Serverless II a

A Study of Database Techniques over Serverless Cloud Platforms

Milestone 4

Nikhil Gupta, Sahithi Kodali, Sachit Kothari

PURDUE UNIVERSITY | Department of Computer Science

A revision of all covered topics

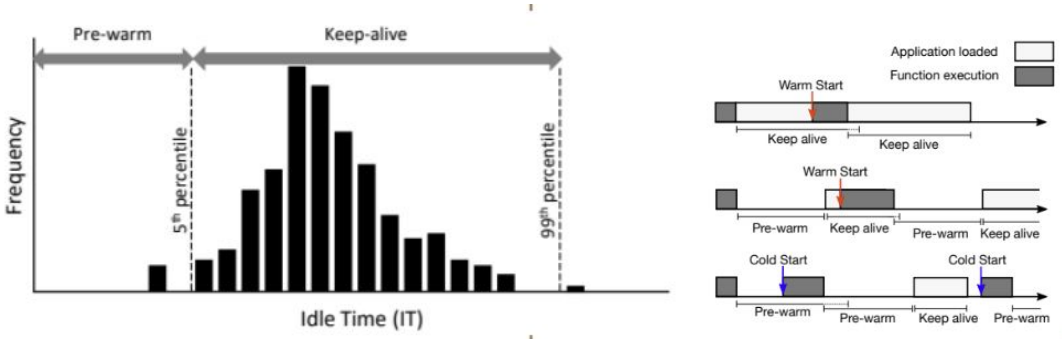
Serverless cloud platforms

- 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.
- Issue at hand: Optimize cold vs warm starts.
- Code in memory: Warm start.
- Code has to be brought in from memory: Cold start.
- Always in memory: Unrealistically expensive for provider.

Mohan, A., Sane, H., Doshi, K., Edupuganti, S., Nayak, N., & Sukhomlinov, V. (2019). Agile cold starts for scalable serverless. In 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19).

Two papers describe two different solutions for this

- Hybrid histogram policy
- Policy has 3 parts: Histogram capturing each applications ‘idle’ times. Standard keep-alive approach when histogram is not representative. Time series forecast component for when histogram does not capture most idle times.



- Pause Container Pool Management (PCPM)
- PC’s have no data retention and take negligible memory. Keep a large number ready.
- Network is largest setup time, so keep network connections pre setup.

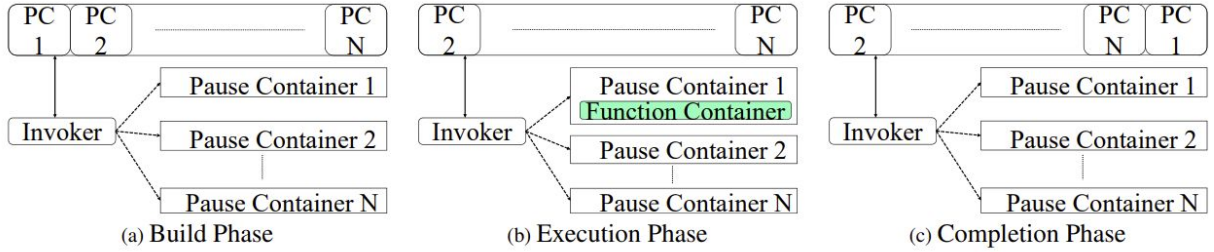
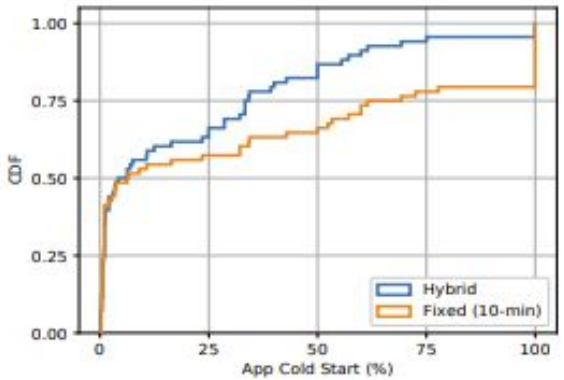


Figure 5: Different phases of Pause Container Pool Manager (PCPM)

