Doppler: Understanding Serverless Query Execution

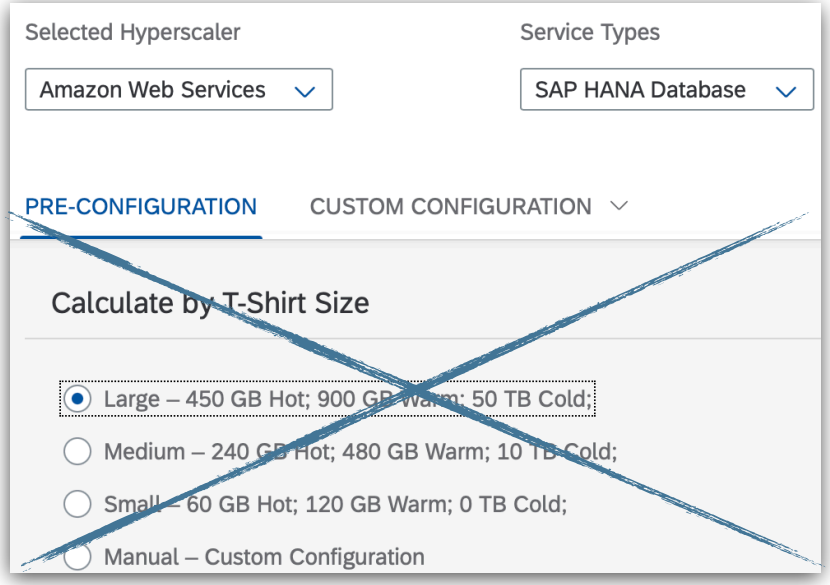
Thomas Bodner, Tobias Pietz, Lars Jonas Bollmeier, Daniel Ritter
Hasso Plattner Institute, University of Potsdam, Germany

SIGMOD Workshop on Big Data in Emergent Distributed Environments
12 June 2022

Serverless Service Paradigm

Economics

- Providers offer services with different economic model
- Customers pay for **consumption** instead of **capacity**
- Customers benefit when **resource utilization** is **low**, i.e., workload prediction and capacity planning are difficult
- Providers **charge extra** to compensate underutilization



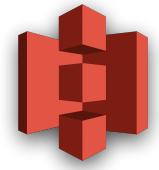
SAP HANA Cloud Capacity Unit Estimator

- Assumption: Providers are better at using resources, because of their scale

Serverless Service Paradigm
Technology

- No burden of provisioning or managing resources (like servers, thus the term *serverless*)
- No scaling, load balancing, failure tolerance, .. in distributed systems
- Infrastructure and database services moving towards consumption-based model

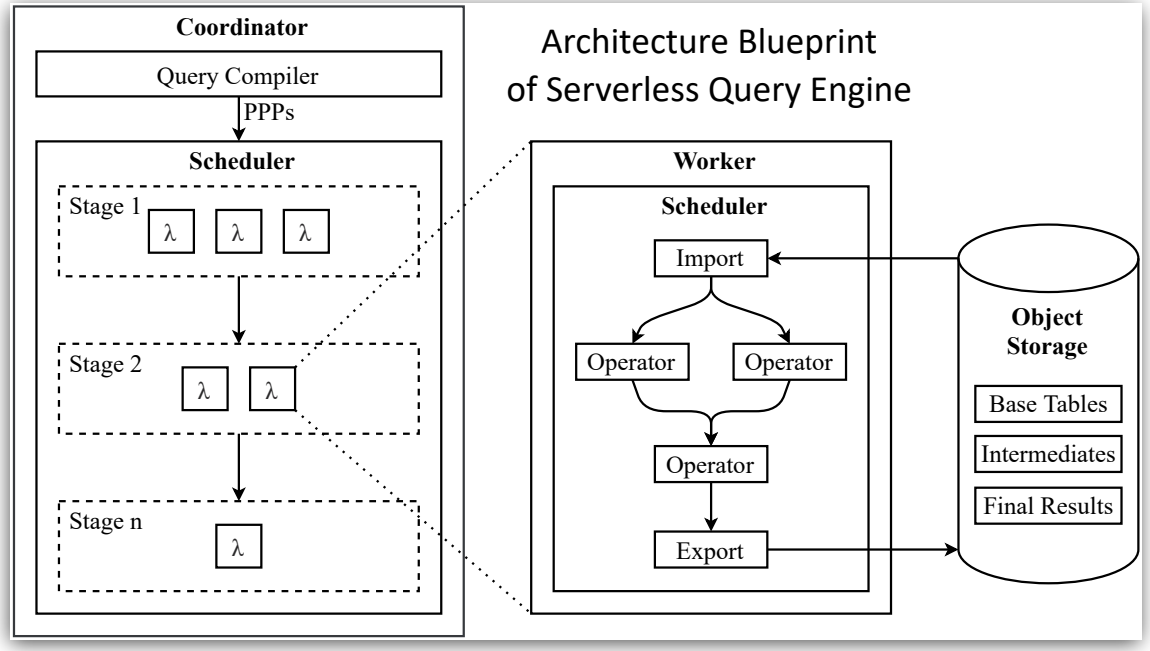
- Serverless infrastructure as an *emergent distributed environment*
  - Serverless storage (e.g., Amazon S3): Scale from bytes to EB, high-throughput, cheap
  - Serverless compute (e.g., AWS Lambda)
    - Execute pieces of code in any programming language
    - Spawn tiny, short-lived, and stateless workers in milliseconds
    - Scale to 10,000s of workers, enabling large-scale query processing


Amazon S3


AWS Lambda

Serverless Query Execution

- FaaS-based, shared-storage database architecture
  - Exploits serverless cloud infrastructure for compute and storage elasticity
  - Demonstrates cost and performance benefits for interactive queries on cold data
- Compute tier runs pipelines of query operators in cloud function workers
- Storage tier holds base tables and intermediate/final results as immutable, column-oriented, compressed objects



