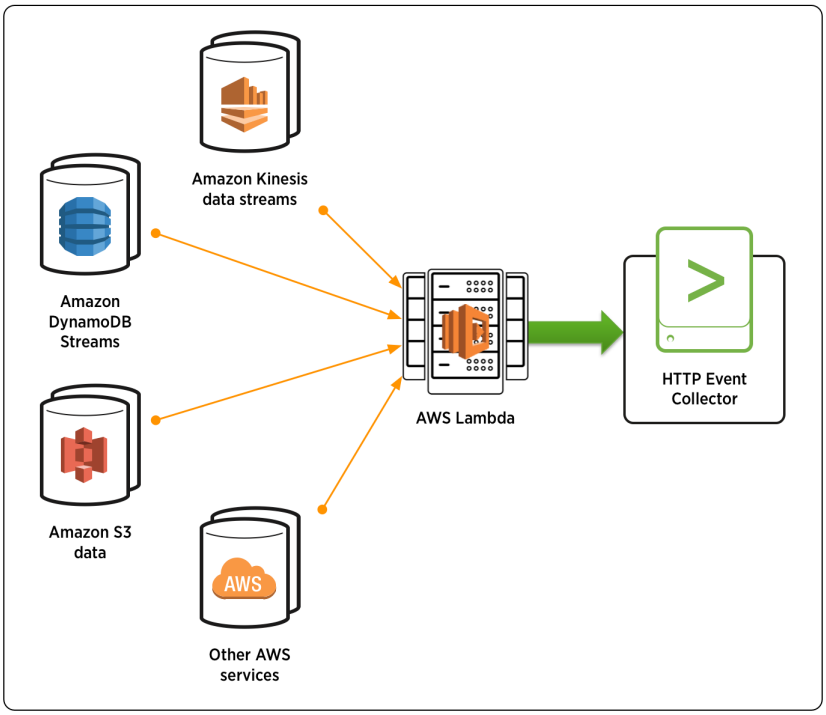
AWS Lambda

TJ CCC

February 2020

1 Introduction

AWS Lambda is a service that allows you to create specific functions that run automatically in response to an event without provisioning or managing servers. These events can be anything ranging from adding or modifying an object in an S3 bucket, updating rows in a DynamoDB table, or receiving an HTTP request. Similar to how formulas in spreadsheets automatically modify the contents of a cell, Lambda functions automatically trigger to do whatever action specified.



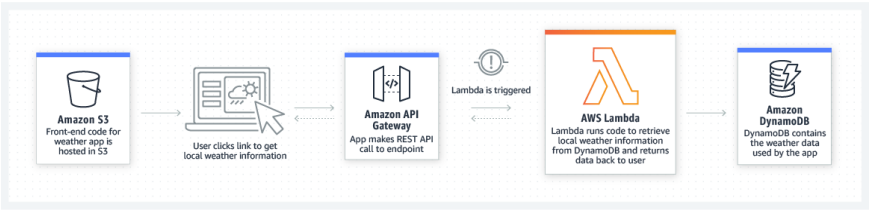
2 Usage

As mentioned previously, there are a wide variety of services that work well with AWS Lambda. Several examples of ways you can use Lambda are shown below:

- **S3 Buckets** – You can use Lambda to process files after they are uploaded or modified in an S3 bucket. Possible functions include thumbnailing images, resizing images, indexing files, and converting file formats.

- **DynamoDB** – AWS Lambda is especially useful for monitoring and manging DynamoDB tables. It can be used to validate, sort, or filter data added to the table, or fetch data from the table to use in your application.

- **Web Applications** – When a user lands on your web page or clicks on a certain element, Lambda can be used to automatically trigger a response, whether it be retrieving data from a DynamoDB table or another website.



3 Benefits

- **Automated** – This is one of the key features of AWS Lambda. Lambda automatically manages underlying resources for you with a fault-tolerant

infrastructure, allowing for you to focus more on other aspects of your products.

- **Serverless** – Another key feature is that Lambda functions allow you to run your code without having to worry about managing servers. When creating a large application, managing the infrastructure to host and provision backend code would generally require you to size and scale multiple servers. Lambda manages the backend itself by creating automatic responses to events, eliminating the need for servers and allowing you to focus more on developing the frontend code.
- **Scalable** – Lambda are highly scalable, being able to either be completely inactive (and you would not be charged for them while they are not used) or handle tens of thousands of requests. There is no limit to how many requests the code can handle. Lambda typically runs its functions within milliseconds of an event, and since it is so highly scalable, it can consistently do this even with a drastic increase to the number of requests.
- **Consistent** – Lambda functions execute with almost flawless consistency. If you were to use a server instead, then any small breakdown in the server could result in devastating effects. Therefore, Lambda eliminates the need to worry about the background performance of your application.
- **Cheap** – There is no up-front cost to creating an AWS Lambda function. You are only charged for every the number of requests it receives and the amount of time the code is run, measured in increments of 100ms. This adds up to be a very low cost and helps both large and small businesses greatly reduce costs.

