Implications of Alternative Serverless Application Control Flow Methods

Sterling Quinn
Advised by: Dr. Wes J. Lloyd
School of Engineering and Technology

7th International Workshop on Serverless Computing (WoSC7) 2021
https://www.serverlesscomputing.org/wosc7/papers/p3
Problem

Serverless Application Control Flow (Orchestration)

> Distributed applications require orchestration

> Different methods of orchestration are available

> The cost and performance implications of different orchestration patterns are not apparent
Definitions

> **Amazon Web Services (AWS)**
  - Lambda - Function-as-a-Service platform
    > Allows users to configure the memory allocated to a function affecting performance and cost
  - Aurora - Serverless Relational Database
    > Implementation based on MySQL 5.6
  - S3 - Object storage service
  - EC2 - Compute service (e.g. VMs -as-a-service)

> **Serverless Application Analytics Framework (SAAF)**
  - Supports profiling workload performance, resource utilization (e.g. CPU, memory, disk, network I/O), and infrastructure
  - Developed by UW Tacoma Cloud and Distributed Systems Research Group
Use Case

Serverless Data Processing Pipeline

> **Transform Load Query**
  - **Step 1: Transform (T)**
    > Downloads CSV file from Amazon S3
    > Removes duplicate rows, adds order processing and gross margin columns
    > Uploads transformation result (new CSV file) to S3
  - **Step 2: Load (L)**
    > Loads CSV to Amazon Aurora in batches of 1,000 rows
  - **Step 3: Query (Q)**
    > Performs 5 separate aggregation queries using UNION
    > Saves results to S3
    > Performs SELECT * query
Control Flow Methods

> Orchestrate function calls from client

> Client could be any computer
Control Flow Methods - 2

State-Machine

> AWS Step Functions
> AWS manages transitions and data passing
> Billed per transition
> Asynchronous
> Poll for completion
Control Flow Methods - 3

Microservice Controller

> Orchestrate with controller function
> Suffers from “double billing”\(^1\)
> Controller can be run at low memory

---

Control Flow Methods - 4

Event-Based-Triggers

> Initial function triggered by client
> S3 Event triggers next step
> Asynchronous
> Poll for completion
Metrics

- **Function runtime**: Sum of runtime of T, L, and Q functions collected by SAAF

- **Pipeline runtime**: Elapsed time from the start of T, to the end of Q

- **Latency**: pipeline runtime - function runtime
  - captures transition time of T $\rightarrow$ L, and L $\rightarrow$ Q
Performance Implications

Experiment: 10 runs of the TLQ pipeline using each control flow method, for 100, 1,000, 5,000, 10,000, 50000, 100,000, and 500,000 row datasets

Considerations:
> Event-Based-Triggers incurs additional function runtime to retrieve previous function output JSON

> Event-Based-Triggers and State-Machines are asynchronous

> Microservice Controller and VM-Client are synchronous
RQ-1: Performance Implications
Cost Implications

Considerations:
> Additional runtime for Event-Based-Triggers results in additional cost

> Asynchronous methods (Event-Based-Triggers and State-Machines) have constant transition cost

> Synchronous methods (Microservice Controller and VM-Client) transition cost scales with pipeline runtime
RQ-2: Overall Cost Comparison

Control Flow Methods

- event-triggers
- microservice
- state-machine
- VM-client
Best Control Flow Method?

**Step Functions**
The best combination of developer experience, performance, and price

**Event Based**
Performance and development penalties

**Client**
Client control flow appropriate for smaller use cases

**Microservice controller**
Serverless synchronous control flow is expensive
Questions?