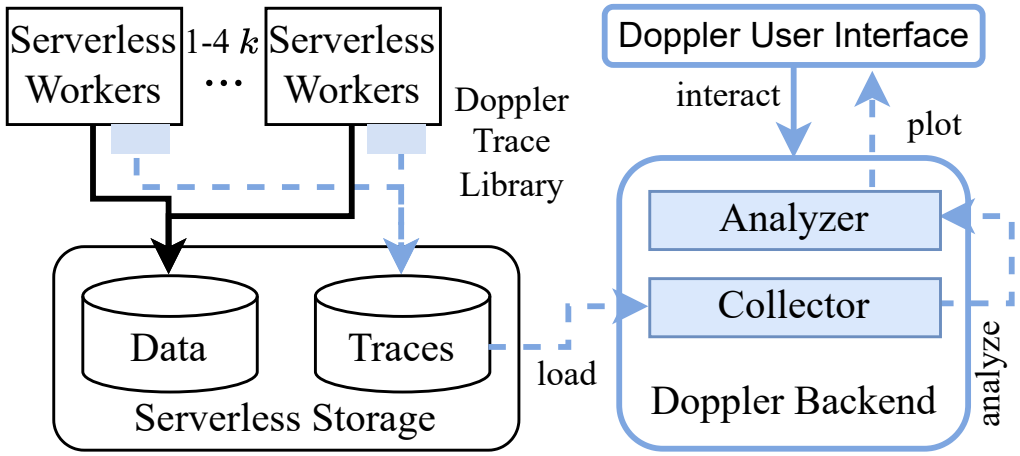
[1] Perron et al. Starling: A Scalable Query Engine on Cloud Function Services. In SIGMOD 2020.
[2] Müller et al. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. In SIGMOD 2020.
[3] Cai et al. Integrated Querying of SQL database data and S3 data in Amazon Redshift. In IEEE Data Engineering Bulletin 2018.

Challenges in Understanding Serverless Query Execution

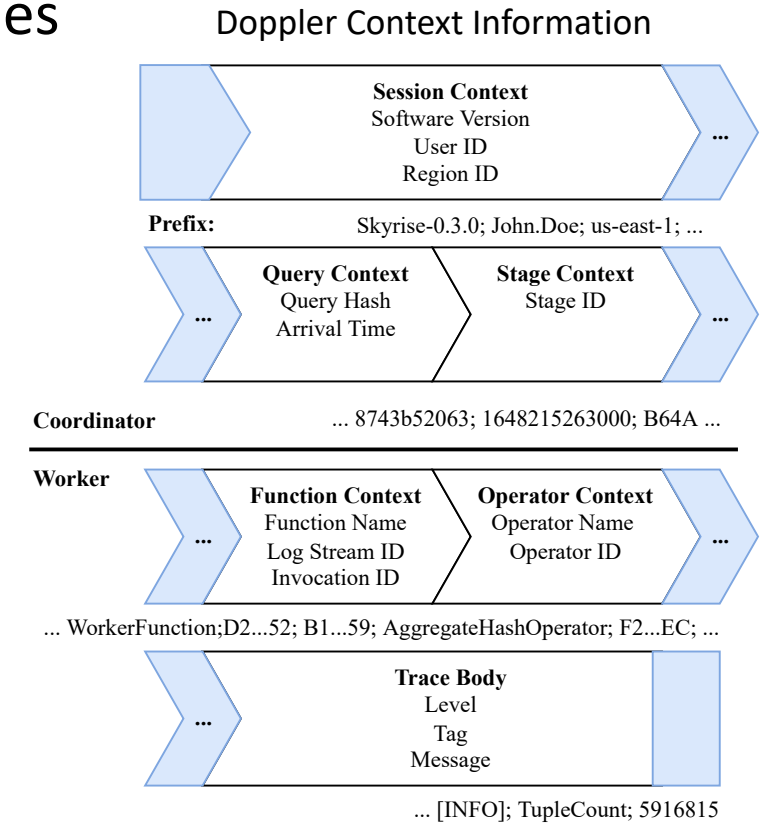
- Analyzing query execution dynamics in cloud-based distributed databases is difficult
- Serverless query execution is more challenging!
  - Impedes understanding, maturity, and adoption of technology
- Observing large ephemeral clusters of stateless workers
- Contextualizing, integrating and analyzing distributed query traces
- Timely and with little cost and performance overhead

Doppler: Debugging and Performance Profiling for Serverless Query Execution

- Toolkit for post-mortem analysis of serverless queries
- Puts query context into traces to relate traces back to semantic parts of a query
- Collects traces from cloud functions via library that wraps most basic persistent log service
- Integrates and analyzes distributed traces