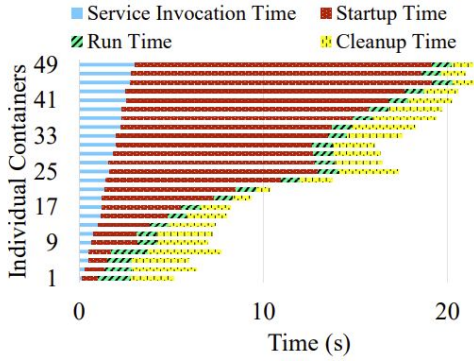
Mohan, A., Sane, H., Doshi, K., Edupuganti, S., Nayak, N., & Sukhomlinov, V. (2019). Agile cold starts for scalable serverless. In 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19).

Results of described methods

- Reduced memory consumption of worker containers by 15.6%
- Reduced the average and 99-percentile function execution time 32.5% and 82.4%, respectively.



- Evaluated with unmodified Open Whisk. 80% improvement in execution time, several order in magnitude of memory saved compared to pre warming containers.



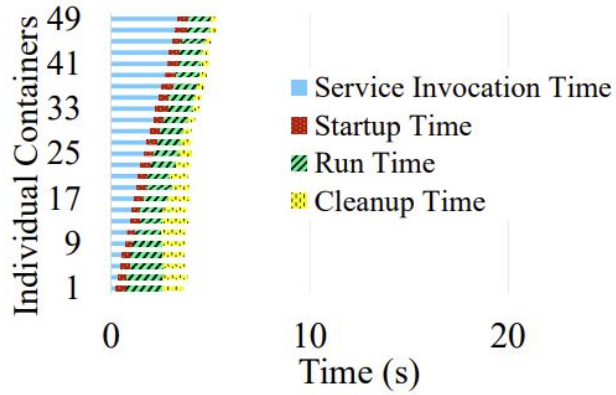


Figure 3: Timeline of 50 concurrent functions with alternate containers. Startup time accounts for 90% of total time.

Container Type	Environment Dependency	Function Dependency	Mem Usage (MB) (50 Containers)	Total Time (s) (50 Functions)
Cold	NO	NO	0	20.01
Warm	YES	YES	1600	0.78
Pre-Warm	YES	NO	1500	1.05
PCPM	**NO**	**NO**	**2**	**4.84**

Mohan, A., Sane, H., Doshi, K., Edupuganti, S., Nayak, N., & Sukhomlinov, V. (2019). Agile cold starts for scalable serverless. In 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19).

Comparing SAND and Lambada

- Focus
  - SAND: FaaS workload efficiency.
  - Lambada: Serverless data analytics.
- Architecture/Policy
  - SAND: Hybrid histogram policy.
  - Lambada: Cloud services integration; serverless functions for compute.
- Performance
  - SAND: Reduces invocation overhead.
  - Lambada: Batch invocation; cloud-native scan and data exchange.
- Use Case
  - SAND: Complex serverless applications.
  - Lambada: Large-scale data analytics.
- Innovations
  - SAND: Efficient function management.
  - Lambada: Distributed data processing components.

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's Approach to Distributed Data Processing

- Parallel Processing Framework:
  - Utilizes multiple serverless workers for concurrent data query processing.
  - Essential for efficiently handling big data analytics tasks across distributed environments.
- Data Partitioning and Management:
  - Distributes data into segments, with each worker processing a specific segment.
  - Leverages Amazon S3 for storing and accessing partitioned data, facilitating distributed access.
- Worker Coordination and Synchronization:
  - Implements coordination through Amazon SQS to manage and synchronize distributed tasks.
  - Ensures all workers process their tasks in harmony and aggregate results accurately.
- Dynamic Scalability:
  - Can scale up or down by varying the number of serverless workers based on the workload.
  - This dynamic scalability is crucial for managing varying computational needs and data sizes efficiently.

