Scaling Data Analytics Workloads on Databricks

Spark + AI Summit, Amsterdam

Chris Stevens and Bogdan Ghit

October 17, 2019

databricks

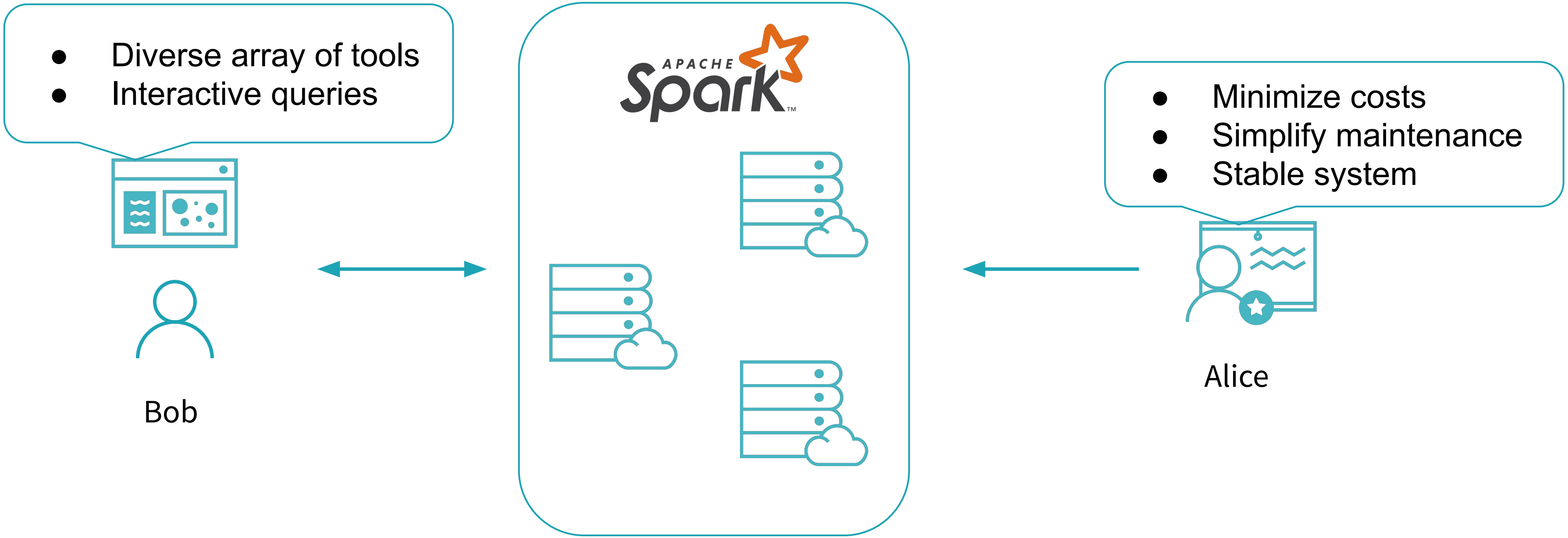
Chris Stevens

- Software Engineer @ Databricks - Serverless Team
- Spent ~10 years doing kernel development

Bogdan Ghit

- Software Engineer @ Databricks - BI Team
- PhD in datacenter scheduling @ TU Delft

A day in the life ...



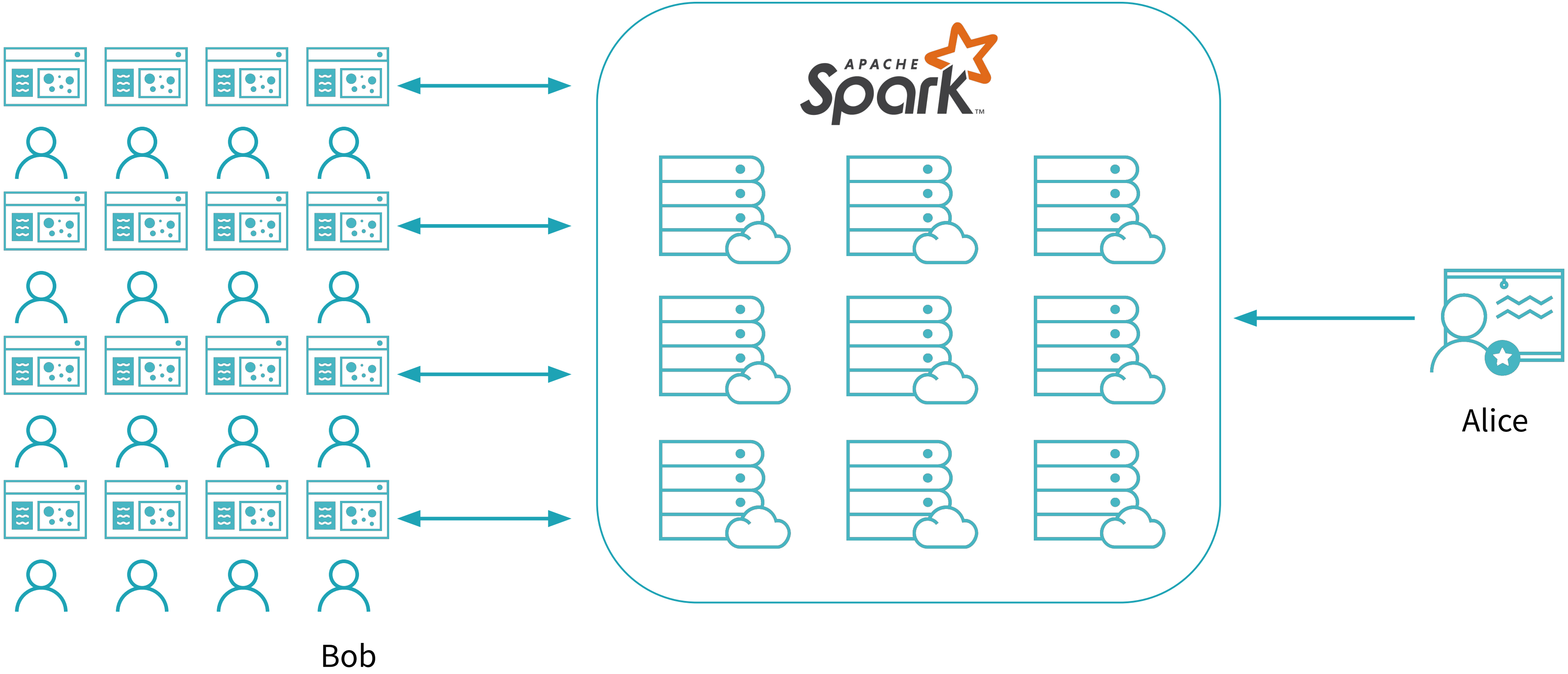
- Diverse array of tools
- Interactive queries

Bob

- Minimize costs
- Simplify maintenance
- Stable system

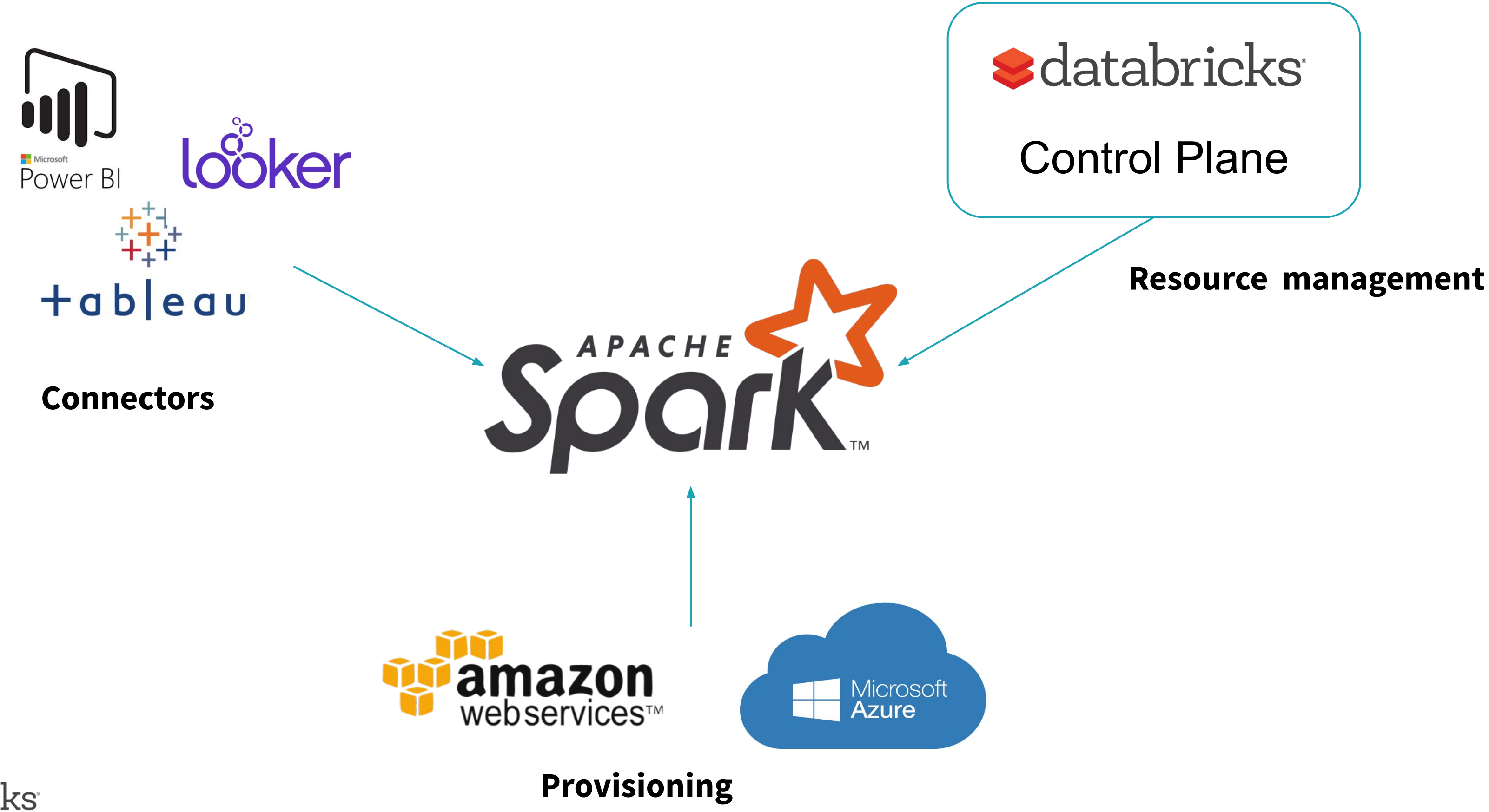
Alice

Reality ...



