A Performance and Cost Analysis of Java-based Function-as-a-Service on AWS

Alan Kavanagh
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00080986@myTUDublin.ie

Introduction

This study consists of a research project that compares and contrasts the performance and cost of Java-based Function-as-a-Service (FaaS) solutions on Amazon Web Services (AWS). The objective of the research is to understand the improvement that each solution offers with regard to cold start mitigation for Java-based serverless functions on AWS. A series of experiments are conducted to assess the start-up performance and associated cost of lambda functions deployed to AWS Lambda, AWS Lambda with SnapStart, and AWS Lambda with Provisioned Concurrency. The results of each experiment are collected, depicted, and analysed, and then the findings and conclusions are presented in the form of a performance and cost comparison.

Hypothesis

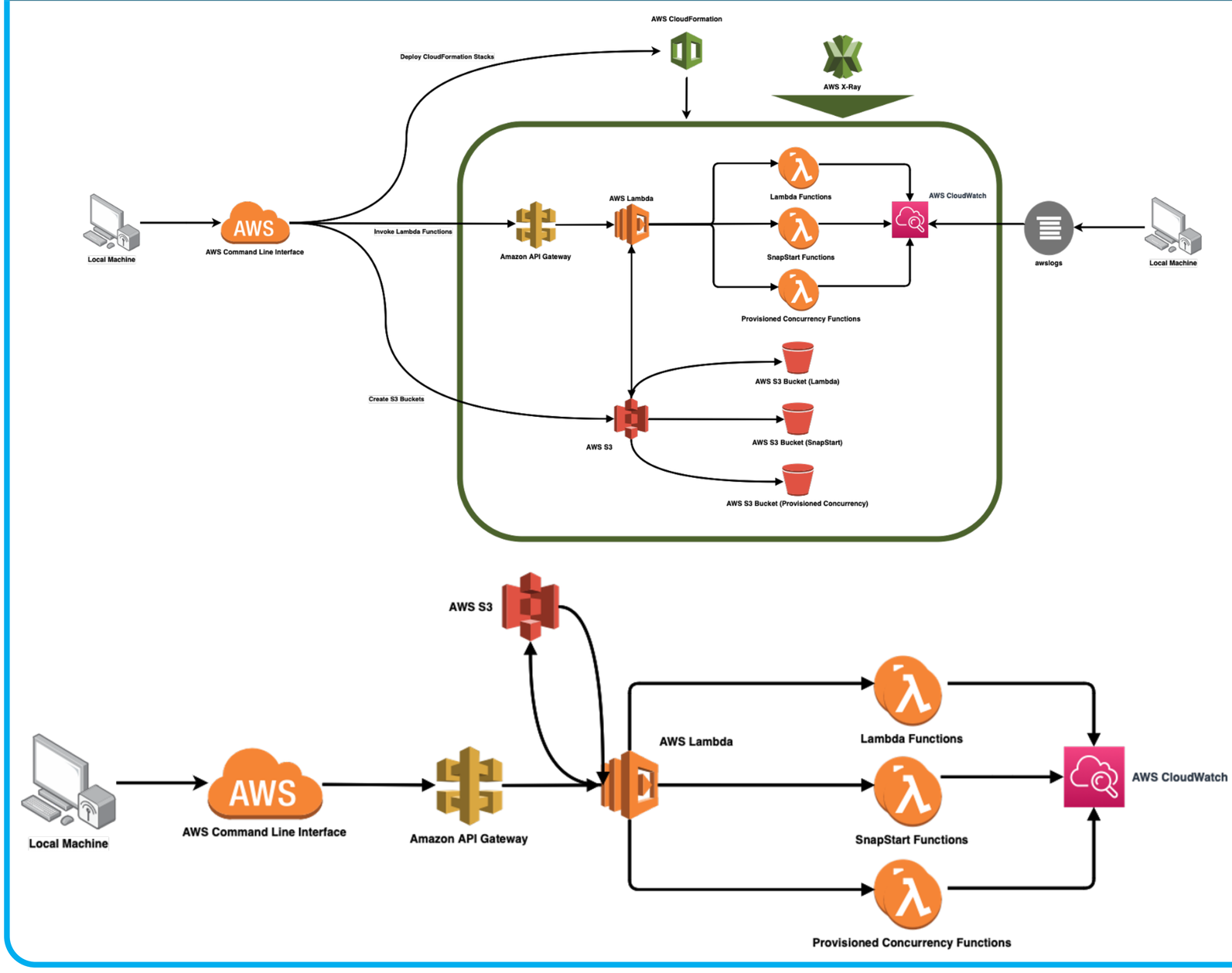
"Lambda SnapStart for Java can improve startup performance for latency-sensitive applications by up to 10x at no extra cost" - AWS

It is expected that the results produced will help identify an increased start-up performance for SnapStart in comparison to Lambda and Provisioned Concurrency. If the hypothesis is correct, the results should indicate a reduced cold-start latency, and reduced number of cold-start occurrences, without incurring any additional costs, when deploying Java-based serverless functions on AWS Lambda with SnapStart enabled