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).*

Addressing Challenges in Distributed Computing with Lambada

- Efficient Invocation of Distributed Workers:
  - Efficient scaling of starting a large number of serverless workers.
  - Reduces latency and resource overhead in initializing multiple serverless functions.
- Optimized Data Exchange Mechanism:
  - Employs an S3-based exchange operator for data transfer among distributed workers.
  - Tackles bottlenecks in data exchange, ensuring high performance in distributed data processing.
- Advanced Cloud-native Scan Operator:
  - Facilitates fast and efficient data retrieval directly from cloud storage.
  - Optimally designed to access and process distributed data, enhancing overall data handling efficiency.
- Cost and Resource Management:
  - Balances execution speed against the cost of cloud resources and serverless service limits.
  - Implements cost-effective strategies while ensuring performance requirements are met in distributed environments.

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).*

On to Starling

Lambada vs Starling

- While both Starling and Lambada operate in serverless environments and aim to optimize cloud-based data analytics, they differ in their architecture, design goals, and implementation strategies.
- Lambada is a serverless distributed data processing framework that has a broader scope, addressing a range of challenges in distributed data processing.
- Starling is a query execution engine built on cloud function services that is more focused on interactive query execution with specific design choices to minimize latency and cost.
- **Architectural Approach:** While both use serverless architectures, Starling's design revolves around a small coordinator that performs task scheduling and execution, whereas Lambada uses a more distributed approach with a focus on efficient data processing and exchange mechanisms.
- **Operational Strategy:** Starling's operational strategy is geared towards interactive query processing with a focus on cost-effectiveness and low latency. Lambada's strategy encompasses a wider range of data processing activities, emphasizing the efficient handling of large-scale distributed data analytics tasks.

What Starling approaches make the process of query execution cost effective and low latency?...

Starling: A Scalable Query Engine on Cloud Function Services

Objectives of Starling

- Handle the challenges with FaaS cloud function services such as:
  - Managing hundreds of tiny stateless resource-constrained workers (computational units performing tasks with limited resources).
  - Handling stragglers i.e, workers that take longer to complete than other tasks within a parallel computation.
  - Data shuffling and aggregation (process of redistributing data across workers via stateful queries) through opaque cloud services.

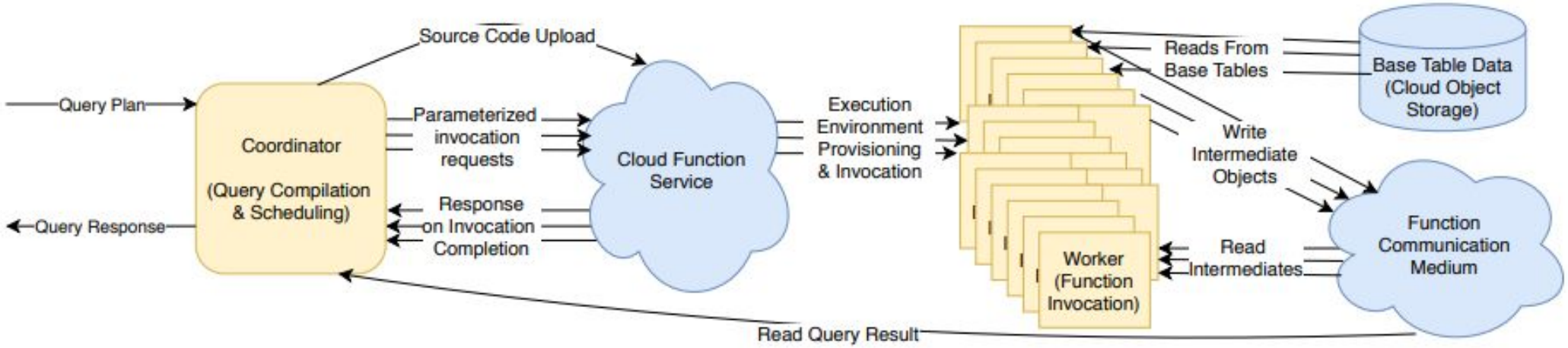
Starling provides data management and query execution strategies to address these challenges to minimize cost and latency…

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

Previously Discussed

- Discussed the Starling's architecture i.e., the working of query engine.
- How query execution, data management is handled in Starling.