4 Languages Supported

AWS Lambda supports several programming languages, as shown below:

- Node.js 12.13.0, 10.16.3, and 8.10
- Java 11 and 8
- Python 3.8, 3.7, 3.6, and 2.7
- .NET Core 2.1
- Go 1.x
- Ruby 2.5
- Rust

5 Other Cloud Service Platforms

Other cloud computing providers provide their own version of AWS Lambda. Each service has their own respective benefits and drawbacks.

- **Google Cloud Functions** – Only three languages are supported through this service: Node.js, Python, and Go. However, it is easier to deploy Google Cloud Functions and AWS Lambda functions.
- **Azure Functions** – Although it offers a bit less overall functionality than AWS Lambda, Functions does offer powerful integrations and practical functionality. Azure also includes integrations with VS Team Services, GitHub, and Bitbucket.

6 Performance of Serverless Platforms

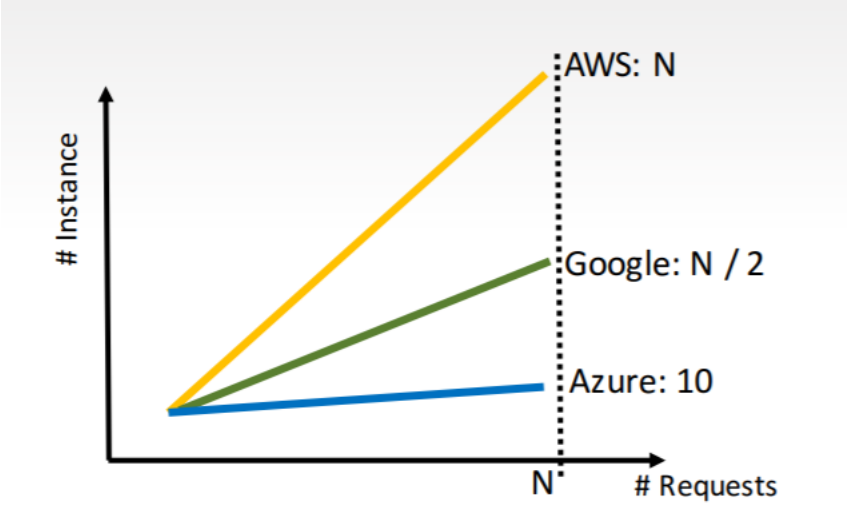


Figure 1: Number of concurrent instances running after sending N concurrent requests. Reprinted from 'Peeking Behind the Curtains of Serverless Platforms' [1]

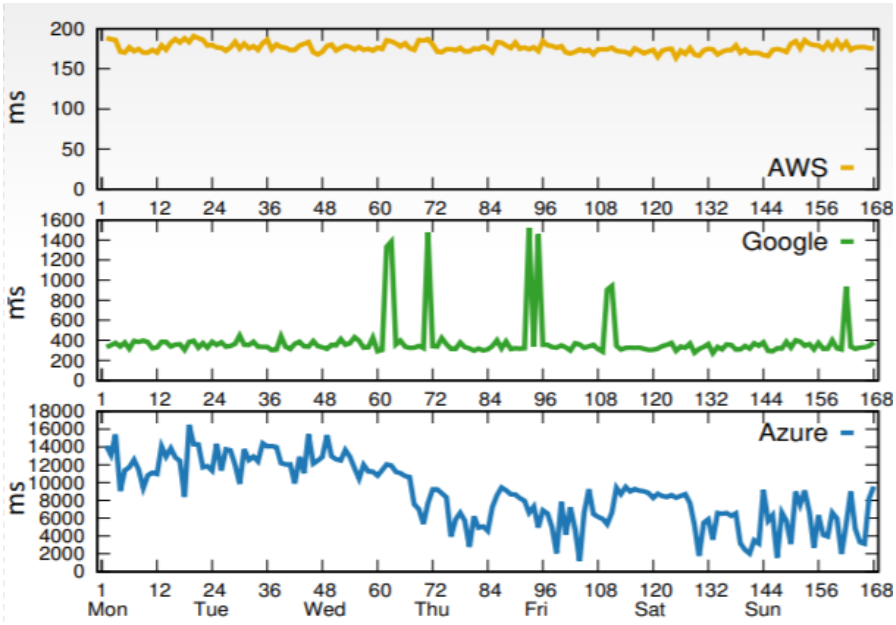


Figure 2: Median coldstart latency over seven days. Reprinted from 'Peeking Behind the Curtains of Serverless Platforms' [1]

References

[1] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 133–146, Boston, MA, July 2018. USENIX Association.