Summary

This paper discusses the benefits and the challenges of serverless computing, especially, cold start problem. The paper emphasizes on the problem and provide a novel machine learning two-layer adaptive approach to tackle this issue [1]. The experiment showed improvements in memory consumption along with the execution invocations compared to the Openwhisk platform [1].

Ever increasing need to store data volumes, maintenance, resources has led to the idea of cloud computing. Cloud computing has three models of services: IaaS, PaaS, SaaS. In 2014, Amazon coined the term of serverless computing, which was successful in dealing with all the problems, however it faces challenges in system, performance, and software engineering [1]. One such performance challenge is cold start delay, which is a delay in preparation of the function execution environment. Mundane methods to deal with this delay are not suitable for dynamic cloud environment. Therefore, to mitigate the issue, the authors determines the policy to keep the containers warm [1]. Among the three models of cloud, serverless falls between PaaS and SaaS. Serverless computing breaks the down an application into small, independent, stateless, short lived, task specific units called functions which are deployed on containers and a new service model called Function as a Service (FaaS). Container with all its dependencies gets created on function trigger [1]. However, initializing new container and allocating memory leads to a delay called cold start which leads to certain problems like if there were more invocations then available container, multiple instances would need to be created which led to further delay. That's why serverless functions has cold start delay [1]. Two approaches to reduce this are: reducing delay duration and minimizing cold start frequency occurrence.

Most of the solutions by other related works to reduce cold start delay duration were based on reducing container preparation or to load function code and other dependencies into memory. For reducing the cold start occurrence approach, the general solution is to prepare function containers or to reuse the containers [1]. Further analysis has been done on related work with the pivot on resource consumption and adaptability in reducing cold start delays. Some of the methods like container reuse, warm container pools, and periodic function calling can lead to resource wastage. Also, for the adaptability reinforcement learning algorithms are proven to be suitable for automated decision making [1]. When it comes to reducing cold start delays, using deep reinforcement learning is the best solution. The efficiency of keeping containers warm depends on invocation patterns for reusing containers. A common method used by popular platforms to reduce cold start delays is to keep containers warm for a fixed period after function execution, known as the idle-container window [1]. However, only Openwhisk has provided solutions among this platform. This article uses Openwhisk as a baseline platform for their research because it uses the machine learning algorithms to predict future invocation times, therefore a two-layer approach is proposed to make platform adaptable. The first layer aims to determine the best time for the idle-container window to reduce the number of cold starts delays and resource consumption using the reinforcement learning algorithm [1]. Longer the window, shorter the cold start delays. The purpose of second layer is to reduce cold start delay time. Deep reinforcement learning algorithm is used to execute the first layer. The algorithm determines the state of the environment, an appropriate action to determine the value of idle container window, acting on the environment by updating the serverless computing platform with a new value for the idle container window parameter, along with the calculation of reward function at the end of a time period [1]. The size of the idle-container window affects both the number of delays and resource consumption. A reward function is used to balance these factors, considering the ratio of cold start delays to total invocations and the penalty for memory wastage. The algorithm aims to find a balance between minimizing delays and reducing memory waste by adjusting the window size. Cold start delays are inevitable due to infrequent invocations and memory wastage. The second layer of the

approach aims to adapt the number of containers based on predicted concurrent invocations using LSTM for time-series analysis [1].

The experimental set up involves two types of data sets. One includes sequential invocations and other includes concurrent invocations. Usage of Openwhisk platform has been done for the computation. Machine learning algorithms have been used for the training set up. The authors evaluated their approach through simulations [1]. They used I/O bound functions that sends an HTTP request to a Web page with a response time of 6 seconds and simulated intervals for asynchronous requests. Two simulations are performed: one with default parameters of Openwhisk and one with new parameters from the proposed approach [1]. In the first simulation, the proposed approach showed improvements in reducing cold start occurrences and memory consumption compared to the Openwhisk platform. Memory consumption showed improvements of 11.11% to 12.73% depending on the average invocation rate. In the second simulation, the proposed approach outclassed the Openwhisk platform by correctly predicting and executing 56 invocations on the prewarmed containers compared to the 32 invocations by the former. Thus, an improvement of 22.65% in the execution of invocations on prewarmed containers [1].

To conclude, the two-layer approach of the authors was significant in reducing the memory consumption and cold start delay by the significant margin compared to the Openwhisk platform.

References

[1] P. Vahidinia, B. Farahani, and F. S. Aliee, "Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2022.3165127, 2022. [Online]. Available: https://ieeexplore-ieee-org.ezproxy.library.dal.ca/document/9749611. [Accessed: 7 June 2023].