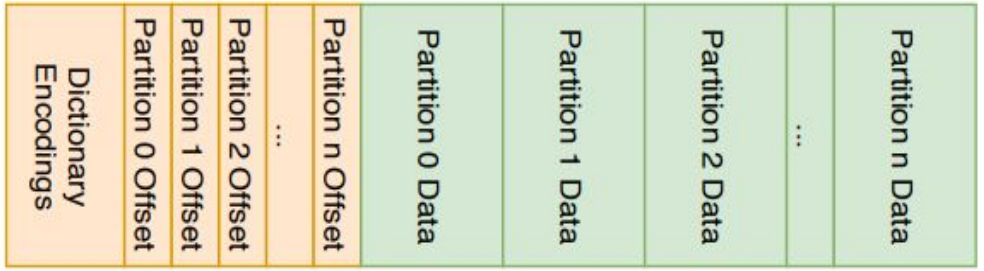


Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Starling: A Scalable Query Engine on Cloud Function Services

Data Management - Key takeaways

- AWS S3 is used for base table data storage and as the communication medium for its low-cost, high-throughput, low-latency, and scalable data exchange solution. S3's high storage latency is overcomed by parallelizing reads approach reducing its time of idleness.
- Agnostic to base table formats. Open Source columnar formats chosen for selective column reading and skipping of data using indexes to improve query performance. Data stored as objects in S3.
- Starling relies on AWS for data shuffling and passing intermediate data between function invocation.
- Optimizes shuffling by writing single partitioned file to S3. Workers write output to a single S3 object in S3 buckets with predetermined keys for readers to poll keys for new object, thereby fetching metadata and required partition.



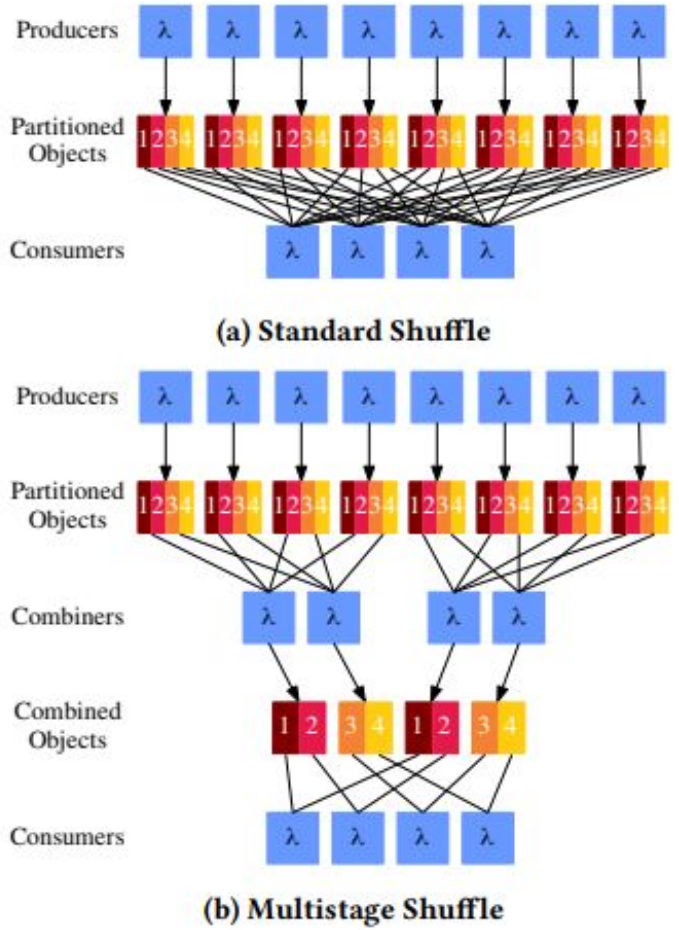
- Partitioned file have metadata & low cardinality strings dictionary encoding. 2 reads to fetch data in partition.