Doppler Architecture



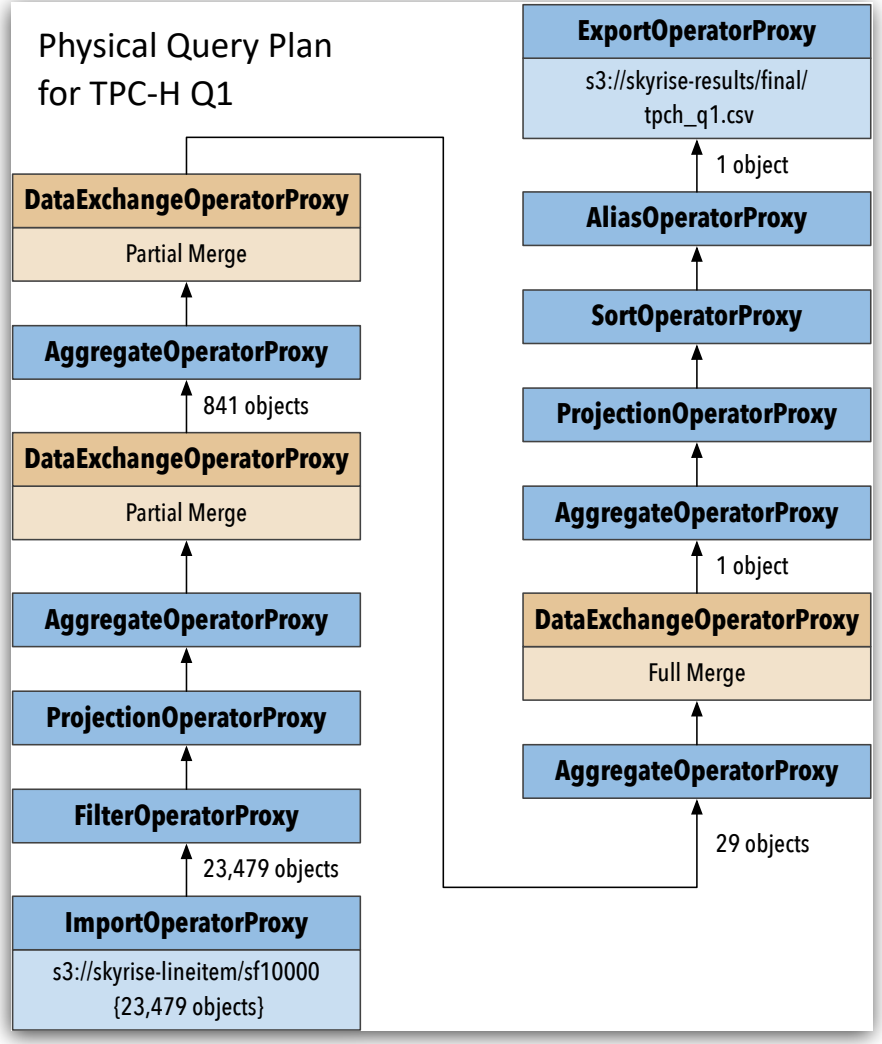
Demonstration of Doppler
Scenarios

- Straggling query workers
  - Stragglers taking much longer than other workers severely impact performance
  - Can occur due to a variety of reasons across hardware/software stack, data skew, ..
  - We induce at random 5s delays into query operators
- Serverless function concurrency limit
  - Cloud functions are subject to provider-side cluster size limits
  - We restrict our worker functions to run at low concurrency
- Local node errors
  - A distributed system is still subject to all local error types
  - We inject at random non-fatal errors into query operators

Demonstration of Doppler

Setup

- We integrate Doppler with the Skyrise query processor
- Standard TPC-H benchmark query #1
- On datasets of scale factors 1 to 10,000 (10 TB)
  - Stored in columnar, compressed ORC files
- In AWS region US-East-1
  - No “serverful” resources running
  - Max. Lambda concurrency 20,000 instances
  - Coordinator instance at 10 GB, workers at 4 GB



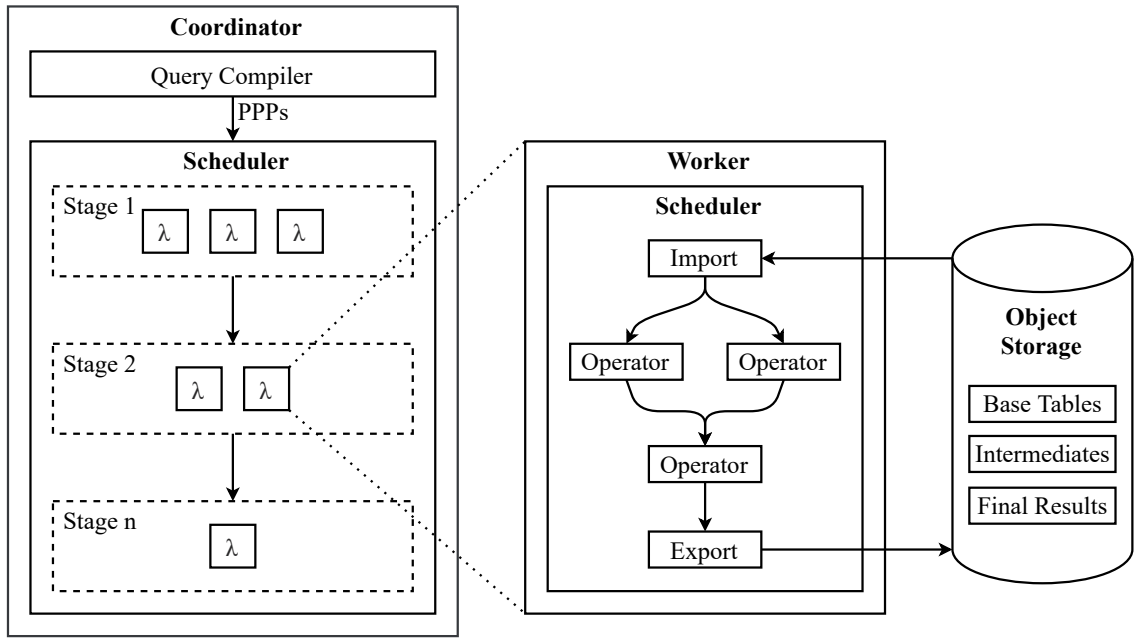
[1] Bodner. Elastic Query Processing on Function as a Service Platforms. In VLDB PhD Workshop 2020.