databricks

This Talk



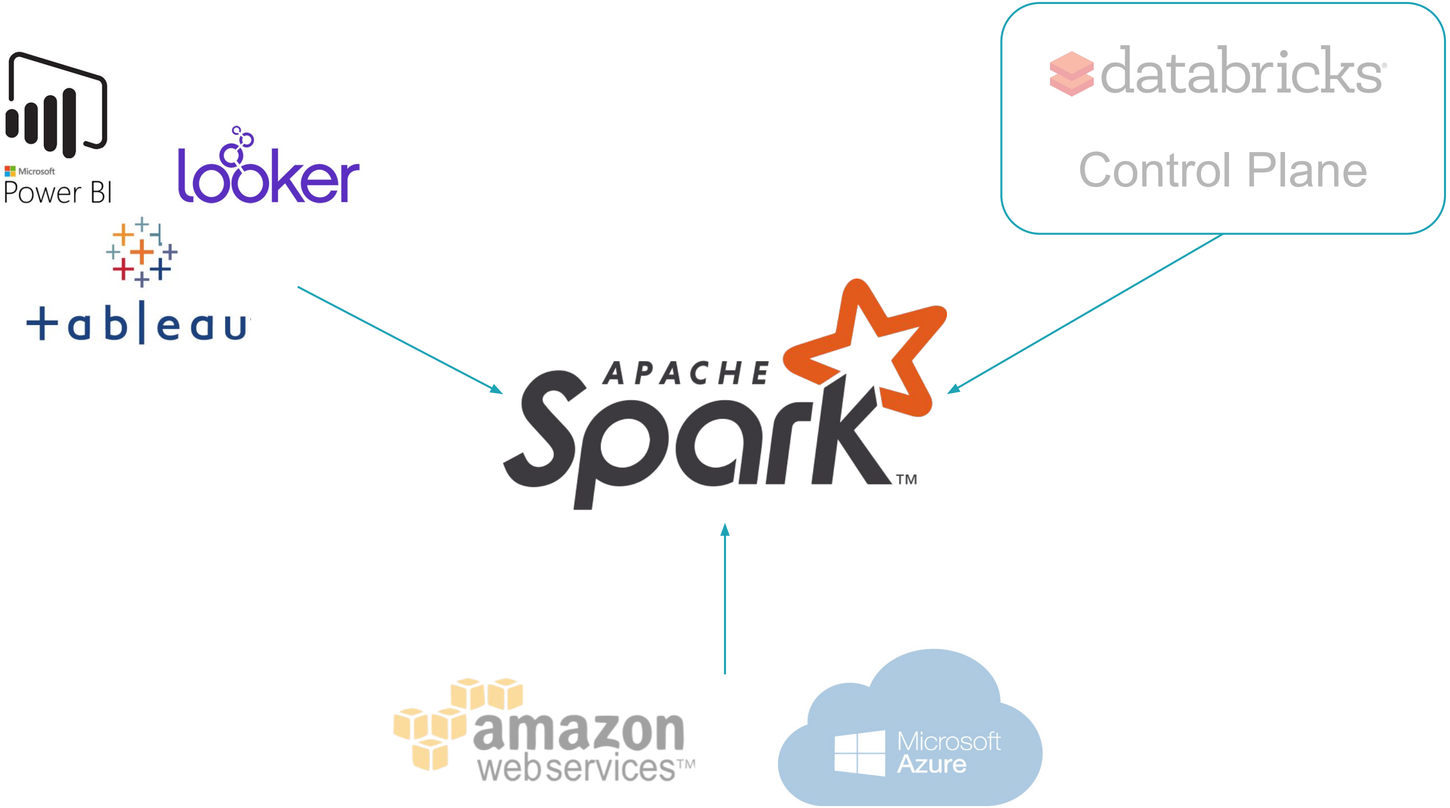
Control Plane

Resource management

Connectors

Provisioning

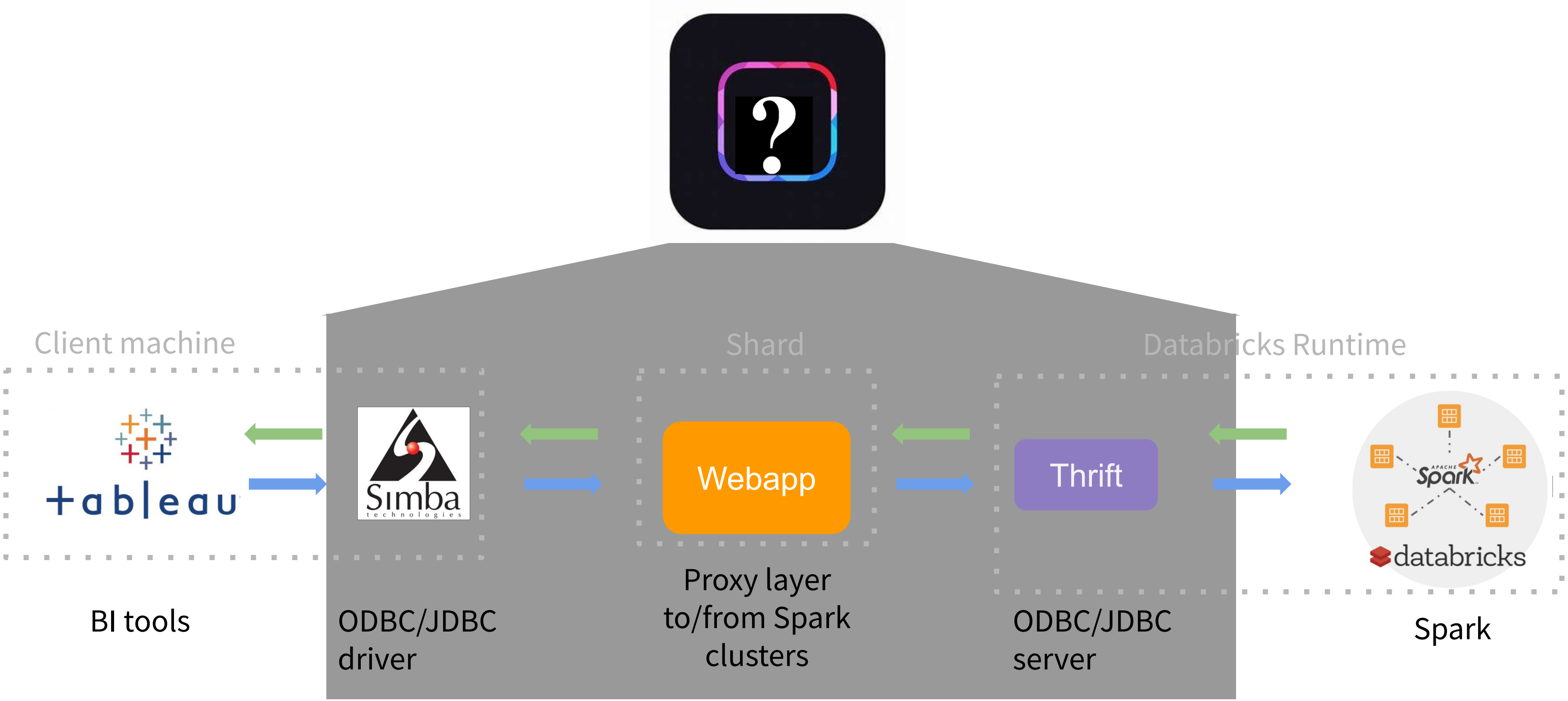
BI Connectors



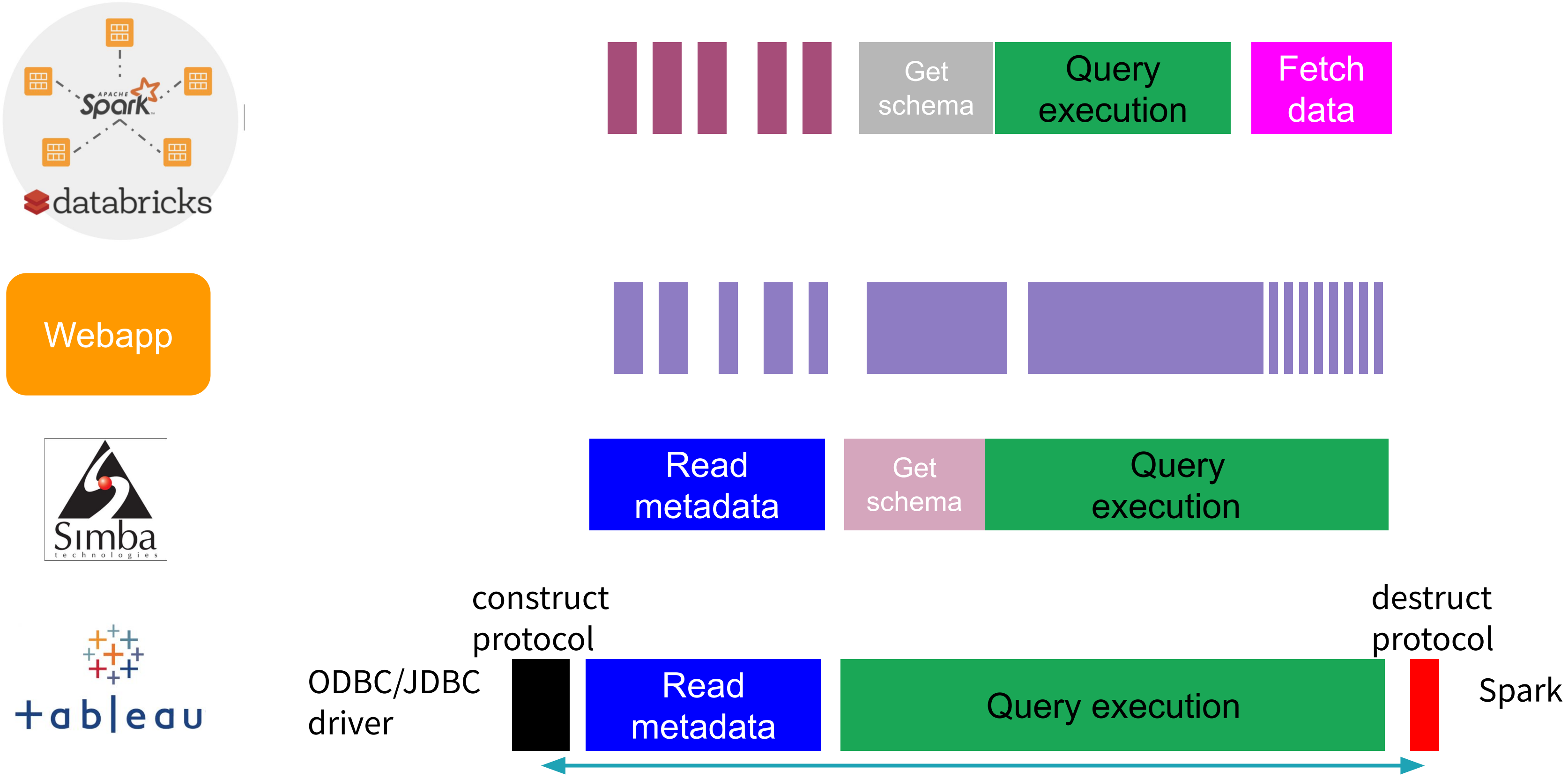
databricks

Control Plane

Tableau on Databricks

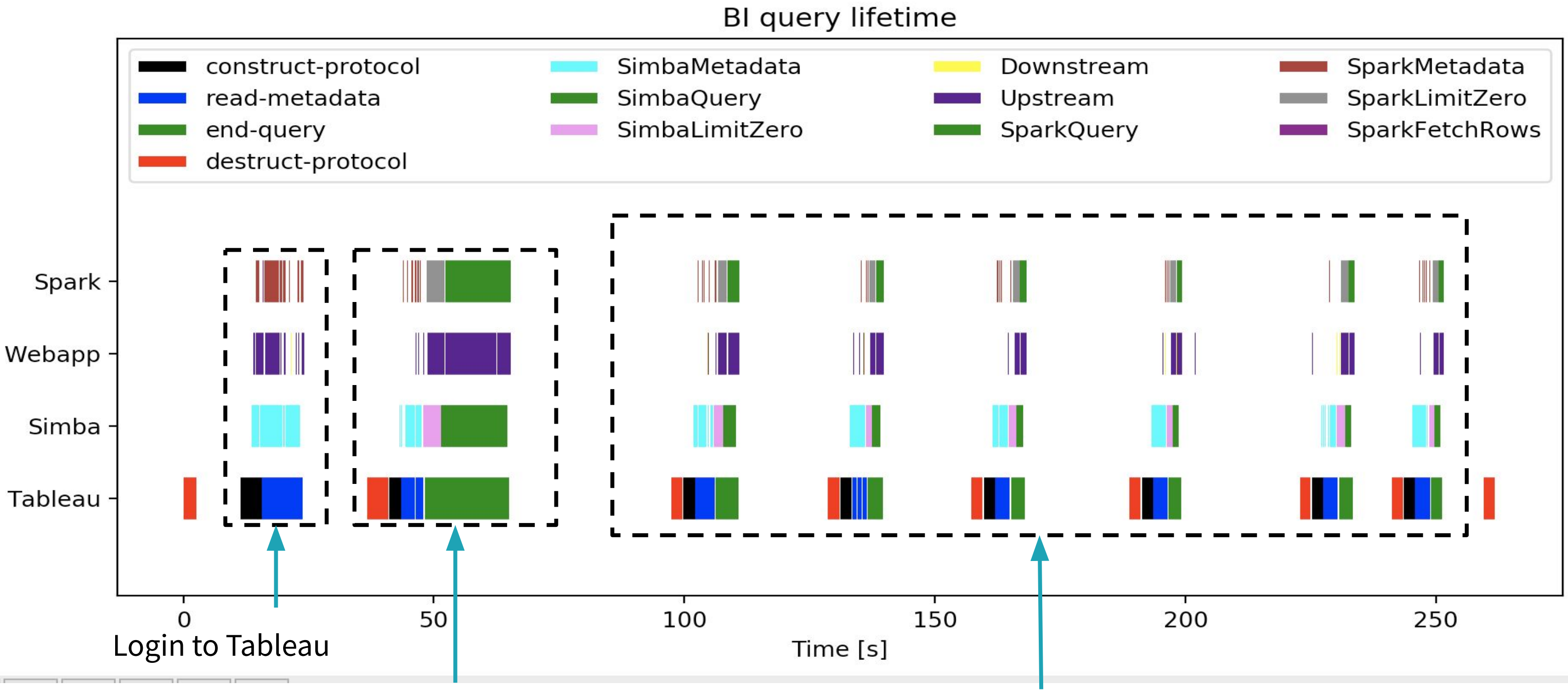


Event Flow



Spark
databricks
Get schema
Query execution
Fetch data
Webapp
Simba technologies
Read metadata
Get schema
Query execution
+ableau
construct protocol
