FaaS as an alternative to Microservices: An Investigation

IIT Bombay

Status Report as on Feb 8th, 2019

Work Done so Far.

Understanding the Internals of AWS

- Getting the container details that hosts the lambda function programmatically
- Can identify cold vs warm containers using the above strategy, how containers are reused for requests
- Information regarding what resources are allocated to lambda
- Actions that can be done from within the container- create files, fetch permission details, etc.
- Tunable parameters of lambda functions: amount of memory allocated, number of concurrent requests, execution duration, region where lambda is instantiated
- Some experiments performed: invoke more than 1000 concurrent requests to lambda, exactly 1000 served, rest were throttled
- Single threaded lambda functions vs multi-threaded lambda functions
- Amount of memory allocated to lambda function is proportional to the amount of CPU allocated to lambda
- Benchmarking AWS lambda function by running CPU intensive code

CPU intensive benchmarking results(Single threaded)

Memory Allocated(MB)	Memory Used(MB)	Duration(ms)	Billed Duration(ms)
128	21	9927.12	10000
256	21	4928.03	5000
384	21	3250.65	3300

512	21	2414.97	2500
1024	21	1193.16	1200
2048	21	660.35	700
3008	21	655.71	700

CPU intensive benchmarking results(Multi threaded)
No of threads: 4

Memory Allocated(MB)	Memory Used(MB)	Duration(ms)	Billed Duration(ms)
128	21	10477.71	10500
256	21	5102.22	5200
384	21	3291.65	3300
384	21	3447.51	3500
512	21	2448.80	2500
3008	24	661.38	700

Building a FaaS based Application

- Invoking a lambda function programmatically using Amazon API gateway.
- Invoking a lambda function through another lambda function

- Chaining of lambda functions (synchronous as well as asynchronous)

Comparison between monolithic and serverless architecture

a) A Real-time File Processing application that demonstrates a Markdown conversion application where Lambda is used to convert Markdown files to HTML and plain text is built using AWS lambda and AWS S3 buckets.
b) Similar application is also deployed as a Web server on an EC2 instance.
c) Initial measurements like response time were taken.
d) Serverless application took 3.75s on an average.
e) Monolithic Node JS web application took 1.04s on an average.

OpenLambda

a) Looked into OpenLambda for both their docker based and process based infrastructure.
b) Several deficiencies were found including the inability to launch multiple parallel worker processes. Thus severely limiting scalability.

Knative

a) Researched on Knative - an open source Serverless platform using Kubernetes cluster management.
b) Deployed Knative on test systems to test usability.
c) Tested sample applications on Knative.

Ongoing Work

Literature Review

Several papers are collected and are being reviewed to get a brief idea about the research that is being undertaken in the area of Serverless Computing. The papers are broadly classified as:

a) Application-centric: In this category, a small application is built just using FaaS. This explores the feasibility of building applications with FaaS.
b) FaaS-ification of Applications: Tools/Processes/Design Patterns for the conversion of any application into a FaaS based applications.
c) Cost evaluation of Lambda/Microservice based applications: Analyzing/Budgeting cost of deploying applications with the help of tools and models.
d) Comparison of Serverless Platforms: Qualitative and Quantitative comparison of various offerings of serverless computing such as AWS, Google Cloud Functions, Microsoft Azure, etc.
e) Optimizing Serverless: Papers that deal with performance optimization techniques for optimizing the current serverless offerings.
f) Extending Serverless: New Serverless platforms that claim better performance, Storage for Serverless, etc.
g) Benchmarking FaaS: Papers that deal with benchmarking lambda functions in one of the aspects such as CPU/Memory/Disk/Network.

h) Comparison between monolithic and serverless architectures: Papers that quantitatively compare an application that is deployed both as a Monolithic application as well as a FaaS based application.

Future Work

Deploying a financial business use case using FaaS

a) Identification of NPA use case to be deployed.
b) Use of Machine learning models.
c) State to be maintained using external DB.

Dispatch overhead of AWS over Knative

a) Set up an environment using Knative similar to AWS in order to find the extra overhead imposed by the platforms.
b) Container provisioning details of AWS to be identified and similar set up to be used in Knative.

Benchmarking Knative, AWS, and other serverless offerings

a) CPU, Memory, Disk, Network intensive benchmarks to be run.
b) Vary various parameters and collect the necessary metrics