Demonstration of Doppler

Outline

1. Local CLI invokes serverless coordinator with SQL query string

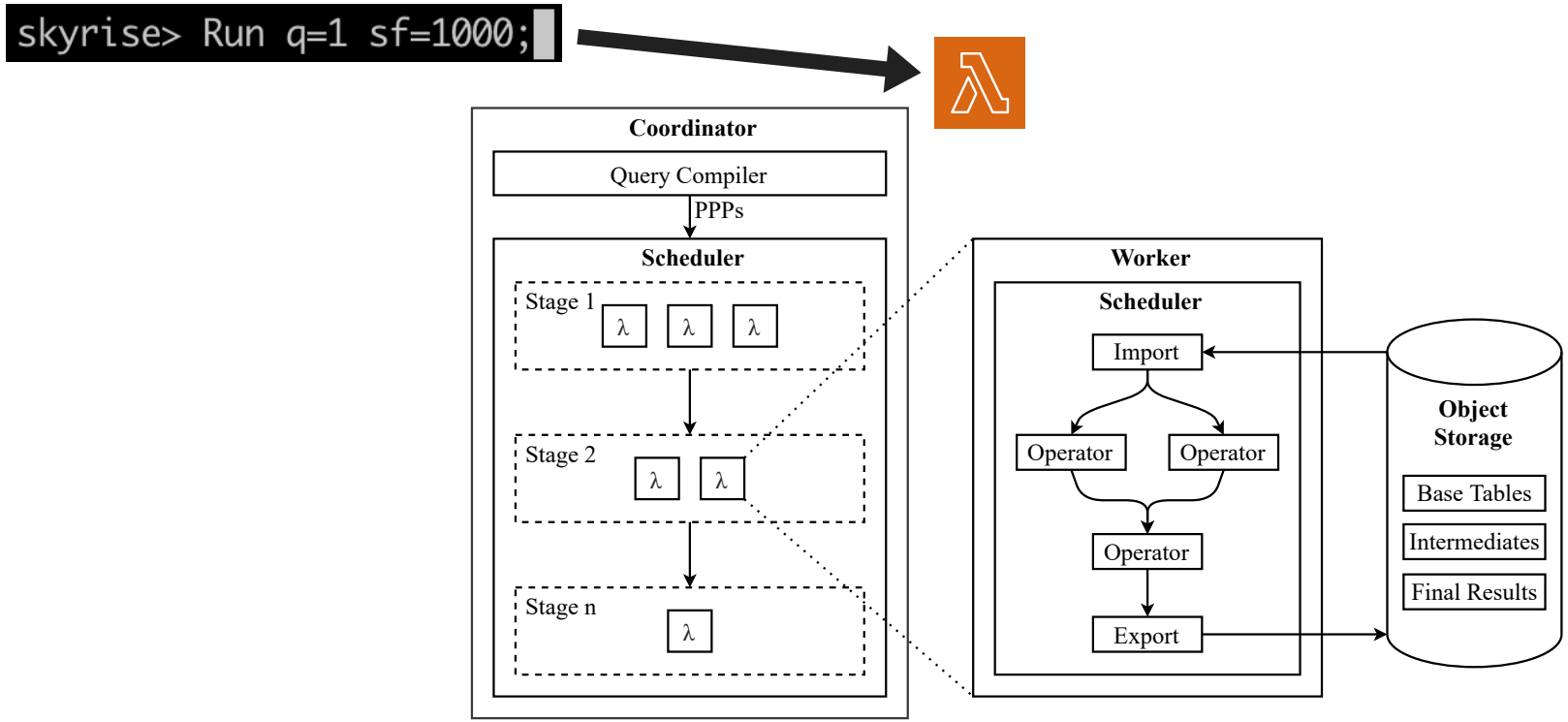
```
skyrise> Run q=1 sf=1000;
```



Demonstration of Doppler

Outline

1. Local CLI invokes serverless coordinator with SQL query string



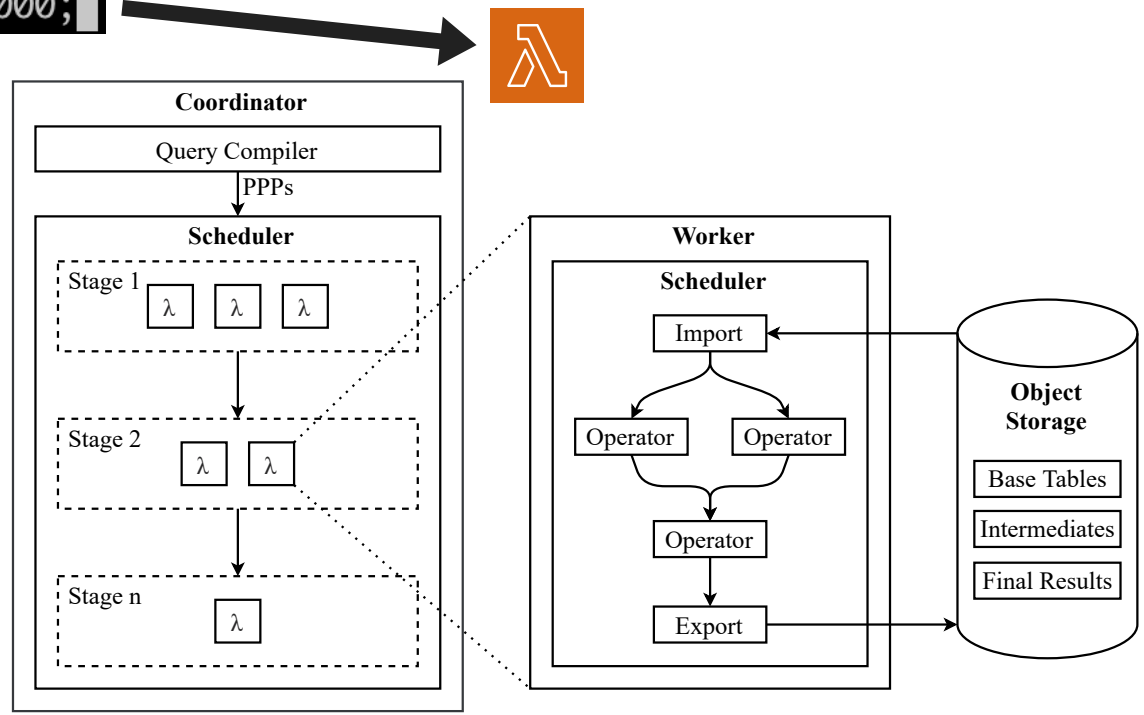
Demonstration of Doppler

Outline

1. Local CLI invokes serverless coordinator with SQL query string

```
skyrise> Run q=1 sf=1000;
```

2. On coordinator, query compiler generates physical query plan fragments



