Serverless II a

A Study of Database Techniques over Serverless Cloud Platforms

Milestone 1

Nikhil Gupta, Sahithi Kodali, Sachit Kothari

PURDUE UNIVERSITY | Department of Computer Science

Agenda

1. Introduction
2. Objectives
3. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider
4. Lambada: Interactive Data Analytics on Cold Data using Serverless Cloud Infrastructure
5. Starling: A Scalable Query Engine on Cloud Function Services
6. Next Steps
7. Q&A

Introduction

What is Serverless cloud platform?

- A cloud computing application development and execution model allowing developers to build and run applications without having to manage servers.
- Providers need to achieve high performance at lowest cost.
- A function is started quickly if the code is in memory (warm start) but slower if it needs to be brought in from persistent storage (cold start).
- Always keeping it warm (in memory) is far too expensive.

Treat your servers like pets - dispose of them humanely.

Reference: Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. USENIX Annual Technical Conference, 2020.

Objectives

- Understand characterizing and optimizing serverless workload on a large cloud provider
  - Challenges in optimizing serverless workload
  - How to mitigate these challenges for efficient management of workload on serverless platforms
- Explore existing serverless architectures, their advantages and limitations
  - SAND
  - Lambada
  - Starling
  - And more …

Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Function as a Service

- Function as a Service (FaaS) offers an intuitive and event based interface for developing cloud based applications. Users upload code of their functions to the cloud which are executed by triggers. Provider allocated resources, handles billings and provides high performance on their own.
- The original keep alive policy has fixed keep alive time for all functions. Every provider had a time set on their own for this like 10 or 20 minutes.
- The fixed keep alive policy is not ideal since every program has different needs for pre warming and keep alive and the standard policy leads to wasted resources.

Reference: Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. USENIX Annual Technical Conference, 2020.

Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Contributions

- A detailed characterization of the entire Faas workload at a large cloud provider (the kinds of triggers, their invocation frequency, etc)
- A new policy for reducing the number of cold start function executions at low resource cost.
- Extensive simulation and experimental results based on real traces showing the benefits of the policy. (Comparing both policies)
- Overview of implementation in Azure Functions. (How the new policy is implemented and its results)
- A large sanitized dataset containing production FaaS traces. (A dataset uploaded to github of all function invocations in two weeks)

Reference: Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. USENIX Annual Technical Conference, 2020.

Lambada: Interactive Data Analytics on Cold Data using Serverless Cloud Infrastructure

- Cloud Data Processing Evolution:
  - Evolution of cloud services from providing infrastructure and development platforms (IaaS, PaaS) to software functionality (SaaS) and finally to serverless computing (FaaS).
- IaaS vs. FaaS in Data Analytics:
  - IaaS offers cost advantage: always-on resources, economies of scale, resource efficiency, predictability, and reduced startup overhead
  - FaaS excels in interactive queries: low-latency responses and rapid scaling for sporadic workloads.

