# Comparison of client-server and serverless architectures in cloud computing

Anastasija Cvetanovic
*Cloud Computing*
*Ss. Cyril and Methodius University - Skopje Faculty of Computer Science and Engineering*
Professor: Marjan Gushev
Lecturer: Dimitar Mileski
Date: 13.02.2024

***Abstract*—This paper examines the performance differences between Client-Server and Serverless architectures in the context of Cloud Computing. Using a web application that determines the smallest value from a CSV file, the processing times of these two architectures under identical load conditions are analyzed and compared, with the results being displayed in the context of decreasing traffic.**

***Index Terms*—Cloud Computing, Serverless, Client-Server, Performance analysis, Google Cloud**

## I. Introduction

The rapid advancement of cloud technologies has significantly influenced the development and deployment of web applications. This study explores the comparative performance and operational dynamics of Client-Server and Serverless architectures within cloud computing. Focusing on a web application designed to identify the smallest value in a CSV file, this research aims to elucidate the practical implications of adopting either architecture in terms of efficiency, scalability, and cost-effectiveness. Through empirical analysis under identical load conditions, this paper contributes to a deeper understanding of the architectural choices available to developers in the cloud computing paradigm.

## II. Methodology

This study's methodology involves a comparative analysis of Client-Server and Serverless architectures within a cloud computing environment. The primary objective was to measure the performance impact of each architecture in processing CSV data to find the smallest value. This was achieved by setting a variable 'Number of Requests' as the initial client load and observing the subsequent processing times, with the unique consideration of decreasing traffic conditions.

## III. Implementation

### A. Client-Server Communication

The implementation began with a user interface utilizing HTML, CSS, and JavaScript.The frontend was deployed in the Google Cloud zone europe-west8-a to leverage geographical proximity and resource availability. The backend service, utilizing Java and Spring Boot on port 8080, was deployed on a VM in the europe-west3-c zone to ensure consistency in performance metrics. Local testing via Postman was successful before deploying to Google Cloud VMs. The application was made accessible through http://34.154.233.208/, with HTTP and HTTPS traffic enabled for internet connectivity.

### B. Serverless Architecture

The Serverless backend was realized by packaging the service into a Docker image, subsequently hosted in Google Cloud's Container Registry, and then actively deployed through Google Cloud Run. This setup was mirrored in the frontend through an interactive feature, enabling users to seamlessly switch between Server and Serverless backends. This functionality was not only critical for user experience but also central to the empirical evaluation of each architecture's performance.

To ensure uniform test conditions, the Serverless deployment was strategically placed in the europe-west3-c zone, identical to the Client-Server backend's location. Such parity in deployment zones was crucial to guarantee a balanced and accurate assessment of processing times and resource efficiency across both architectures.

## IV. Analysis and Comparison of Testing Procedures

### A. Serverless Architecture Analysis

The empirical data from the Serverless architecture indicated a pronounced variability in processing times. This fluctuation is postulated to originate from the dynamic management of computing instances and the occurrence of cold starts—a phenomenon where dormant functions need to be instantiated, adding latency to the initial response. Such observations underscore the trade-off inherent in Serverless computing: the advantage of elastic scalability may come at the cost of predictability in performance.

### B. Client-Server Architecture Analysis

Conversely, the Client-Server model, characterized by a virtual machine (VM) setup, exhibited a more consistent processing time across tests. This consistent behavior aligns with the architecture's continuous operation, negating the cold start penalty and thus offering a more predictable performance profile.

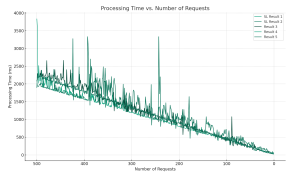


Fig. 1: Enter Caption

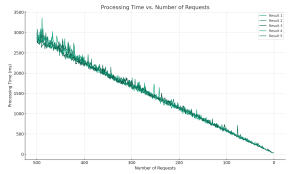


Fig. 2: Enter Caption

### C. Comparative Scalability and Cost-Efficiency

When juxtaposing scalability and cost-efficiency, the Serverless model displayed a favorable adaptation to fluctuating loads, potentially translating to economic efficiency during periods of low utilization. However, the Client-Server architecture, while potentially less resource-efficient during downturns, could offer a steadier cost structure and performance consistency.

### D. Synthesis of Observations

These contrasting patterns highlight the quintessential considerations for architectural choice in cloud computing: the balance between performance predictability and operational flexibility. The Serverless model is poised to offer agility in resource allocation, whereas the Client-Server paradigm may be preferable for applications demanding uniform response times.

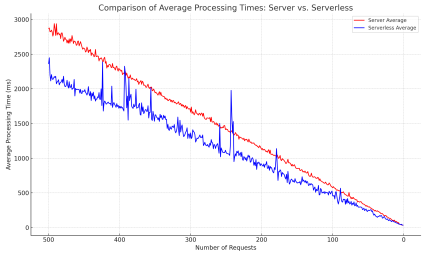


Fig. 3: Comparison of Average Processing Times: Server vs. Serverless

### E. Interpretation of Processing Time Data

The diagram in Figure 3 depicts the comparison of average processing times between Server and Serverless architectures. A notable observation is the blue line, representing the Serverless architecture, which exhibits significant variability in processing times, particularly with pronounced spikes. This variability can be attributed to the instance initialization phenomenon known as cold starts. Conversely, the red line, indicative of the Server (VM) architecture, shows a more gradual decline in processing time, implying a stable and warm environment with readily available resources. The descending trend observed in both lines correlates with the decreasing number of requests, suggesting that as the load diminishes, the systems become more responsive, reducing the average processing time.

## V. Conclusions

The choice between architectures should consider the specific application requirements, with Serverless providing flexibility and Server models offering consistent performance. The study's findings are supported by visual data representations, with figures illustrating these comparisons.