Demonstration of Doppler

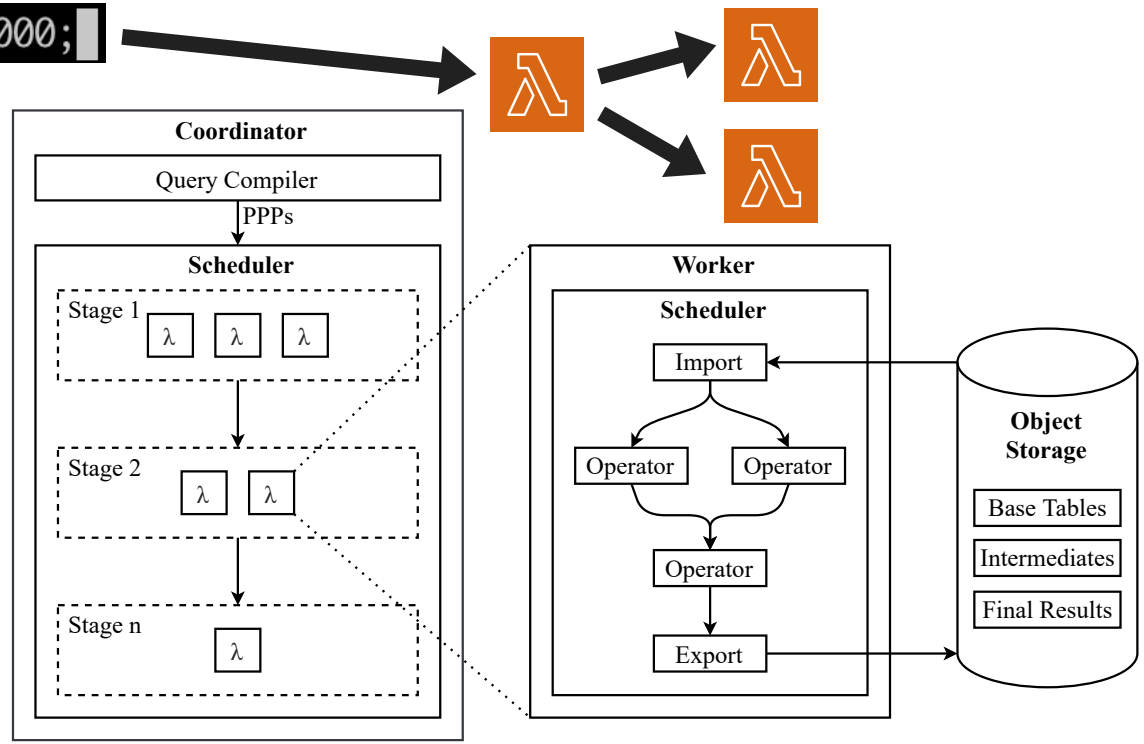
Outline

1. Local CLI invokes serverless coordinator with SQL query string

```
skyrise> Run q=1 sf=1000;
```

2. On coordinator, query compiler generates physical query plan fragments

3. On coordinator, scheduler invokes a worker per plan fragment, tracks their progress



Demonstration of Doppler

Outline

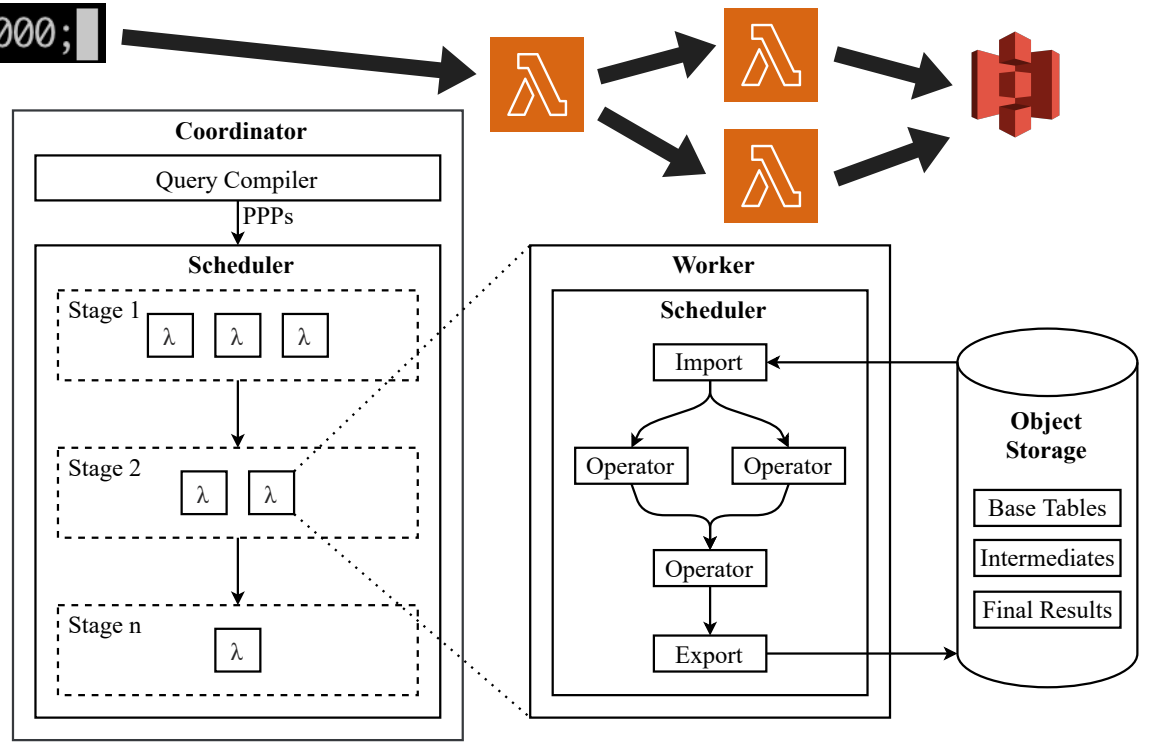
1. Local CLI invokes serverless coordinator with SQL query string

```
skyrise> Run q=1 sf=1000;
```