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

Query execution - Key takeaways

- Coordinator takes a query plan, compiles it and uploads query executable to AWS lambda. Each task's input and output object names are determined prior to workers execution to minimize latency.
- Strategies to execute relational operators (joins and aggregation), data shuffling and task scheduling.
- Relational operators are executed as nested loops allowing pipelining and thus reduce query latency.
- Data shuffling handled by allowing tasks to read only relevant partitions of input files. For large scale queries, multi-stage shuffle approach that combines tasks to read contiguous subsets of partitions is implemented to reduce the number of reads significantly.
- Task scheduling and pipelining finetune the number of tasks per stage to find a cost-performance balance. Option to enable/disable pipelining is provided to avoid costs in case of execution delays.

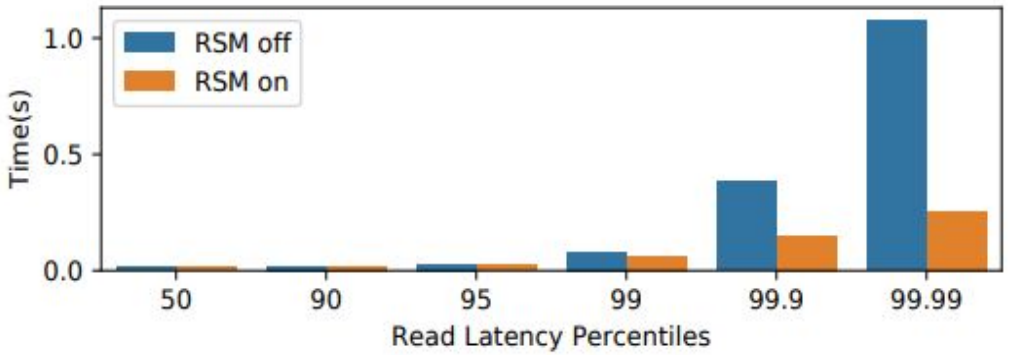


Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Starling: A Scalable Query Engine on Cloud Function Services

Handling Stragglers

- Can occur while reading base table data or exchanging intermediate state, as stages must wait for their inputs to be available before query processing. Straggler optimization leverages the power of two choice technique, a well known theoretical framework using randomization and task duplication.
- Read Straggler Mitigation (RSM)
  - Uses a dynamic approach by detecting delays compared to expected completion times based on observed latency and throughput of S3 requests and AWS Lambda invocations. Workers open new connections and retry requests if responses exceed anticipated duration. Though the read expense is incurred, it significantly improves query latency.

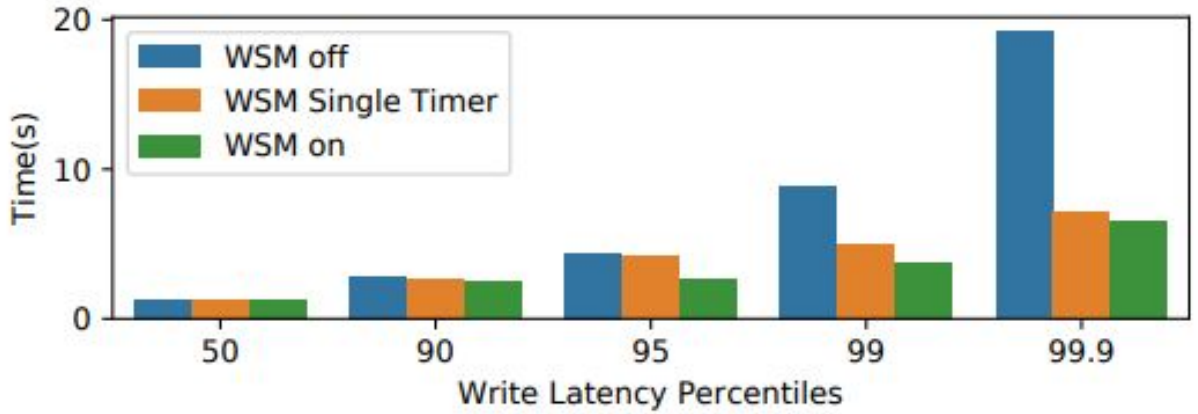


Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

Starling: A Scalable Query Engine on Cloud Function Services

Handling Stragglers

- Write Straggler Mitigation (WSM)
  - Address delay in large S3 write requests by using a model similar to RSM to predict response times and initiate a second write request. Incurs additional write expenses but saves about 2100 seconds of compute time while incurring a cost of 314 seconds only, thus giving low latency within optimized compute resources.
  - Variations of WSM (single timer and full WSM that includes a second timeout) are seen in comparison below.

