54	7.034	99	63.3
56	7.168	99	64.2
58	7.035	99	63.7
60	7.089	99	64.7

TABLE II
STANDALONE RESULTS OVER 1 MINUTE

outperformed the standalone execution in key metrics, including lower power consumption, reduced CPU usage, and decreased operational temperature. These outcomes affirm the initial hypothesis that orchestrated containerization enhances the efficiency and availability of resource-constrained systems. Compared to the findings in existing literature, these findings align with the outcomes of Kaiser et al. [6], who demonstrated that orchestrated workloads using containers could improve energy efficiency by dynamically allocating resources. Similarly, Wang et al. [5] highlighted how container orchestration frameworks enable resource-aware scheduling and efficient task management on edge and IoT nodes. Our results support these conclusions, showing that lightweight orchestration like Docker Swarm can reduce unnecessary CPU usage and thermal stress, even with compute-heavy tasks. Despite its contributions, the research does have limitations. The experiments were limited to a single-node Raspberry Pi setup and did not include network traffic or real-world IoT conditions such as environmental variability. Furthermore, while Docker Swarm provided basic orchestration capabilities, it does not offer the advanced features of more sophisticated platforms like Kubernetes, which could yield even better results under different conditions. Future work should explore a broader range of orchestration tools, include networked devices and traffic simulations, and scale testing to multi-node environments to fully assess the implications of orchestration in real-world agricultural or industrial IoT deployments. Nonetheless, this study successfully supports the hypothesis and offers a replicable foundation for further research into power-efficient containerized IoT systems.

APPENDIX A
SUPPORTING MATERIAL

Prior studies on container orchestration and IoT have investigated secure deployment, high availability, edge computing, management, real-time orchestration, and system design [1]–[6].

REFERENCES

[1] B. Pearson and D. Plante, "Secure deployment of containerized iot systems," in *2020 SoutheastCon*, Mar. 2020, pp. 1–8.

[2] M. Šimon, L. Huraj, and N. Búčik, "A comparative analysis of high availability for linux container infrastructures," *Future Internet*, vol. 15, no. 8, p. Art. no. 8, Aug. 2023.

[3] M. Alam and et al., "Orchestration of microservices for iot using docker and edge computing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 118–123, Sep. 2019.

[4] C. Figueroa and et al., "Iot management with container orchestration," in *2023 IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Apr. 2023, pp. 49–54.

[5] Z. Wang and et al., "Container orchestration in edge and fog computing environments for real-time iot applications," arXiv preprint, 2022.

[6] S. Kaiser and et al., "Edge system design using containers and unikernels for iot applications," *IEEE Access*, 2023, university of Texas.