ODBC/JDBC driver
Read metadata
Query execution
destruct protocol
Spark

Behind the Scenes



Login to Tableau

First cold run of a TPC-DS query

Multiple warm runs of a TPC-DS query

The Databricks BI Stack

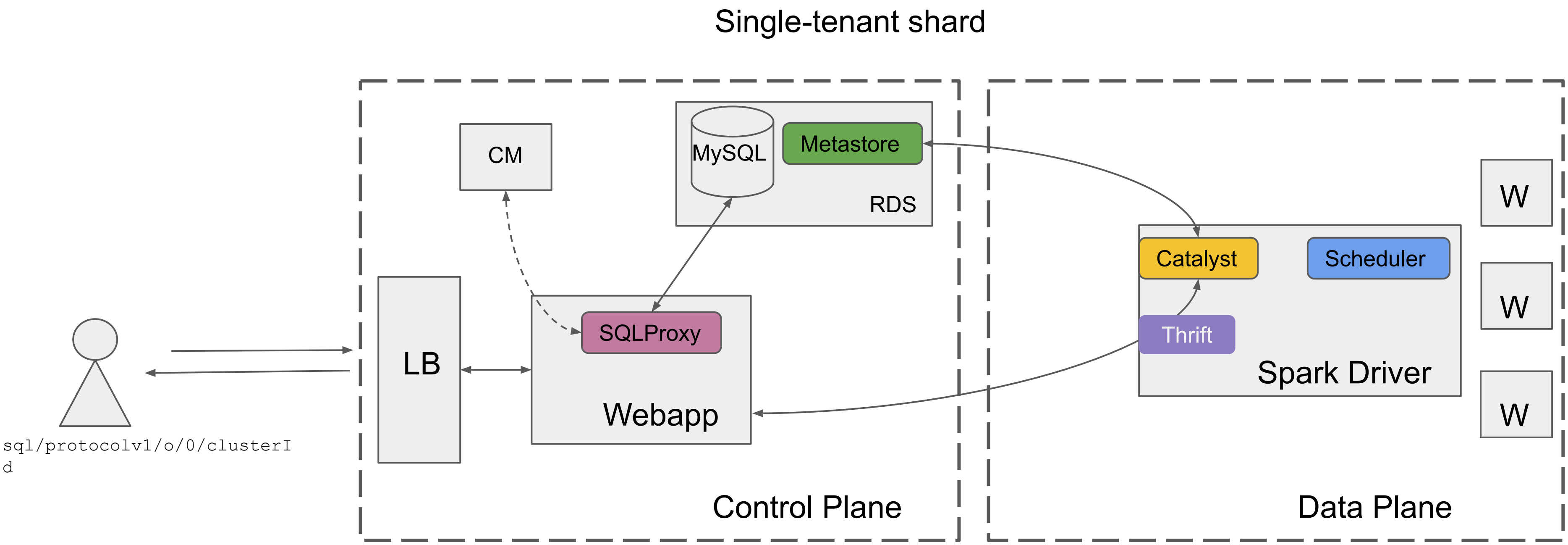
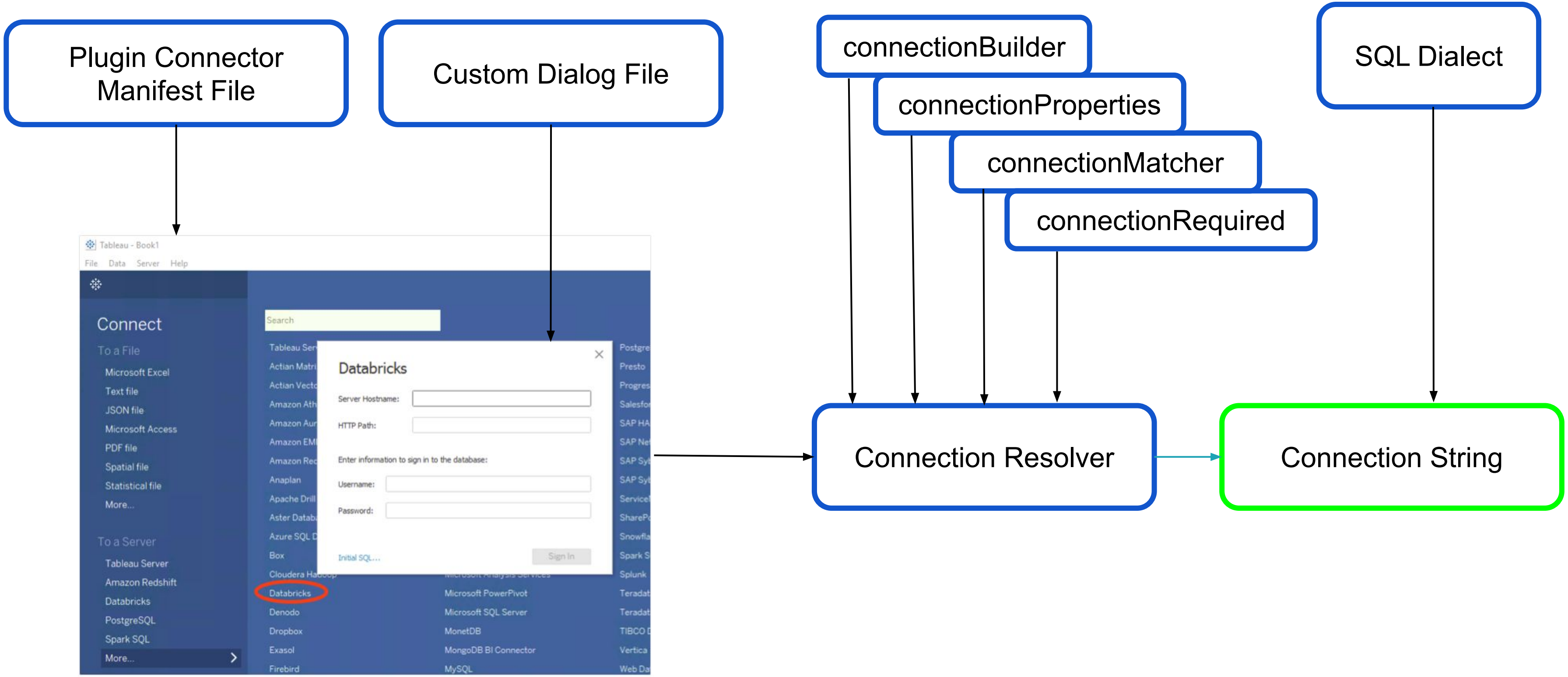


Tableau Connector SDK



Databricks

Server Hostname:

HTTP Path:

Enter information to sign in to the database:

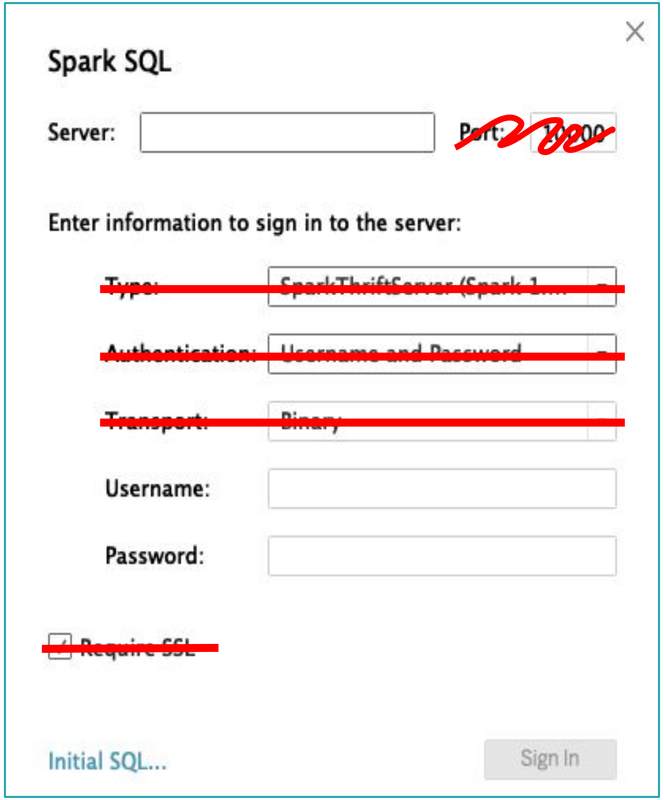
Username:

Password:

Initial SQL...