2. On coordinator, query compiler generates physical query plan fragments

3. On coordinator, scheduler invokes a worker per plan fragment, tracks their progress

4. Workers run their respective fragments, write results to shared storage, inform scheduler about progress



Demonstration of Doppler

Outline

1. Local CLI invokes serverless coordinator with SQL query string

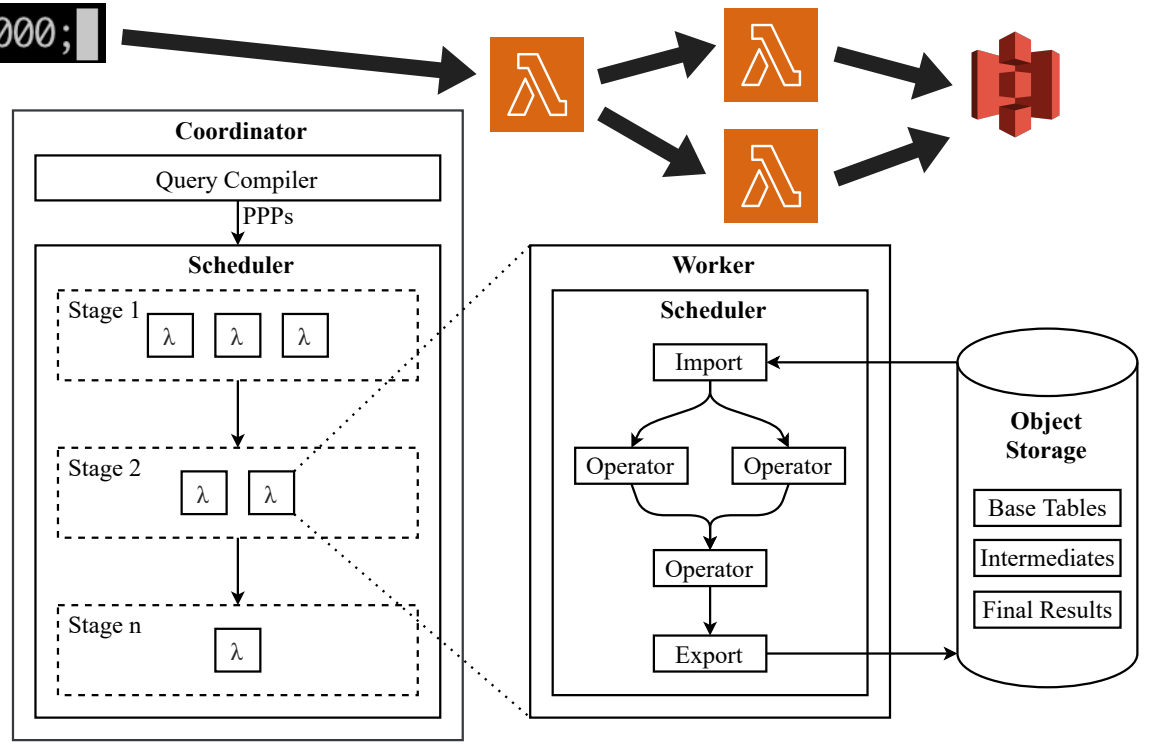
```
skyrise> Run q=1 sf=1000;
```

2. On coordinator, query compiler generates physical query plan fragments

3. On coordinator, scheduler invokes a worker per plan fragment, tracks their progress

