# A Comparison of Data Transport Options for AWS Lambda for Relational Joins

Mina Dalirrooyfard, Farnaz Jahanbakhsh, Matthew Perron
{mperron, minad, farnazj}@mit.edu

October 9, 2018

## Problem Statement

The purpose of this project is to determine the feasibility of using Lambda for a distributed database application given Lambda's communication constraints. Serverless compute infrastructures like AWS Lambda have the potential to support a wide range of applications. Yet AWS disables opening ports from within Lambda functions, making it impossible to send data directly from one Lambda to another. This means that users must use other methods to pass data between function instances. There are several services that might fit this purpose including; AWS Simple Queuing Service (SQS), Kinesis, EC2 instances for routing traffic, and Amazon Simple Storage Service(S3). Each of these approaches had different performance and cost properties. For instance, S3 is highly elastic and charges by the data usage and request, while setting up EC2 instances has fixed cost (per instance), but might have higher throughput and lower latency. We will investigate the cost and performance trade-offs of each of these services to support a single application, hash-partitioned joins in a distributed database management system(DBMS) executing on Lambda workers.

## Research Methodology

The amount and frequency of data communication between Lambda instances are application dependent. Applications like video encoding have embarrassingly parallel components that are shown to map well to Lambda [2]. Instead we are interested in applications requiring more communication. We focus on hash-partitioned joins (Grace Hash Join [3]) as an application that is commonplace and could potentially benefit from execution on Lambda. Hash-partitioned joins on distributed systems have a partitioning phase, where each table is hash partitioned on its join key, and a join phase where workers collect partitions from each table and execute the join on each partition. In common parlance, a partitioned hash join requires a data shuffle.

We will implement a simple join query, query 12 from TPC-H [1], a well known decision support benchmark, with scale factor 1000 (1TB of raw data). This query requires a join on the lineitem and order tables, the two largest tables in the database.

Due to the constraints that Amazon enforces on opening ports on Lambda, data transfer between instances needs to happen through a communication intermediary. We are interested in exploring how different approaches to data communication between Lambda instances impact the end-to-end latency of our specific application as well the latency-cost trade-off.

One data transfer approach that we may examine is Amazon Simple Queuing Service (SQS) which is a fully managed message queuing service provided by amazon. It allows you to create and destroy queues rapidly, has a maximum message size of 256KB, and charges by the number of messages. Clients push and pull from the queue via http and must delete messages explicitly once they are read.

Another approach we could explore is establishing a virtual machine to act as a router between Lambda instances. By opening a port on a virtual machine, Lambda instances would be able to connect to the machine and use it to send and receive data.

We plan to implement these approaches and tune them to achieve the best latency and cost we can obtain in each scenario for our specific application. We then will compare and contrast these performance metrics.

## Plan and Schedule

We have an 8 week period to do the project and we plan to start with characterizing each of the data transport options and give a comparison of each in terms of the services they provide, with respect to performance, pricing model, elasticity, and fault tolerance (If we need to re-execute the join on a failure of a lambda). Next we will implement the workload that each of the nodes need to carry out. We plan to complete this part by the end of third week. Then we will implement the data communication between Lambda instances using each of the services, which requires tuning the communication algorithm with the service characteristics. This part is planned to be done by the end of the 6th week. At the end, we will measure the parameters such as throughput, cost and latency for each. We will give a comparison of the services using the data that we collected.

## Required Resources

For testing and comparing different approaches of communicating data between Lambda instances, we need access to Amazon's Lambda, Simple Queue (SQS), and Simple Storage (S3) services. To use these services, we require $500 in AWS credit. Matt believes he can get access to Amazon credit for this purpose.

## References


[1] Transaction Processing Performance Council. Tpc-h benchmark specification. *Published at http://www. tcp. org/hspec. html*, 21:592–603, 2008.

[2] Sadjad Fouladi, Riad S Wahby, Brennan Shacklett, Karthikeyan Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. In *NSDI*, pages 363–376, 2017.

[3] Masaru Kitsuregawa, Hidehiko Tanaka, and Tohru Moto-Oka. Application of hash to data base machine and its architecture. *New Generation Computing*, 1(1):63–74, 1983.