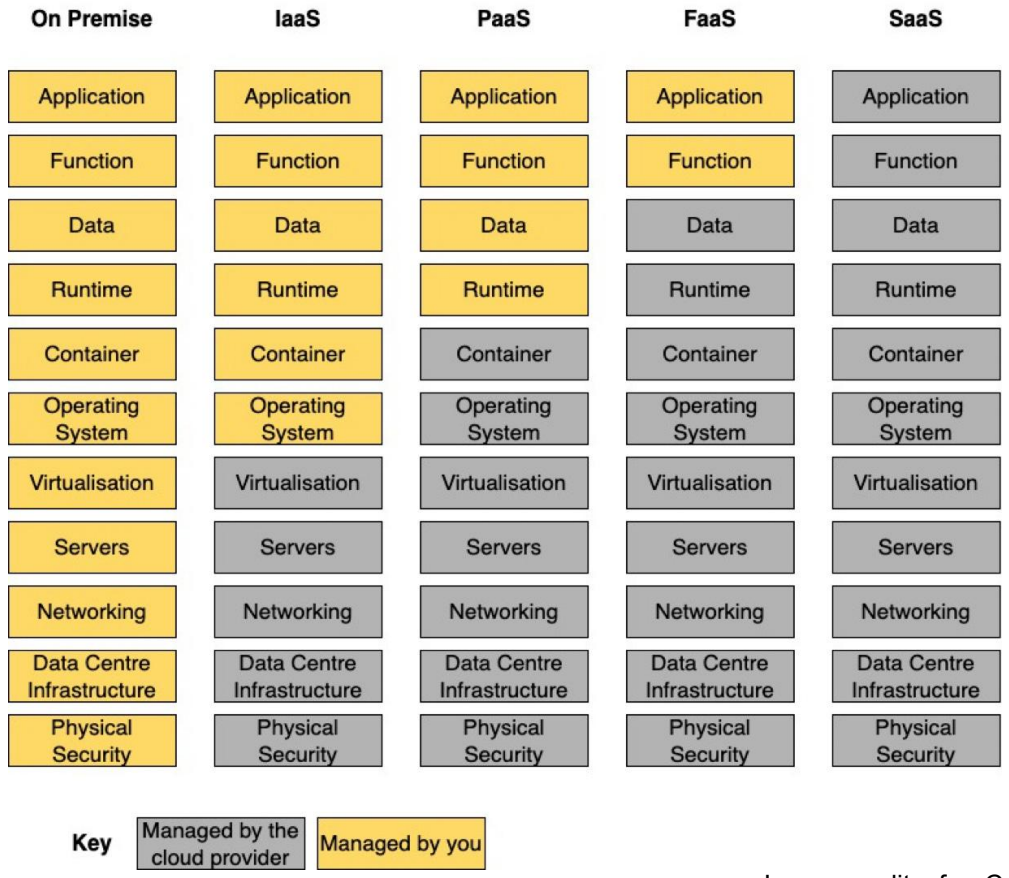


Image credits: freeCodeCamp

Reference: Ingo Müller, Renato Marroquín, and Gustavo Alonso. 2020. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. *ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*.

Lambada: Interactive Data Analytics on Cold Data using Serverless Cloud Infrastructure

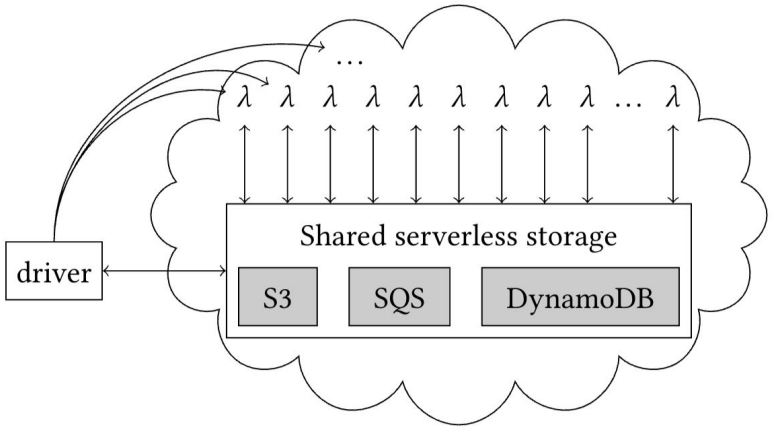
- FaaS Limitations:
  - Limited intra-function parallelism: Functions can create only a limited number of threads, which can impact the overall performance and scalability of the system.
  - Inability to accept network connections: One of the most severe limitations of serverless computing for data analytics. This makes direct communication between function invocations impossible, which can hinder certain types of data processing and analysis tasks.
  - Lack of control over function scheduling: Serverless computing platforms like AWS Lambda do not provide fine-grained control over the scheduling of functions. This can result in suboptimal resource allocation and inefficient execution of data processing tasks.
  - Cost implications: The pricing models for serverless infrastructure incur costs for resource utilization, which can be optimized but may also result in higher expenses for bugs and corner cases.

Reference: Ingo Müller, Renato Marroquín, and Gustavo Alonso. 2020. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. *ACM SIGMOD International Conference on Management of Data (SIGMOD '20).*

Lambada: Interactive Data Analytics on Cold Data using Serverless Cloud Infrastructure

Enters Lambada

- Lambada is a scalable data processing system that leverages serverless cloud infrastructure for interactive analytics on cold data.
- It achieves faster and more cost-effective performance compared to commercial Query-as-a-Service (QaaS) offerings.
- The system exploits intra-function parallelism and reduces invocation time to handle a large number of functions efficiently.
- It optimizes input block size and designs an efficient scan operator to fully utilize network bandwidth.
- Lambada implements automatic optimizations through rewrites of intermediate representation and optimized native code.



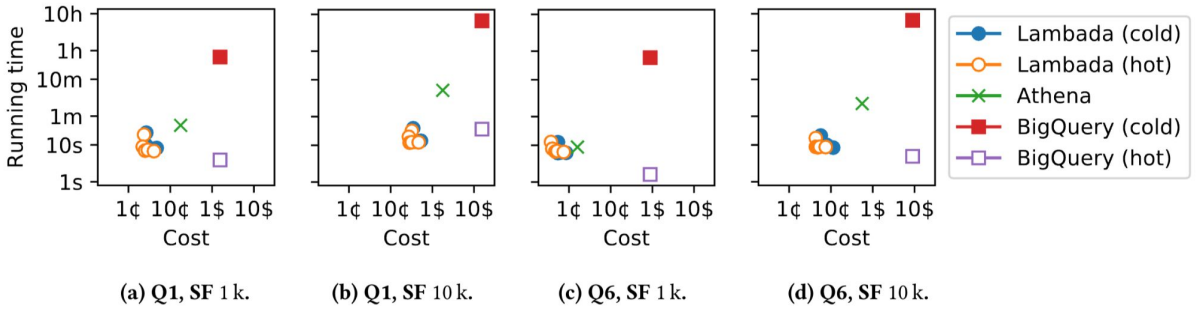


Figure 12: Comparison of Lambada (using $F = 1$ and varying M) with commercial QaaS systems.

