Energy-Consumption Analysis for Cloud-Native Development Approaches

Michele Danieli[1] · Giuseppe De Palma[2]

[1] Imolainformatica, Imola, Italy

[2] University of Bologna, Bologna, Italy

✉ giuseppe.depalma2@unibo.it

Abstract. Energy consumption analysis of cloud computing approaches (microservices, serverless computing, and WebAssembly-based serverless) provides insights for informed decision-making in application design, platform selection, and energy efficiency. We analyse an application use case and compare its energy consumption across the three approaches: containerized service-oriented application, serverless app on OpenWhisk, and on our custom WebAssembly-based serverless platform.

1 Introduction

The growing demand for efficient software solutions has spurred innovative approaches to application development and deployment. Among these approaches, serverless computing has gained significant attention for its scalability and cost reduction potential. While initially dominated by main cloud providers, open-source serverless platforms like Apache OpenWhisk [2] have emerged and now power the serverless offerings of various cloud vendors [7, 11, 9]. As cloud-based solutions become more essential, the energy consumption of software has become a critical issue in today's world. The energy demands and environmental impact of data centers are now central concerns for environmentally conscious individuals and organizations. Consequently, comprehending the energy implications of various architectural choices and platforms is vital for promoting sustainable software development. In the case of serverless platforms, one of the most significant factors impacting the energy consumption is the functions' cold-start time experienced by virtual machine and container-based serverless platforms [13]. The cold-start problem arises when a serverless function is triggered, but the underlying infrastructure must first allocate resources and initialize an execution environment leading to noticeable delays. A novel cloud technology with the potential to contribute in this aspect is WebAssembly [15]. WebAssembly is a binary instruction format designed for a stack-based virtual machine, used by several serverless platforms [8, 10, 14]. WebAssembly functions can be pre-compiled into a binary format, therefore the worker node of a platform can directly execute the function code without the need for an extensive setup processes.

Our work provides an analysis comparing the energy consumption of three different approaches: a containerized service-oriented application, a serverless app on OpenWhisk, and an equivalent serverless app with WebAssembly-functions on our custom serverless platform. We consider the influence of underlying infrastructure and deployed services, aiming to offer valuable insights for developers, architects, and organizations. By evaluating the environmental sustainability of these paradigms, we aim to assist decision-making regarding application design, platform selection, and energy-efficient software development.

2 Energy Consumption Comparative Analysis

In recent years, the rise of serverless systems and application architectures has drawn attention to their environmental impact. In recent years, researchers have emphasized the importance of developing a model to measure the energy footprint of serverless computing and evaluate resource consumption at various abstraction layers [13]. There has also been an increasing focus on analyzing power consumption within serverless workloads, particularly in the context of machine learning, which has gained significant popularity nowadays [12]. These analysis can help to better comprehend how non-functional aspects such as sustainability and costs are relevant in architectural decisions, therefore we aim to provide a comparative analysis of energy consumption in application use cases across different cloud computing approaches, including containers and WebAssembly-based solutions. We established a controlled environment using Kubernetes, a widely-used container orchestration platform [4]. Within this environment, we employed Kepler [3] and Prometheus [6] for metrics gathering, along with Power API [5] to measure energy consumption on the nodes. In our analysis, we gather real-time CPU power consumption data through the Intel CPU HWPC Sensor, while with Kepler we have access to container-level data such as total time for CPU usage and total packets sent and received for network usage.

Figure 1 depicts the application use-case. The application is composed of several sensors that send data, which are processed by a parser service/function to normalize it and then sent to an aggregator service/function. The processed data is then stored and used by a dashboard. The scenario depicts a typical IoT application, e.g., a lab or a smart home, with possibly a division between the edge (with the parser and raw store) and the cloud (with the aggregator service and the rest of the application). We deployed the containerized application on a Kubernetes installation and conducted a series of tests using JMeter [1] to simulate varying scenarios. We considered several intensity and traffic profiles to compare the behaviors of the different approaches on a 3 nodes cluster setup.

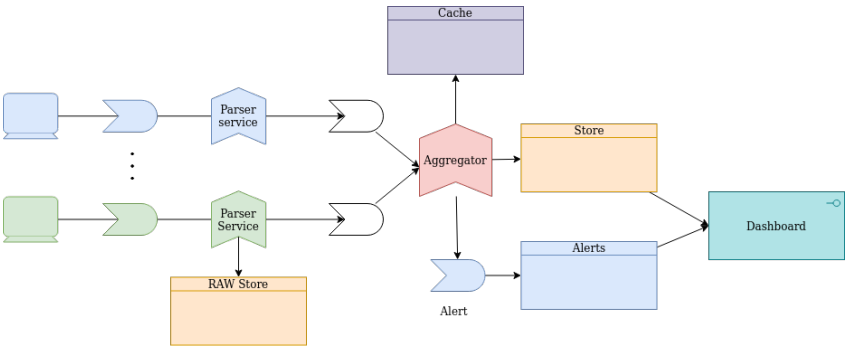

Figure 1: The application scenario schema used in the experiments.

3 Our Presentation

In our presentation at Microservices 2023, we will provide an overview of the Apache OpenWhisk and our custom WebAssembly-based platform. We will discuss energy-consumption implications of using the serverless approach and how the cold-start problem's impact can affect sustainability, together with how WebAssembly can help. Finally, we will exhibit a use case to show the contrast its energy usage between a traditional implementation (with containers), a traditional serverless app and a WebAssembly-based serverless app.

References

[1] Apache jmeter. https://jmeter.apache.org/.

[2] Apache openwhisk. https://openwhisk.apache.org/.

[3] Kepler. https://sustainable-computing.io/.

[4] Kubernetes. https://kubernetes.io/.

[5] Powerapi. https://powerapi.org/.

[6] Prometheus. https://prometheus.io/.

[7] IBM Cloud. Ibm cloud functions. https://cloud.ibm.com/functions/.

[8] Cloudflare. Cloudflare workers. https://workers.cloudflare.com/.

[9] Adobe Cloud Native Code. Adobe cloud native code. https://developer.adobe.com/runtime/.

[10] Fastly. Fastly compute@edge. https://www.fastly.com/products/edge-compute.

[11] Digital Ocean. Digital ocean functions. https://www.digitalocean.com/products/functions.

[12] Panos Patros, Josef Spillner, Alessandro V Papadopoulos, Blesson Varghese, Omer Rana, and Schahram Dustdar. Toward sustainable serverless computing. *IEEE Internet Computing*, 25(6):42–50, 2021.

[13] Alexander Poth, Niklas Schubert, and Andreas Riel. Sustainability efficiency challenges of modern it architectures–a quality model for serverless energy footprint. In *Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, Proceedings 27*, pages 289–301. Springer, 2020.

[14] WasmEdge. Wasmedge. https://wasmedge.org/.

[15] WebAssembly. Webassembly. https://webassembly.org/.