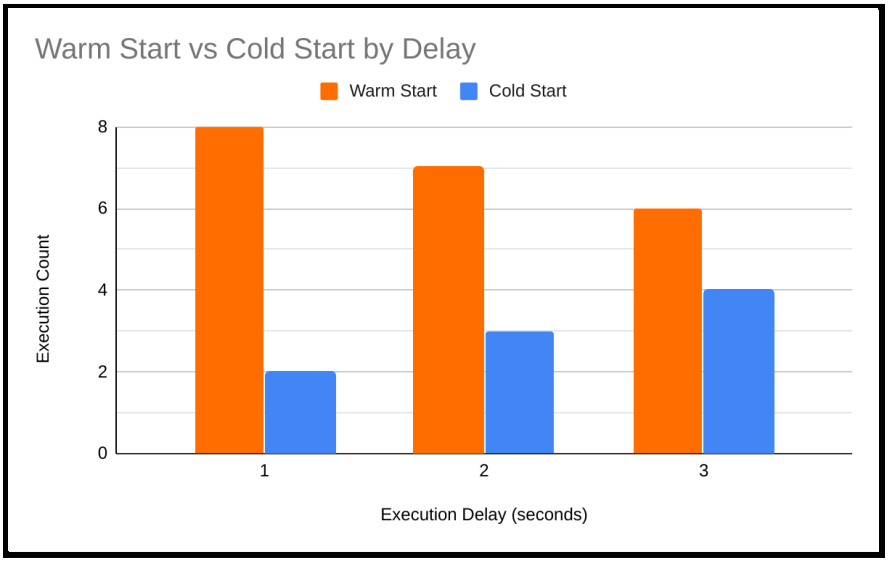
Thien Tran
Dr. Phani Vadrevu
Cloud Computing
December 1st, 2021

Analysis of the Relationship between AWS Lambda Execution Delay and Container Reuse

Amazon Web Service (AWS) Lambda functions are serverless compute models that execute their developer-provided code when they are invoked by some trigger. The way that they work is that AWS provisions a container - a virtualized, isolated computing environment - that executes the lambda event handler code in response to a trigger.[1] AWS may provision a new container to execute the code, otherwise known as a cold start, which would take additional time compared to reusing a pre-existing container, otherwise known as a warm start, since the environment and such must be initialized in a new container.[1] I, for Task-3 in Lab 2 of CSCI-4452: Cloud Computing, briefly experimented with the effects of function execution delay on frequency of cold and warm starts and found that the increase in execution delay led to a higher frequency of cold starts.

What I did to gather the data and determine the relationships are as follows. FIrst, I reused most of the Lambda code from Lab 2 which - most importantly - allowed for the execution of either a cold start function or warm start function depending on the context of the container used to execute the Lambda function. Then I added a modification that sleeps the function for a predetermined, user-provided amount of time prior to the return call that ends the function execution. I then made three functions of this type, each with an execution delay of 1, 2, and 3 respectively. A bash script was then used to sequentially invoke each function 10 times on the freshly deployed Lambda function with a 1 second delay between each batch to imitate asynchronous invocations. The responses were saved to a text file, cleaned, and then aggregated. The results are shown below.



As you can see in the graph above, it shows that each of the 3 functions were able to successfully execute their code for a total of 10 times, each for one increment of the batch invocations. Interestingly, it seems that the frequency of cold starts increases proportionally to the Lambda functions' execution delay. This is likely due to Lambda's autoscaling concurrency policy forcing more new containers to be made on functions with higher execution delay due to them having less or no available container to meet and handle growing workloads since the existing ones have to wait for a longer period of time compared to Lambda functions with lower execution delay. In short, lower execution delay brings the cold start amount closer to 1, while higher execution delay brings the cold start amount closer to N where N is the amount of executions.

Although the time-delay between function invocation requests were set at 1 to imitate realistic asynchronous invocations and that the task was more about modifying the Lambda's execution delay instead of the invocation delay, it's worth mentioning that I also experimented with timing between function invocations. The results were that for 10 invocations, no delay led to Lambda ALWAYS cold starting, and a 5+ second delay ALWAYS led to a warm start after the first initial cold start for each function.

Shown below are the data of the Lambda invocations of this experiment separated by their which includes their status, IP Address, Execution time, and cold/warm start context. Each table is separated by its execution delay.

Table 1: Execution Delay of 1 Second

Execution Delay	Status	IP Address	Execution Time	Context
1	200	169.254.138.85	0.0148332119	Cold Started
1	200	169.254.128.53	0.01981973648	Cold Started
1	200	169.254.138.85	6.94E-05	Warm Started
1	200	169.254.138.85	7.20E-05	Warm Started
1	200	169.254.128.53	7.30E-05	Warm Started
1	200	169.254.138.85	7.46E-05	Warm Started
1	200	169.254.128.53	6.96E-05	Warm Started
1	200	169.254.128.53	7.58E-05	Warm Started
1	200	169.254.138.85	7.08E-05	Warm Started
1	200	169.254.138.85	7.25E-05	Warm Started

Table 2: Execution Delay of 2 Secondss

Execution Delay	Status	IP Address	Execution Time	Context
2	200	169.254.196.165	0.01916623116	Cold Started
2	200	169.254.41.85	0.01011347771	Cold Started
2	200	169.254.122.125	0.005056858063	Cold Started
2	200	169.254.196.165	7.25E-05	Warm Started
2	200	169.254.41.85	6.96E-05	Warm Started
2	200	169.254.122.125	7.39E-05	Warm Started
2	200	169.254.41.85	6.79E-05	Warm Started
2	200	169.254.122.125	7.53E-05	Warm Started
2	200	169.254.41.85	7.06E-05	Warm Started
2	200	169.254.196.165	8.06E-05	Warm Started

Table 3: Execution Delay of 3 Seconds

Execution Delay	Status	IP Address	Execution Time	Context
3	200	169.254.56.45	0.01637125015	Cold Started
3	200	169.254.46.113	0.009649038315	Cold Started
3	200	169.254.207.61	0.006757736206	Cold Started
3	200	169.254.217.57	0.005466938019	Cold Started
3	200	169.254.56.45	7.18E-05	Warm Started
3	200	169.254.46.113	7.77E-05	Warm Started
3	200	169.254.207.61	0.0002248287201	Warm Started
3	200	169.254.56.45	7.06E-05	Warm Started
3	200	169.254.46.113	7.44E-05	Warm Started
3	200	169.254.207.61	0.0002529621124	Warm Started

Sources

[1] https://aws.amazon.com/blogs/compute/container-reuse-in-lambda