Sign In

Simplified Connection Dialog



Spark SQL

Server:

~~Port: 10000~~

Enter information to sign in to the server:

~~Type: SparkThriftServer (Spark 1....~~

~~Authentication: Username and Password~~

~~Transport: Binary~~

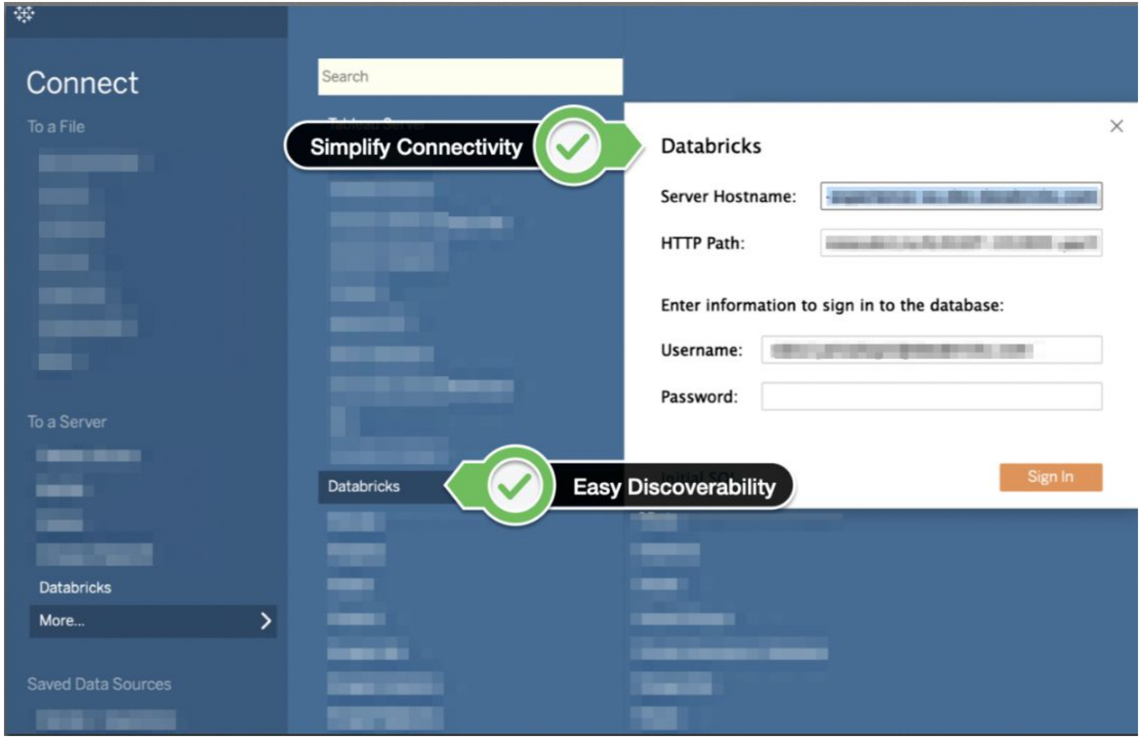
Username:

Password:

~~☑ Require SSL~~

Initial SQL...

Sign In



Connect

To a File

To a Server

Databricks

More...

Saved Data Sources

Search

Databricks

Server Hostname: [redacted]

HTTP Path: [redacted]

Enter information to sign in to the database:

Username: [redacted]

Password:

Sign In

Simplify Connectivity

Easy Discoverability

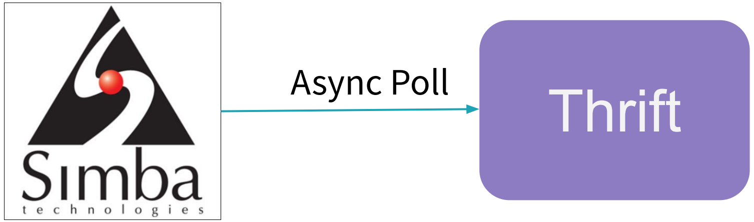
Controlling the Dialect

```
<function group='string' name='SPLIT' return-type='str'>
<formula> CASE
  WHEN (%1 IS NULL) THEN
    CAST(NULL AS STRING)
  WHEN NOT (%3 IS NULL) THEN
    COALESCE(
     (CASE WHEN %3 &gt; 0 THEN SPLIT(%1, '%2')[%3-1]
           ELSE SPLIT(
              REVERSE(%1),
              REVERSE('%2'))[ABS(%3)-1] END), '')
  ELSE NULL END
</formula>
<argument type='str' />
<argument type='localstr' />
<argument type='localint' />
</function>
```

- ✓ Missing operators: IN_SET, ATAN, CHAR, RSPLIT
- ✓ Hive dialect wraps DATENAME in COALESCE operators
- ✓ Strategy to determine if two values are distinct
- ✓ The datasource supports booleans natively
- ✓ CASE-WHEN statements should be of boolean type

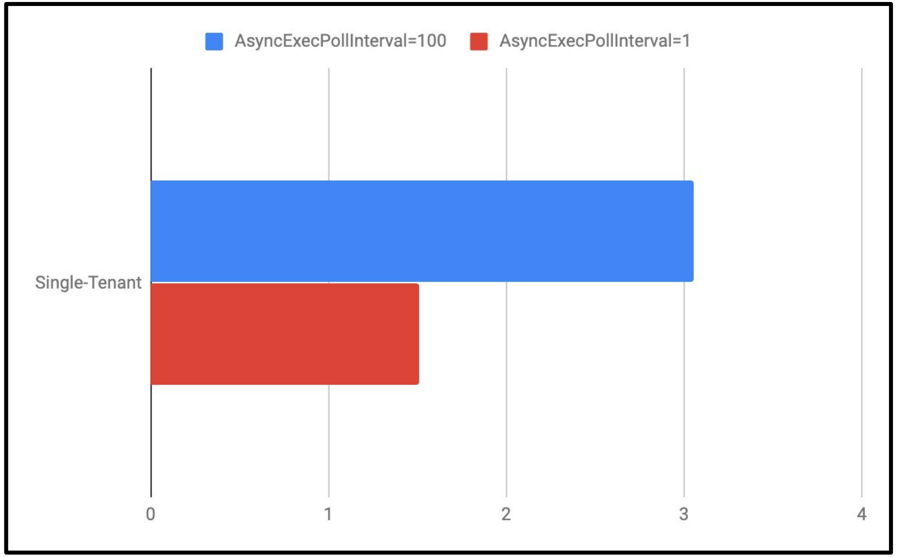
Achieved 100% compliance with TDVT standard testing

Polling for Query Results



Sends async polls for result

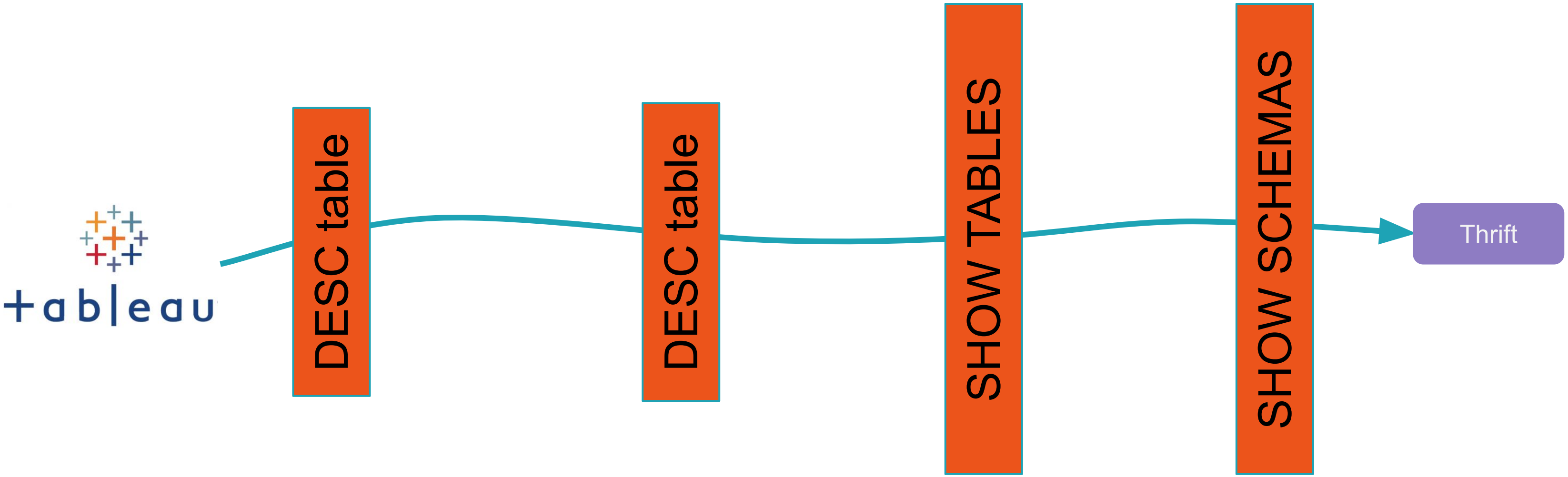
Blocks for 5 sec if query is not finished



