Dynamic IoT Applications and Isomorphic IoT Systems Using WebAssembly: A Case Study

Uddhav P. Gautam, Ram Mude
upgautam@vt.edu, ramm@vt.edu

Abstract

This case study examines the implementation and evaluation of WebAssembly (Wasm)-based IoT systems as presented in [1]. With the proliferation of Internet of Things (IoT) devices, developers need to respond to diverse and rapidly changing user requirements quickly. The research addresses two critical challenges in modern IoT development [2]: the need for rapid and frequent IoT device updates, and the complexity of developing and maintaining multi-layer IoT systems in cloud/edge computing environments.

The study proposes two innovative solutions using WebAssembly. First, dynamic IoT applications enable on-demand partial updates of IoT device behavior without requiring system reboots, where the IoT device consists of a combination of the primary programming language implementing the application and the Wasm runtime. Traditional firmware updates require 10+ seconds for complete system restart, while the proposed Wasm-based approach reduces update time to approximately 1.4 seconds by only restarting the Wasm runtime. Second, isomorphic IoT systems utilize the same Wasm binary built from a common codebase across different layers (device, edge, cloud) of IoT infrastructure, enabling efficient development and maintenance of multi-layer systems.

The implementation demonstrates these concepts through a specific use case of compiling machine learning models for image recognition and classification into Wasm binaries. Researchers compiled widely used image recognition and classification models (ResNet-50 and MobileNetV2) into Wasm binaries using [3] and built an isomorphic architecture where the inference process is executed using the same Wasm binary in each layer of the IoT system. The system achieved successful inference processing with MobileNetV2 running in approximately 0.6 seconds even on the lowest-performing Raspberry Pi 4.

Key technical innovations include the development of Wasmtube [4], a bridging library that enables communication between [5] applications and Wasm modules, and the use of Linux inotify API for automatic detection of Wasm binary updates. A quantitative evaluation was performed to confirm the effectiveness of the proposed method. Although the proposed method introduces an overhead caused by calling Wasm functions from the application, the impact thereof is limited and becomes negligible for complex processing tasks.

The study demonstrates significant practical benefits: 7x faster update times compared to traditional firmware updates, reduced development complexity through code reuse across platforms, and successful deployment of machine learning models across heterogeneous IoT infrastructure. The performance evaluation using ResNet-50 and MobileNetV2 confirmed that the proposed method is practical in the current situation and offers promise for the future.

This case study provides valuable insights into modern IoT system architecture, demonstrating how [6] can address critical challenges in IoT development while maintaining practical performance characteristics suitable for real-world deployment.

References

[1] K. Kuribayashi, Y. Miyake, K. Rikitake, K. Tanaka, and Y. Shinoda, "Dynamic iot applications and isomorphic iot systems using webassembly," in *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*, pp. 1–8, 2023.

[2] S. Bansal and D. Kumar, "Iot ecosystem: A survey on devices, gateways, operating systems, middleware and communication," *International Journal of Wireless Information Networks*, vol. 27, 09 2020.

[3] Apache Software Foundation, "Apache tvm: An end to end machine learning compiler stack." `https://tvm.apache.org/`, 2023. Accessed 2023.

[4] K. Kuribayashi, "Wasmtube: A bridging library which allows you to communicate between elixir and wasm." `https://github.com/kentaro/wasmtube`, 2023. Accessed 2023.

[5] The Nerves Project, "Nerves: Build bulletproof, embedded software in elixir." `https://nerves-project.org/`, 2023. Accessed 2023.

[6] WebAssembly Community Group, "Webassembly specification." `https://webassembly.org/`, 2023. Accessed 2023.