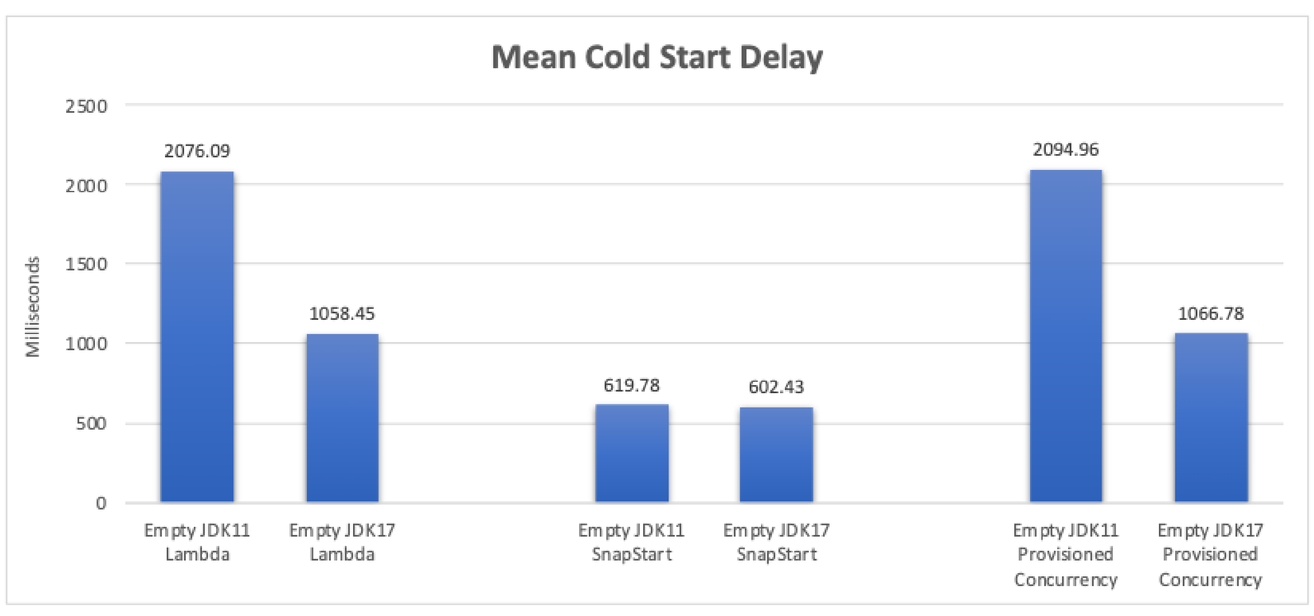
Research Questions

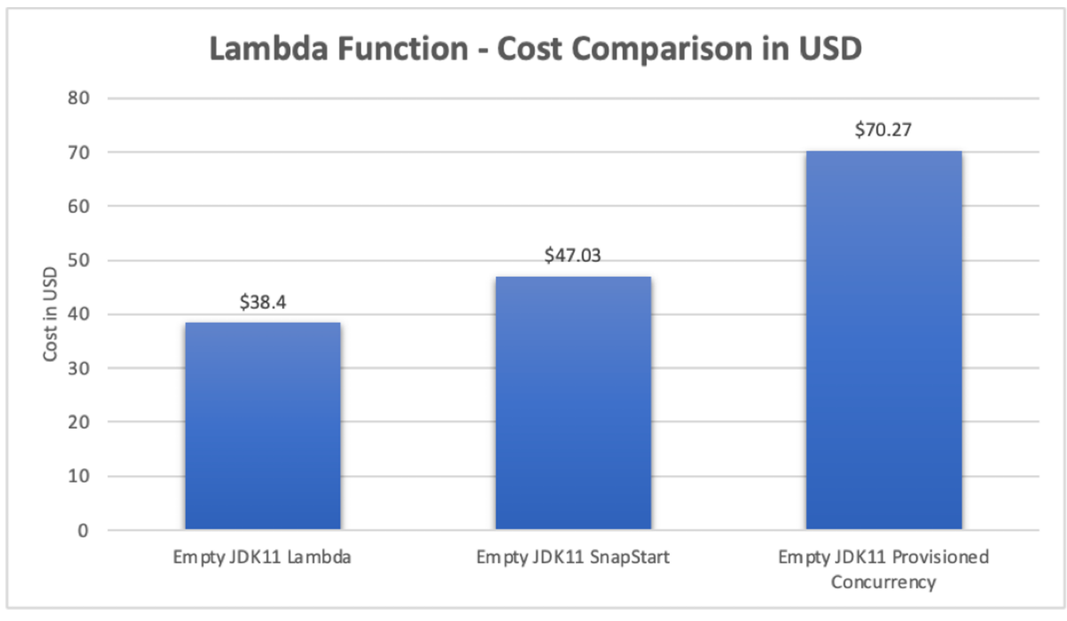
- RQ1. What impact does AWS Lambda with SnapStart have on the cold start latency, and cold start occurrences, of Java-based serverless functions?
- RQ2. How does AWS Lambda with SnapStart perform in comparison to AWS Lambda, and AWS Lambda with Provisioned Concurrency?
- RQ3. Does AWS Lambda SnapStart increase start-up performance by 10x at no extra cost?

Deployment Architecture and Invocation Flow



Results and Findings





Conclusions

SnapStart does not decrease the number of cold start occurrences in comparison to AWS Lambda. SnapStart consistently has a 600ms restoration time, and outperforms AWS Lambda for functions with a higher initialisation time. SnapStart incurs the same costs as AWS Lambda for higher execution time functions, or slightly more for lower execution time functions. Provisioned Concurrency outperformed SnapStart in all scenarios, however, it incurred additional costs up to x5 or x6 in comparison. Upgrading from JDK11 to JDK17 can decrease the average execution time by 75%.

QR Code for Recording