4. Workers run their respective fragments, write results to shared storage, inform scheduler about progress

5. Upon query completion, coordinator reports query result and trace handle back to local CLI



Demonstration of Doppler

Outline

1. Local CLI invokes serverless coordinator with SQL query string

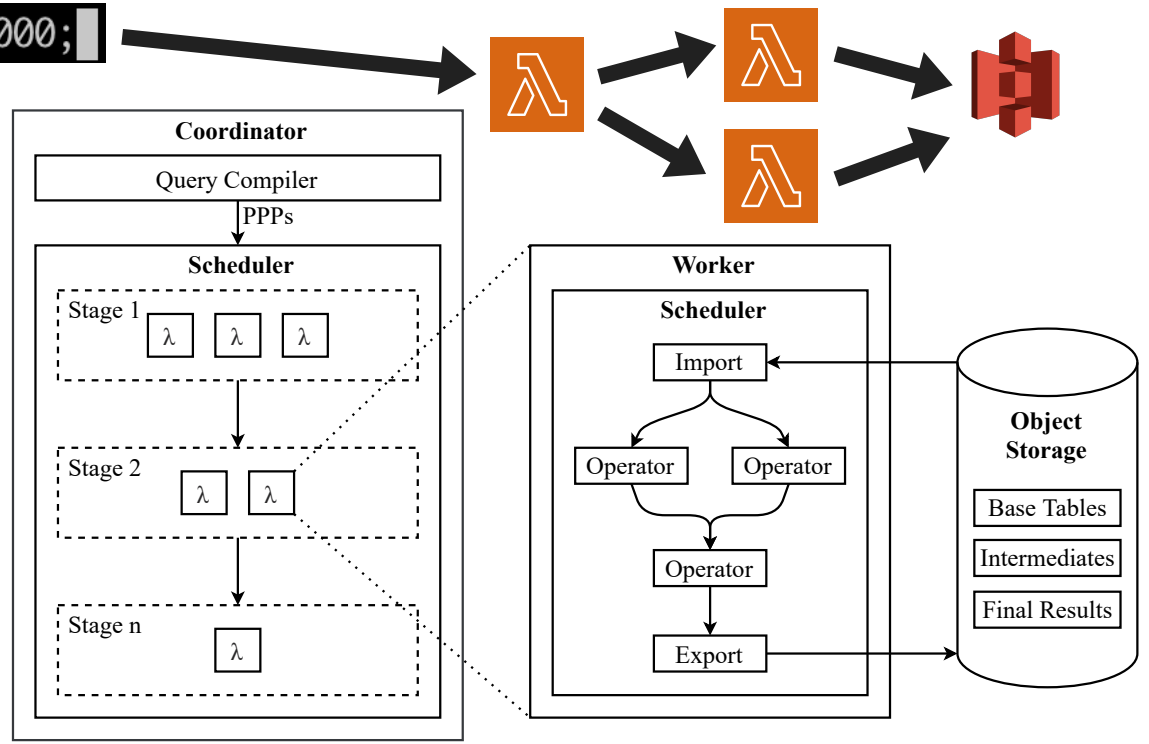
```
skyrise> Run q=1 sf=1000;
```

2. On coordinator, query compiler generates physical query plan fragments

3. On coordinator, scheduler invokes a worker per plan fragment, tracks their progress

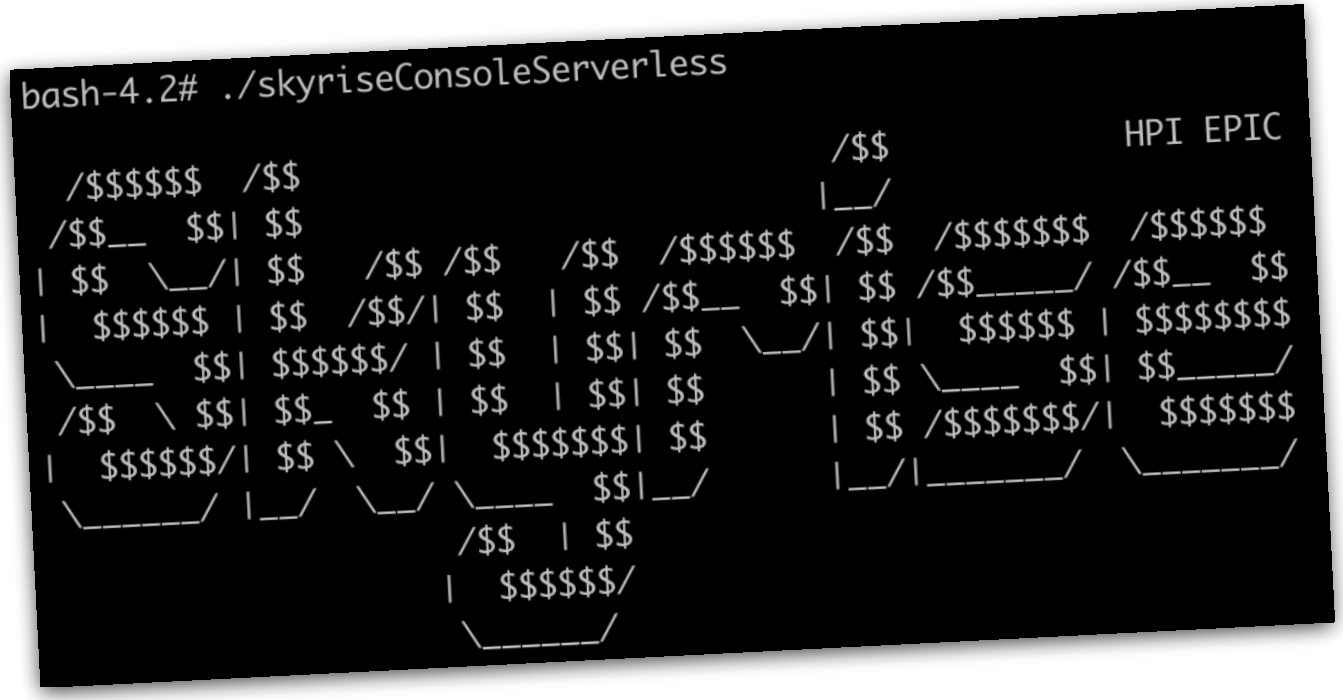
4. Workers run their respective fragments, write results to shared storage, inform scheduler about progress