First poll after 100 ms causing high-latency for short-lived metadata queries

Cuts in half latency by lowering the polling interval

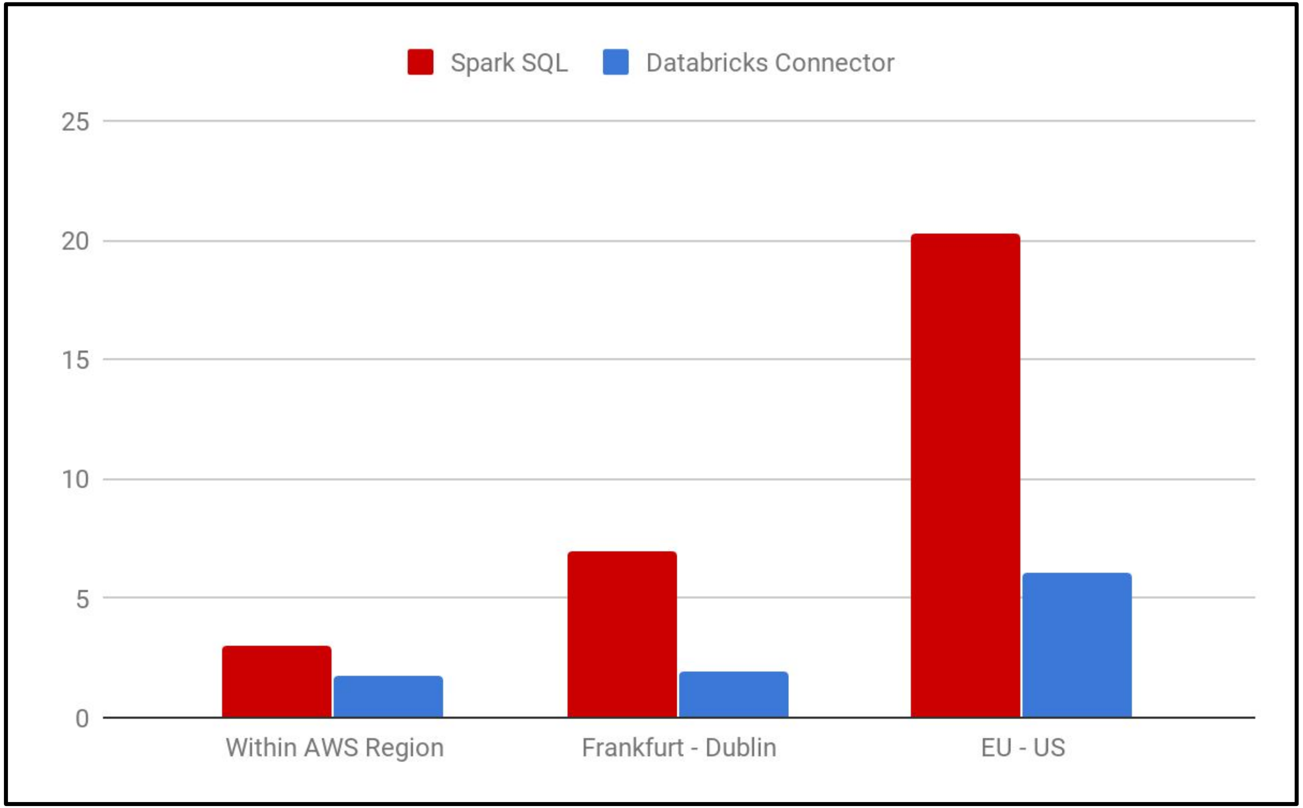
Metadata Queries are Expensive



Each query triggers 6 round-trips from Tableau to Thrift

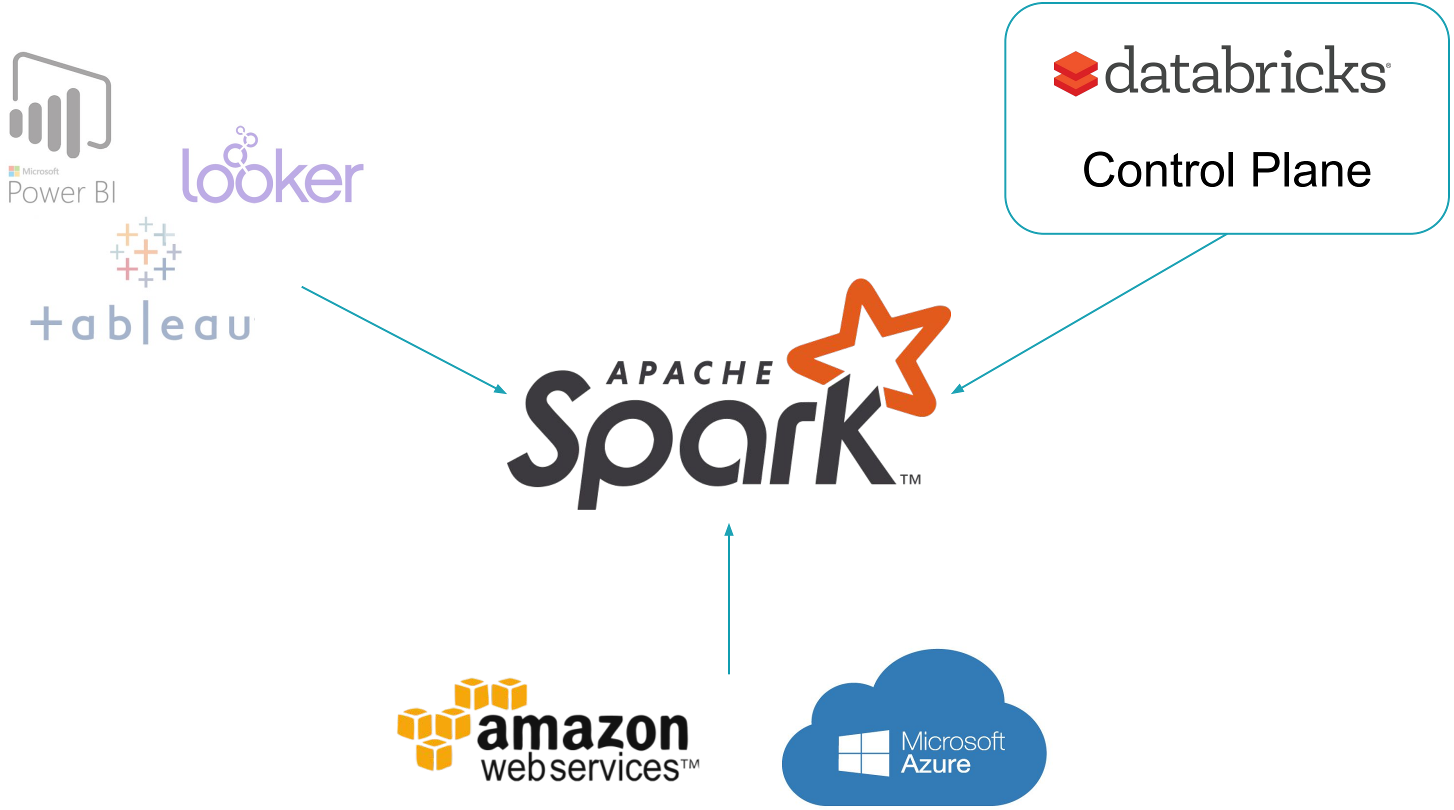
We optimize the sequence of metadata queries by retrieving all needed metadata in one go

Data Source Connection Latency



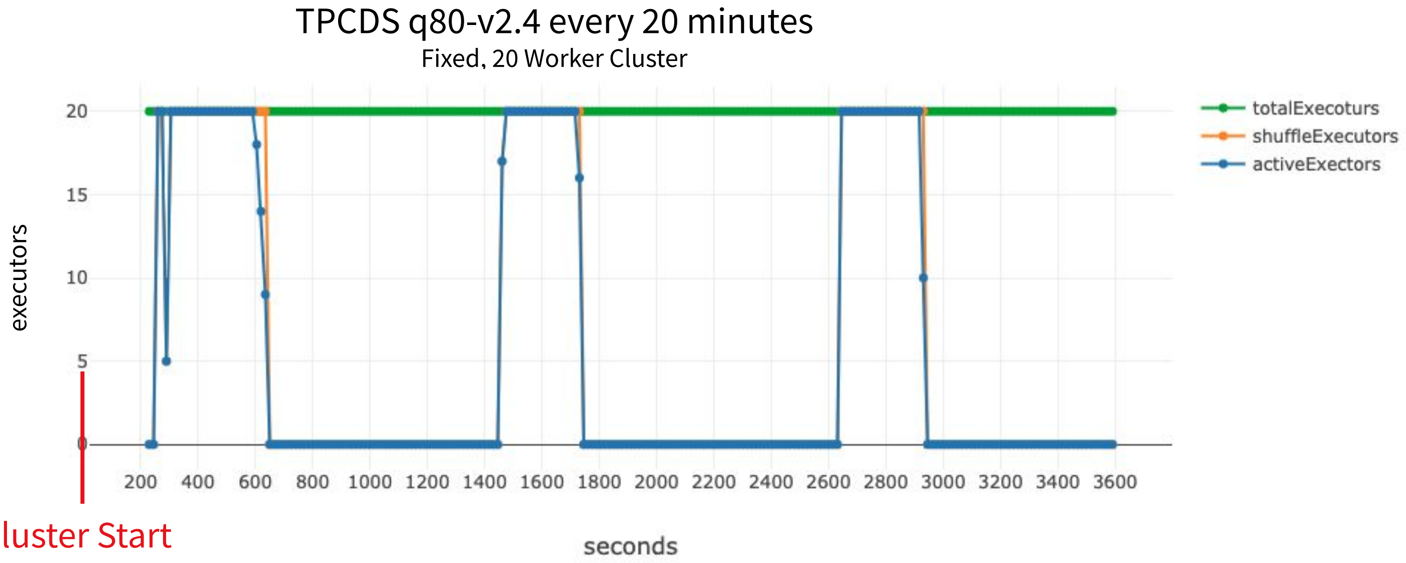
New connector delivers 1.7-5x lower latency

Resource Management



Control Plane

Execution Time Optimized



Cluster Start

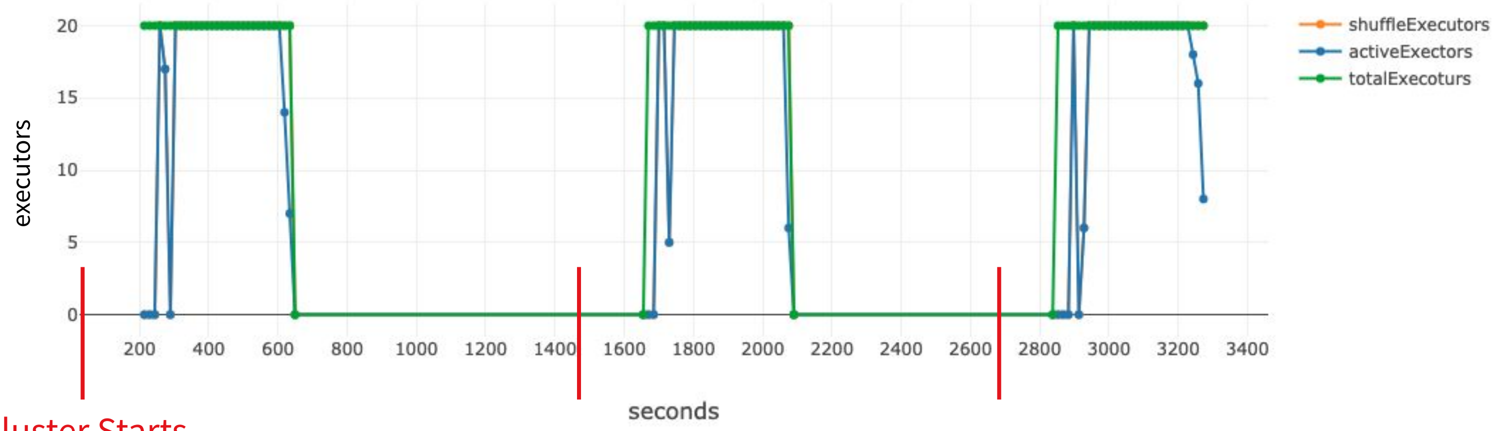
Execution Time: 315 seconds
Cost: $14.95 per hour