Reference: Ingo Müller, Renato Marroquín, and Gustavo Alonso. 2020. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. *ACM SIGMOD International Conference on Management of Data (SIGMOD '20).*

Starling: A Scalable Query Engine on Cloud Function Services

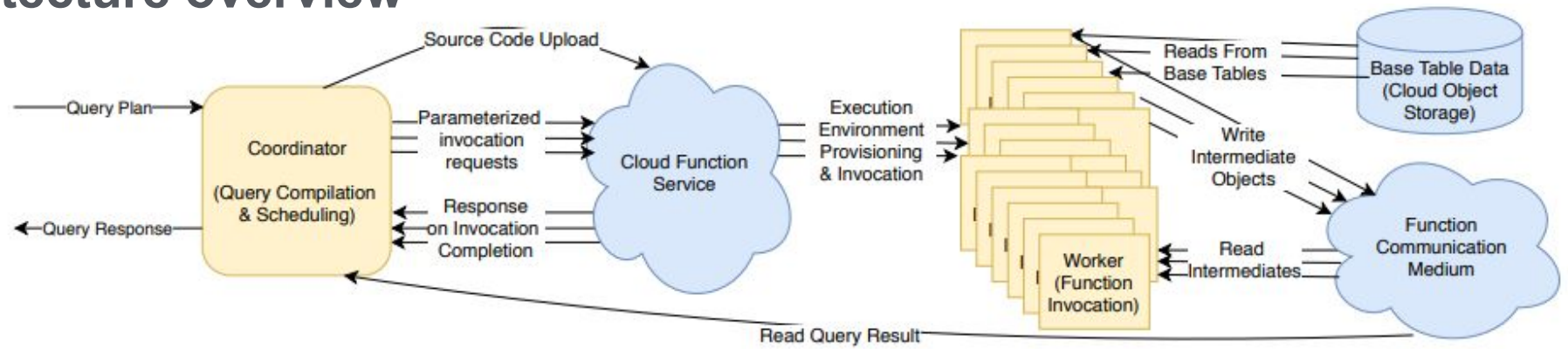
What is Starling?

- Starling is a query execution engine built to run on serverless platforms by utilizing the cloud function services like AWS Lambda and Azure.
- It can handle the problems with AWS Lambda and Azure like cloud functions such as:
  - Managing hundreds of tiny stateless resource-constrained workers (computational units performing tasks with limited resources)
  - Handling stragglers i.e, processes that take longer to complete than other tasks within a parallel computation
  - Data shuffling (process of redistributing data across workers) through opaque cloud services
- Other important challenges that need to be handled in a query execution engine are query optimization, cost reduction, increase elasticity, latency minimization, increase resource utilization and the overall performance.

Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Starling: A Scalable Query Engine on Cloud Function Services

Architecture overview



- The architecture aims at processing the query plan submitted by users and send back the response to the query.
- Includes three main parts: Coordinator, Cloud Function Service and the Worker.
- The coordinator compiles the query and uploads query plan to function service and schedules the task.
- The function service provides execution environments & invocation to the worker.
- The worker reads inputs form the base table data/communication medium (a shared storage to handle stateless functions) and writes to the communication medium, which sends the result to the user once all tasks are completed successfully.

Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Starling: A Scalable Query Engine on Cloud Function Services

Results

- High resource utilization: Maps tasks to function invocations to let user pay for the computing resources their query uses.
- Cost reduction: Pay-by-request model of services reduces the cost and improves the performance by taking advantage of on-demand elasticity of cloud storage services.
- Minimize query latency: Introduced a tuned model to detect straggling requests by optimizing queries based on ad-hoc workloads and does not require loading of data.

System	Does not require loading	Pay by query	Tunable performance
Amazon Athena	✓	✓	✗
Snowflake	✗	✓*	✓
Presto	✓	✗	✓
Amazon Redshift	✗	✗	✓
Redshift Spectrum	✓	✗	✓
Google BigQuery	✓	✓	✗
Azure SQL DW	✓	✗	✓
Starling	✓	✓	✓

Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Next Steps

- Explain the three papers discussed in much detail for the next milestone.
- Find and present more related works in connection with the research papers.
- Assess the strengths and weaknesses of the different approaches.
- Understand how this research translates to real world application applications in cloud services. (e.g. AWS Athena)

THANK YOU

Q&A