5. Upon query completion, coordinator reports query result and trace handle back to local CLI

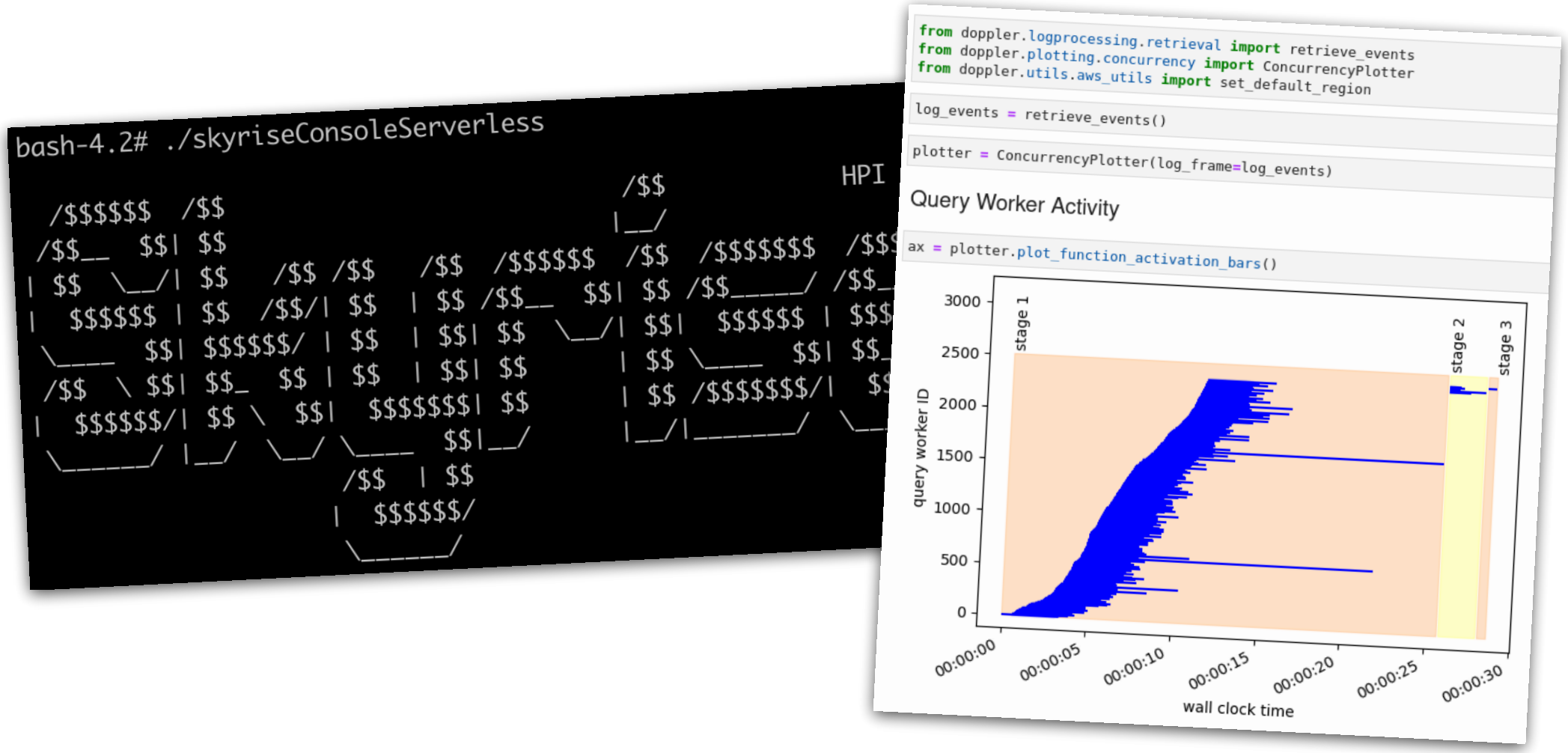


6. Local Doppler backend collects traces based on query handle and analyzes traces

Demonstration of Doppler Debugging and Performance Profiling Toolkit



Demonstration of Doppler Debugging and Performance Profiling Toolkit



Thank you!
Questions?

Backup Slides

Serverless Infrastructure

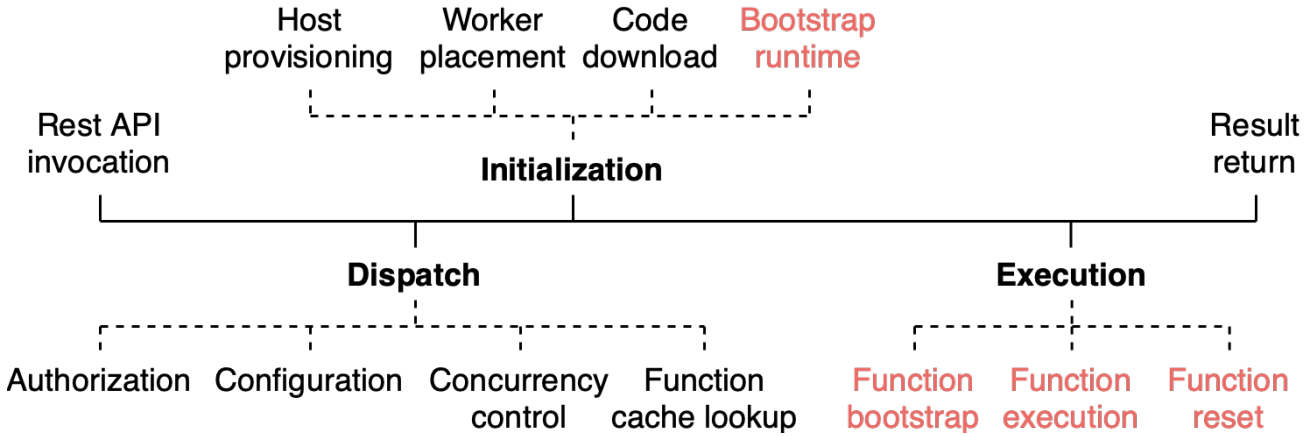
AWS Lambda under the Hood

HPI Hasso Plattner Institut

Lambda Node

- Database Code
- Guest Kernel
- Firecracker Hypervisor
- Host Kernel + KVM
- Nitro ASICs for Acceleration
- Regular C5/C6g EC2 Instance

Lambda Control Plane



Serverless Infrastructure
Lambda Limits and Performance Characteristics

- Service limits
  - Concurrent executions: Up to 10,000s
  - CPU: Up to 6 vCPUs
  - RAM: 128 to 10,240 MB in 1 MB increments
  - Disk: 512 to 10,240 MB in 1 MB increments
  - Network bandwith: 50-300 MB/s
  - Timeout: 15 min

- Performance
  - Hypervisor starts >150 VMs/s/host at <125ms with 5 MB overhead
  - Accumulated network bandwidth: 100s of GB/s, matches memory bandwidth of x86 machines
  - Accumulated compute: TFLOPS for 1,000s of functions, surpasses large x86 machines

Serverless Infrastructure
Current Challenges for Query Processing

- Launch overheads (potentially 10s of seconds)
- Tight resource limits (6 vCPUs, 10 GB RAM/Disk and 15 min runtime)
- Observability of blackbox cloud function services
- Indirect communication due to disabled inbound network connections
- Fault tolerance via transparent re-execution

- All of the above can be dealt with today

- Expectation for them to go away as compute variants converge
- Or to stay as part of the new programming model