databricks

Cost Optimized

TPCDS q80-v2.4 every 20 minutes

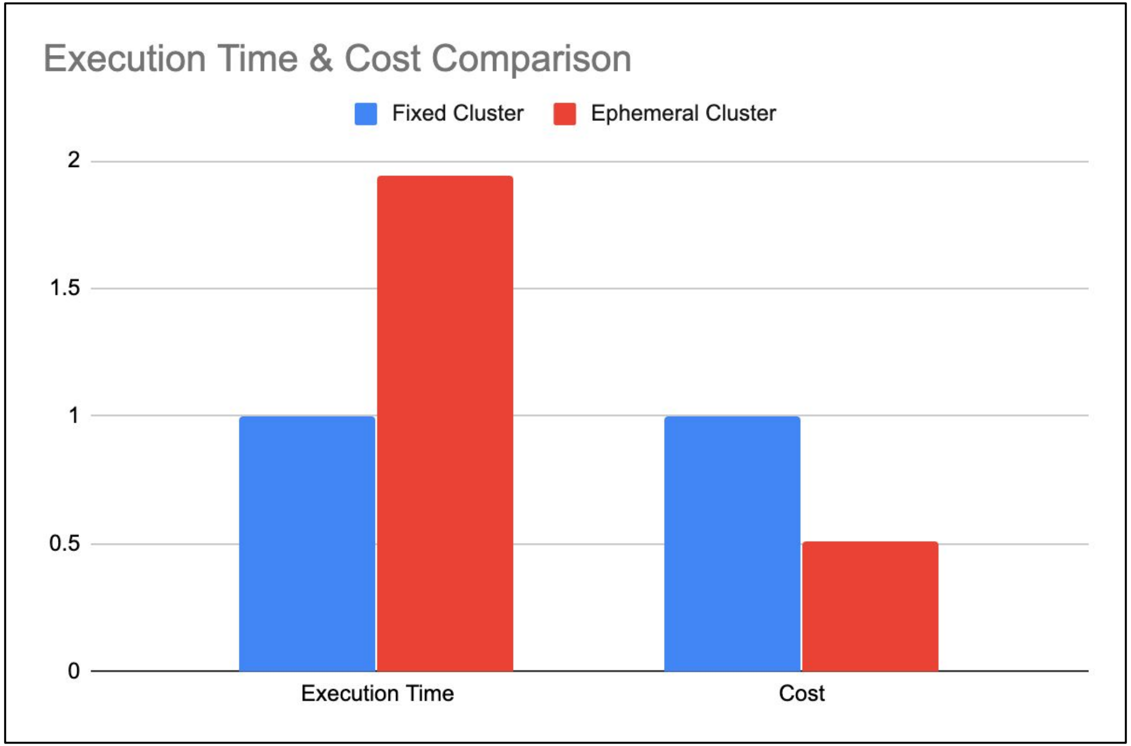
Auto-terminating, 20 Worker Cluster



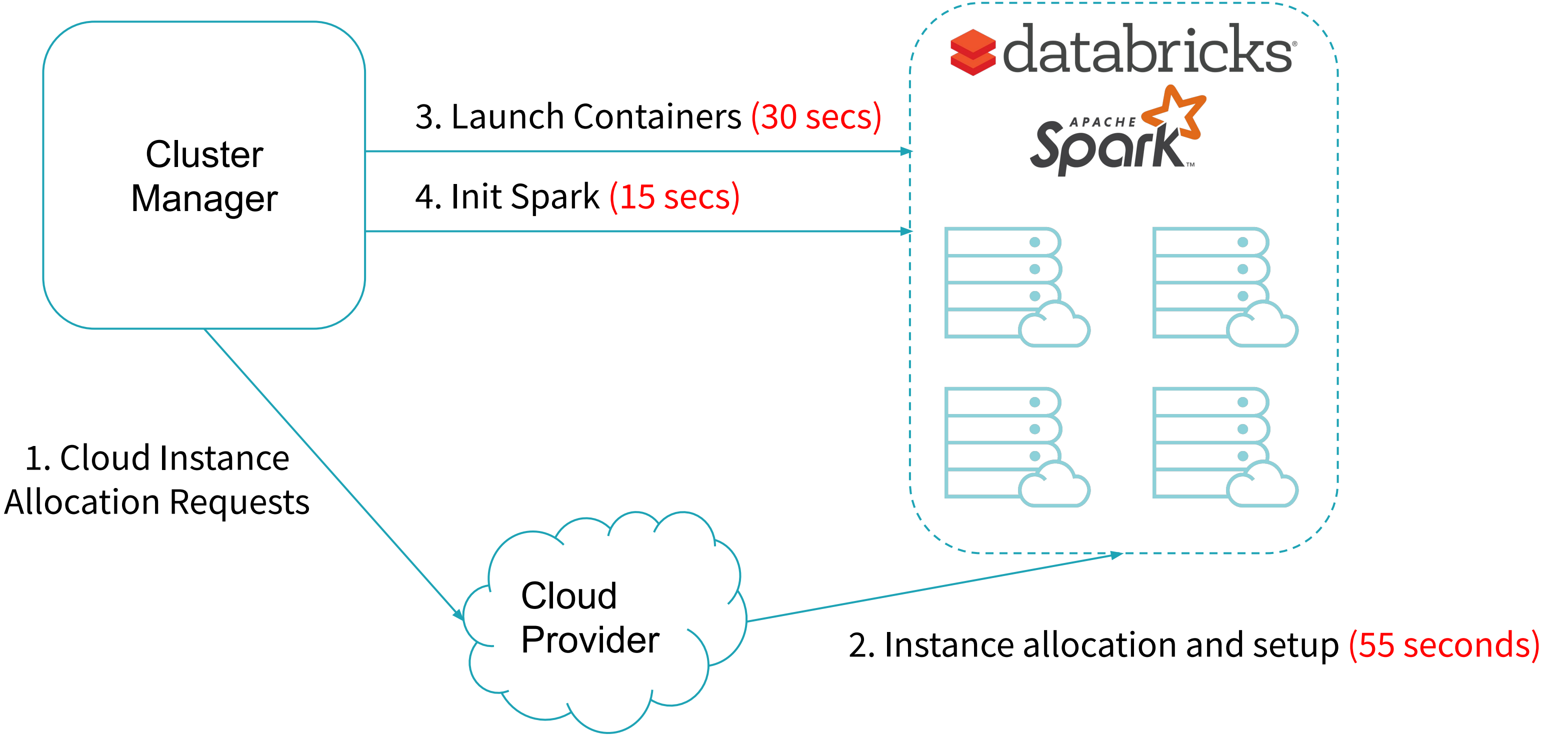
Cluster Starts

Execution Time: 613 seconds
Cost: $7.66 per hour

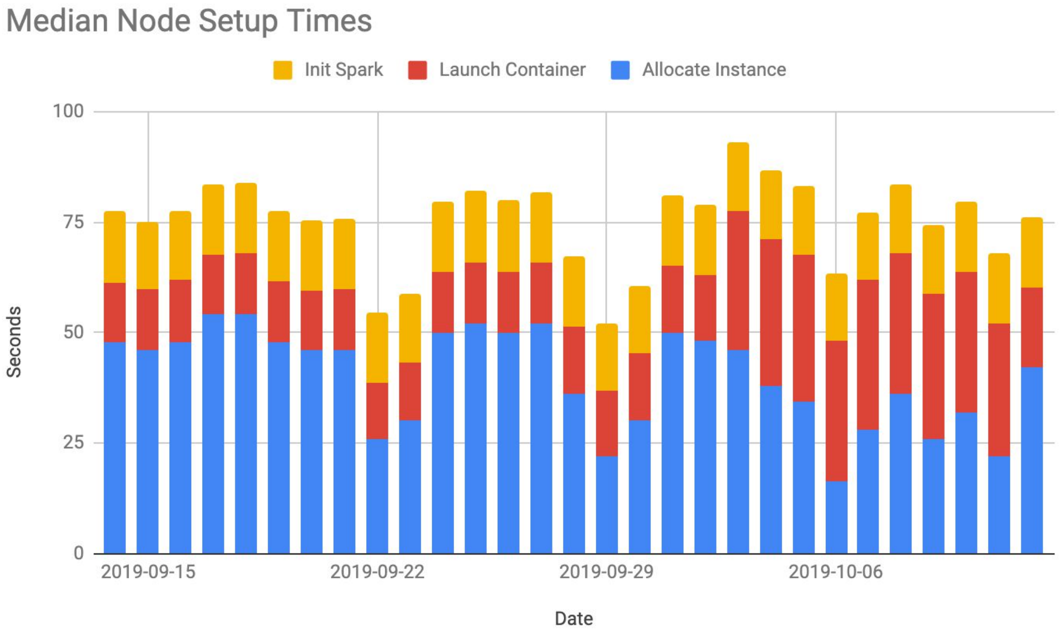
Cost vs Execution Time



Cluster Start Path


Cluster Manager
3. Launch Containers (30 secs)
4. Init Spark (15 secs)
databricks
APACHE Spark
1. Cloud Instance Allocation Requests
Cloud Provider
2. Instance allocation and setup (55 seconds)