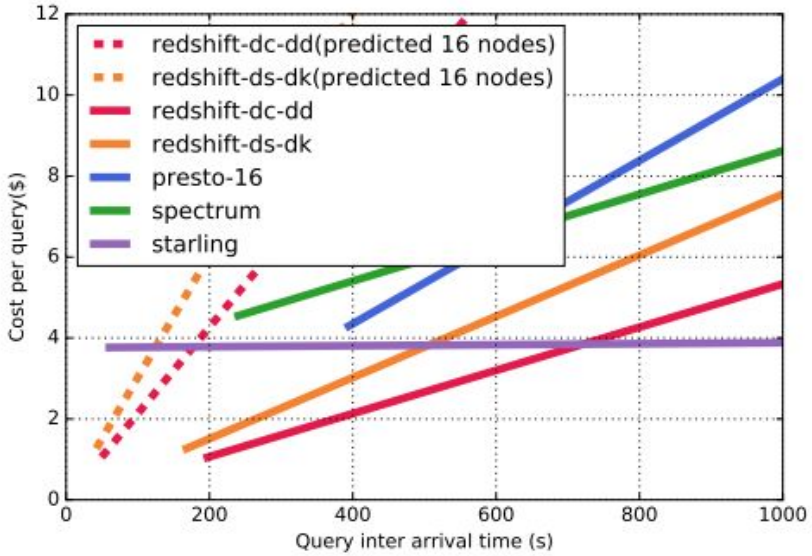


Reference: Perron, Matthew, et al. "Starling: A scalable query engine on cloud functions." Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020.

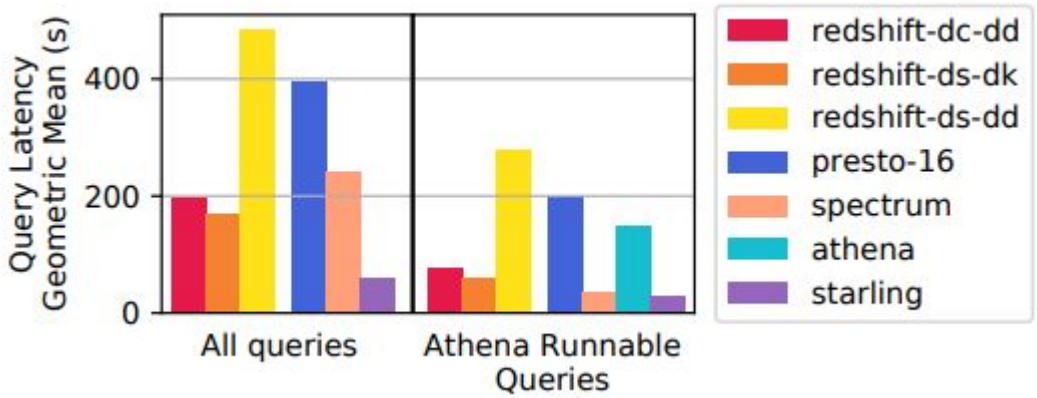
Starling: A Scalable Query Engine on Cloud Function Services

Evaluation of Starling results

- Comparing a few systems that load data from a 10TB dataset, the daily costs incurred with increasing queries is minimal in Starling as query increases. The cost incurred is to read the base table, intermediate data from S3 & AWS Lambda.



- Comparing the query latency of different systems on a 10TB dataset, Starling shows minimal latency with some systems close to Starling, but they have a few query execution conditions that do not make them as ideal choices.



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.

Conclusion

- Discussed the evolution of serverless computing starting with a characterization and handling serverless workloads (cold starts and warm starts), moving through specialized frameworks and engines (Lambada and Starling) that improved data processing and query execution approaches.
- Compared each of the above frameworks/engines to address the challenges, improvements, along with their architecture details.
- Explained the approaches chosen by each of the above frameworks/engines and the technical details behind their implementations, reflecting the dynamic and innovative nature of serverless computing.

THANK YOU

Q&A