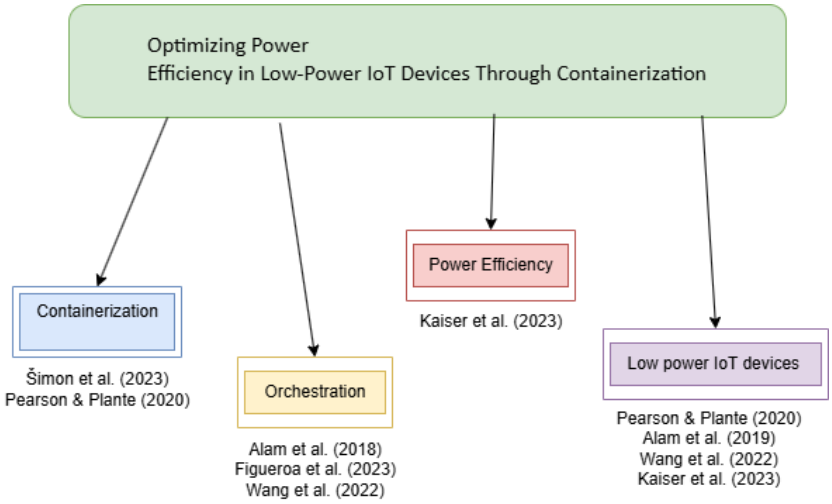
Optimizing Power Efficiency in Low-Power IoT Devices Through Containerization



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% 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% less memory and 40% 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.

Study	Devices	Host Operating System	Cluster	Aim
[1]	Multiple VMs (simulated RPi Zero)	Linux	Docker	Hybrid Orchestration for Real-time IoT
[2]	3 Raspberry Pi 3	Linux	Docker	Secure Containerized Deployment
[3]	3 Raspberry Pi 3	Linux	Docker	Edge-Fog-Cloud Orchestration
[4]	2 Raspberry Pi	Linux	Kubernetes	Smart Home IoT Management
[5]	Simulated RPi Zero (VM)	Linux	K3s	Lightweight Container Orchestration
[6]	Raspberry Pi 4 Cluster	Linux	Docker/Unikernel	Energy-Efficient Hybrid Deployment

Bibliography

[1] B. Pearson and D. Plante, “Secure Deployment of Containerized IoT Systems,” in 2020 SoutheastCon, Mar. 2020, pp. 1–8. doi: 10.1109/SoutheastCon44009.2020.9368276.

[2] M. Šimon, L. Huraj, and N. Búčik, “A Comparative Analysis of High Availability for Linux Container Infrastructures,” Future Internet, vol. 15, no. 8, Art. no. 8, Aug. 2023, doi: 10.3390/fi15080253.

[3] M. Alam et al., “Orchestration of Microservices for IoT Using Docker and Edge Computing,” IEEE Commun. Mag., vol. 56, no. 9, pp. 118–123, Sep. 2019, doi: 10.1109/MCOM.2018.1701233.

[4] C. Figueroa 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. doi: 10.1109/ICEIB57887.2023.10170261.

[5] Z. Wang et al., “Container Orchestration in Edge and Fog Computing Environments for Real-Time IoT Applications,” 2022, arXiv:2203.05161.

[6] S. Kaiser et al., “Edge System Design Using Containers and Unikernels for IoT Applications,” University of Texas, IEEE Access, 2023.