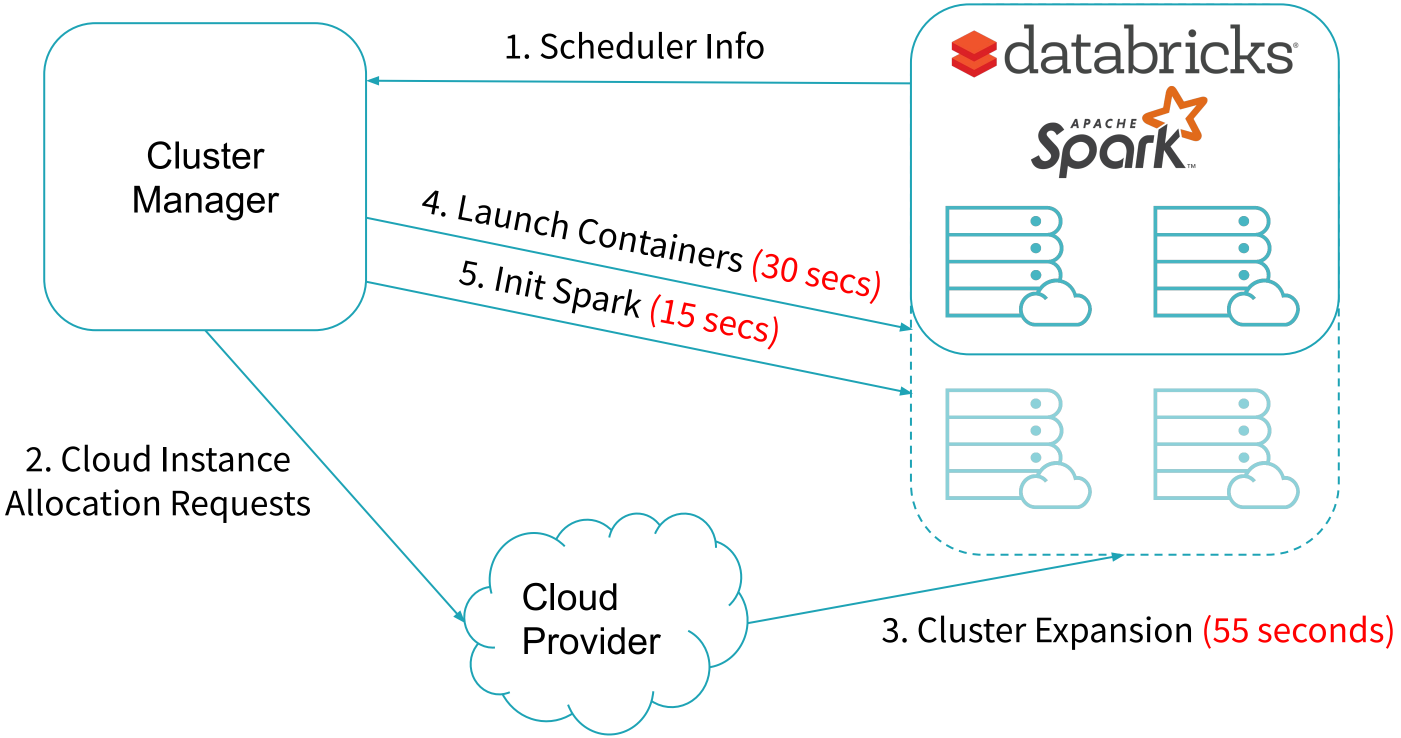
Node Start Time



- Up to 55 seconds to acquire an instance from the cloud provider and initialize it.
- Up to 30 seconds to launch a container (includes downloading DBR)
- ~15 seconds to start master/worker
- Median for cluster starts is 2 mins and 22 seconds.

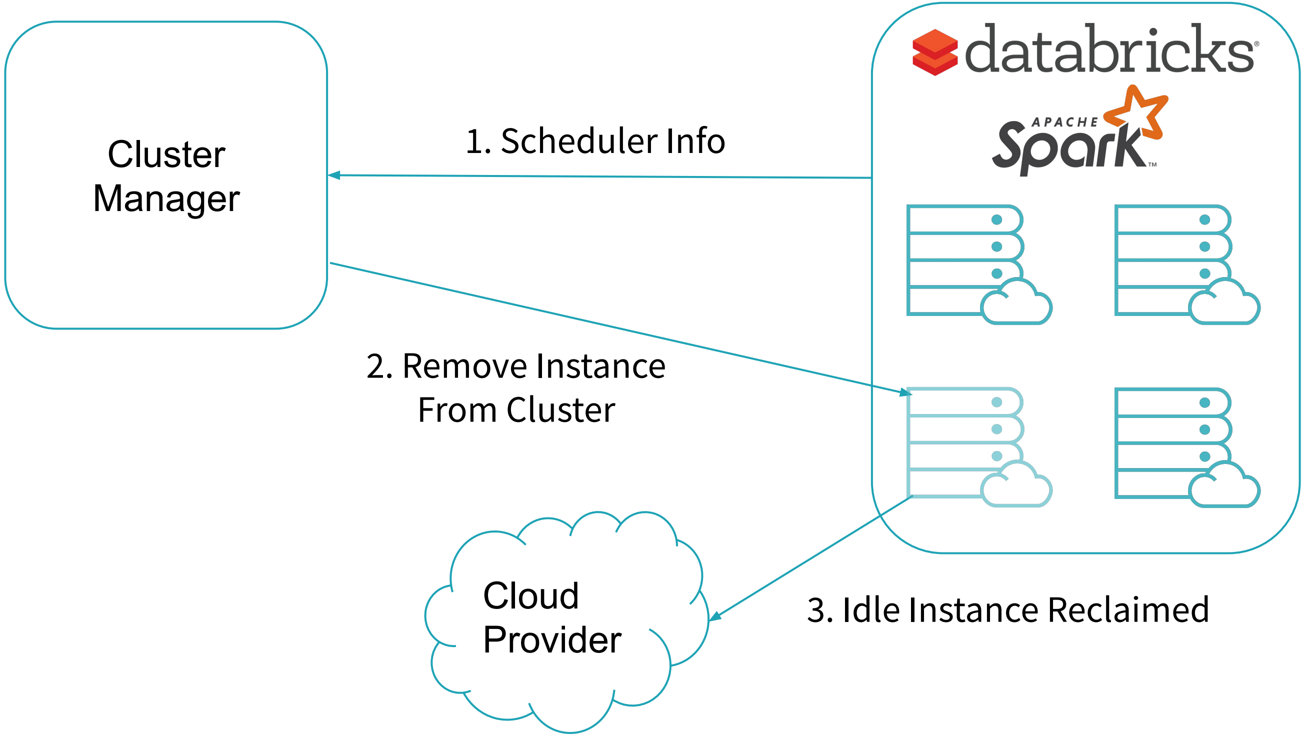
databricks

Up-scaling



1. Scheduler Info
Cluster Manager
databricks
APACHE Spark
4. Launch Containers (30 secs)
5. Init Spark (15 secs)
2. Cloud Instance Allocation Requests
Cloud Provider
3. Cluster Expansion (55 seconds)

Down-scaling



Cluster Manager

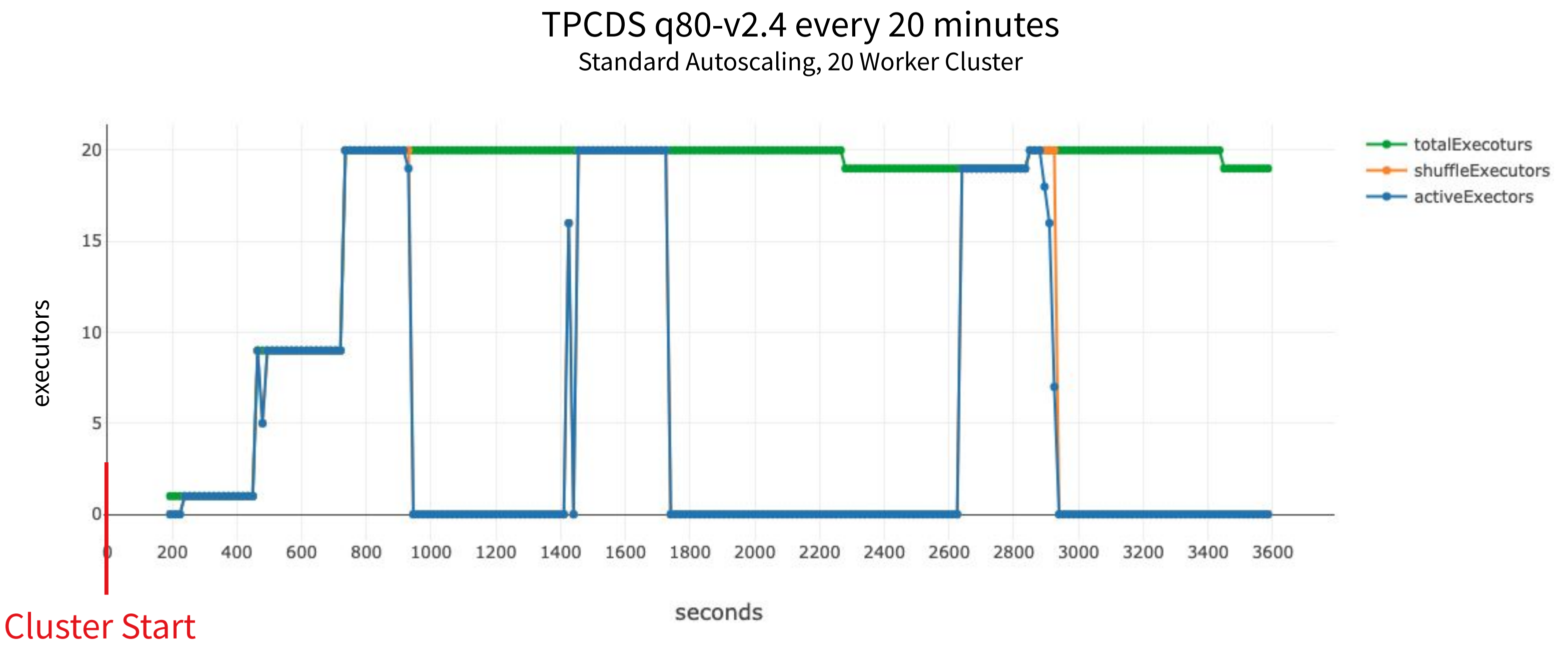
1. Scheduler Info

2. Remove Instance From Cluster

3. Idle Instance Reclaimed

Cloud Provider

Basic Autoscaling

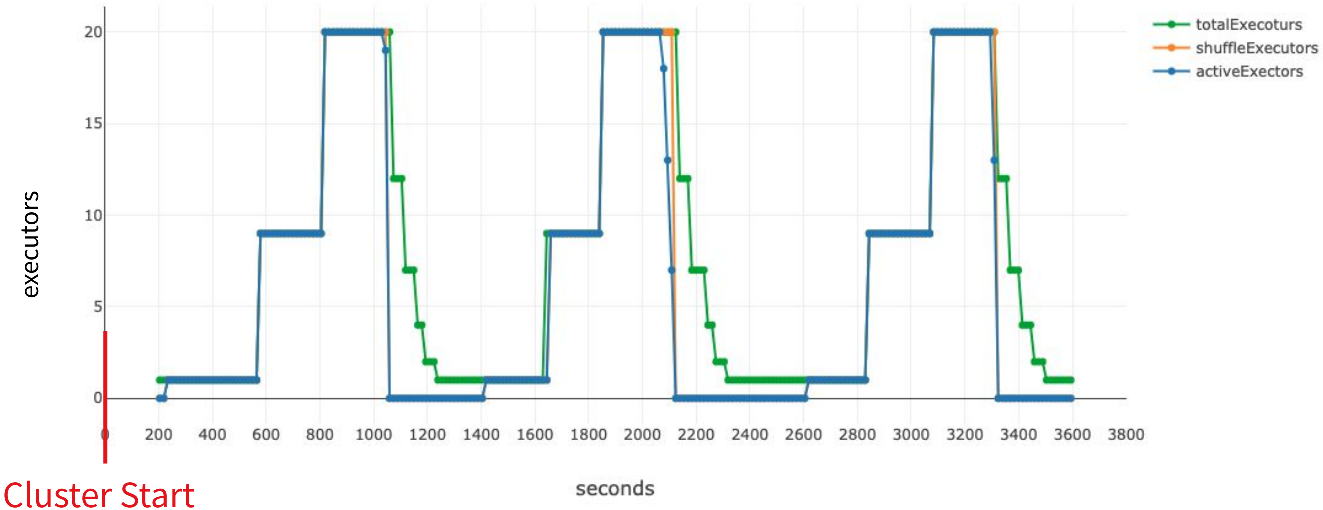


Execution Time: 318 seconds
Cost: $14.24 per hour

Optimized Autoscaling

TPCDS q80-v2.4 every 20 minutes

Optimized Autoscaling, 20 Worker Cluster



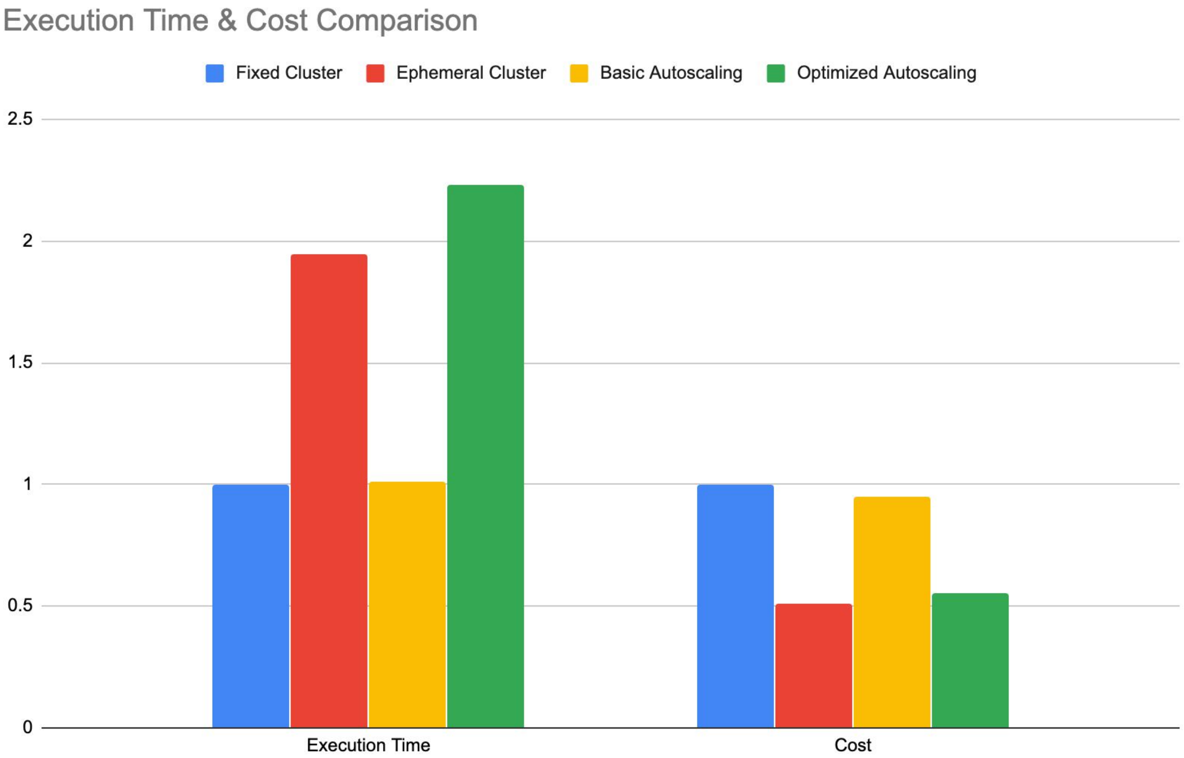
Cluster Start

“spark.databricks.aggressiveWindowDownS” -> “40”

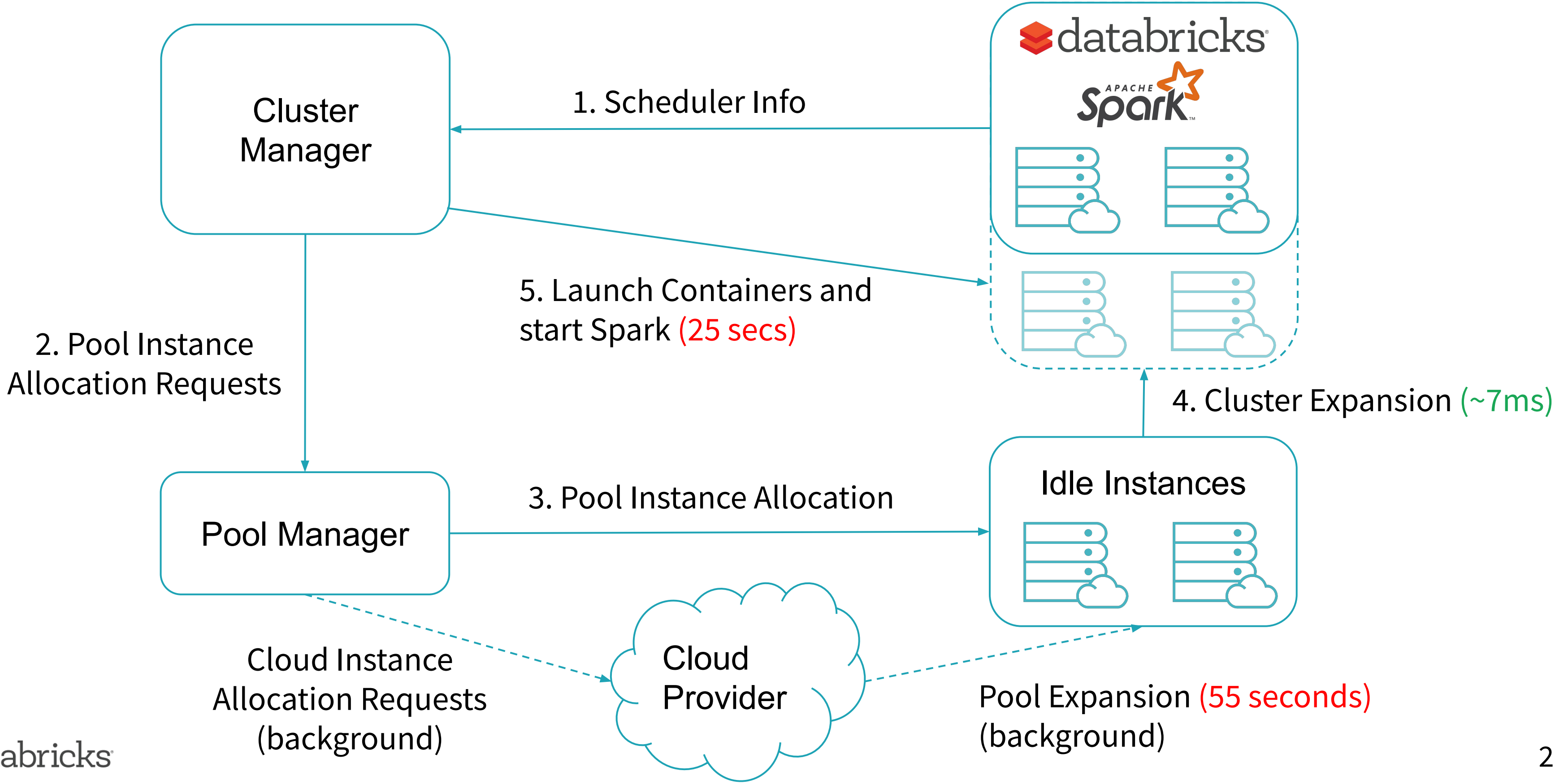
Execution Time: 703 seconds
Cost: $7.27 per hour

databricks

Cost vs Execution Time

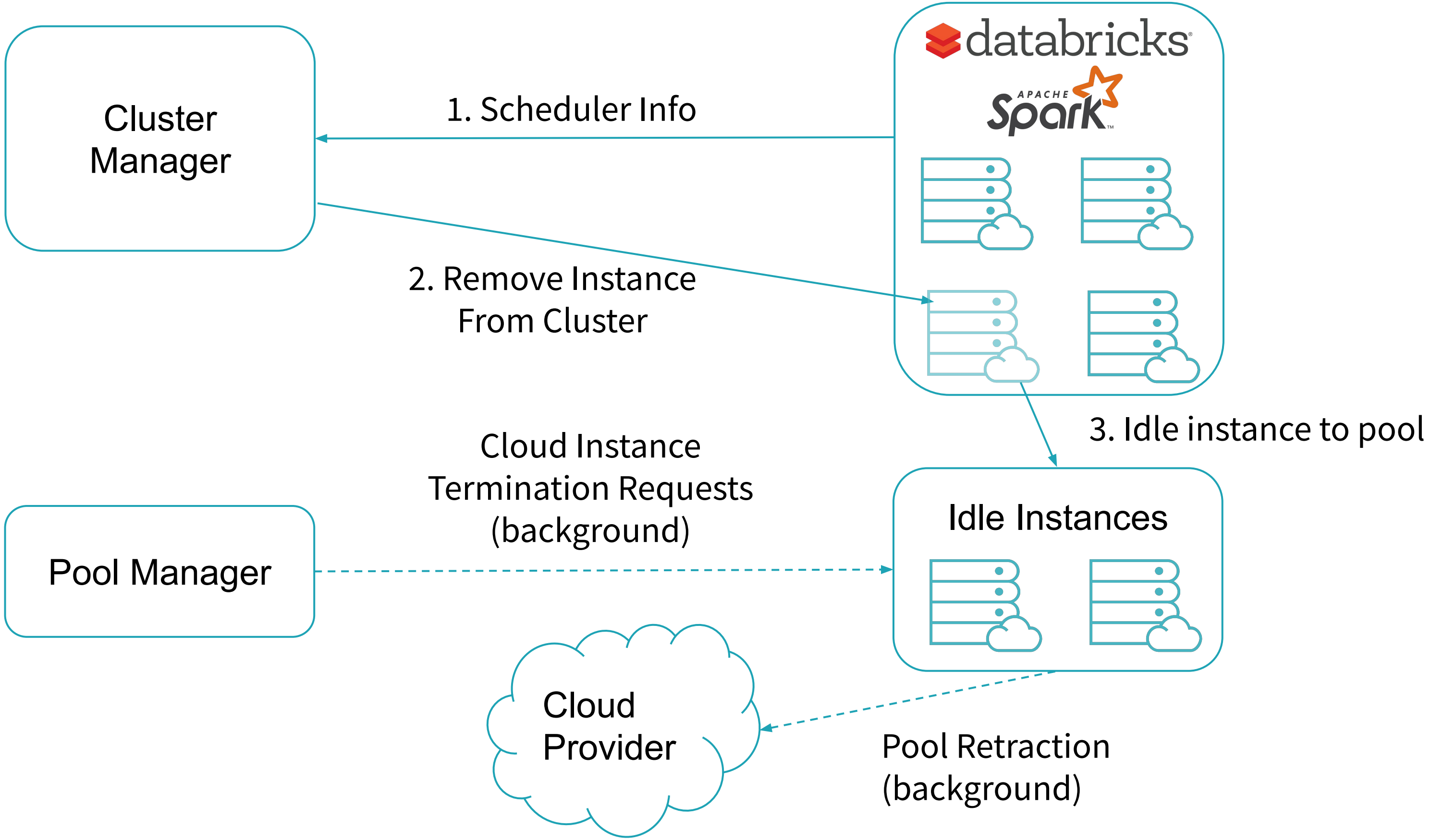


Up-scaling with Instance Pools

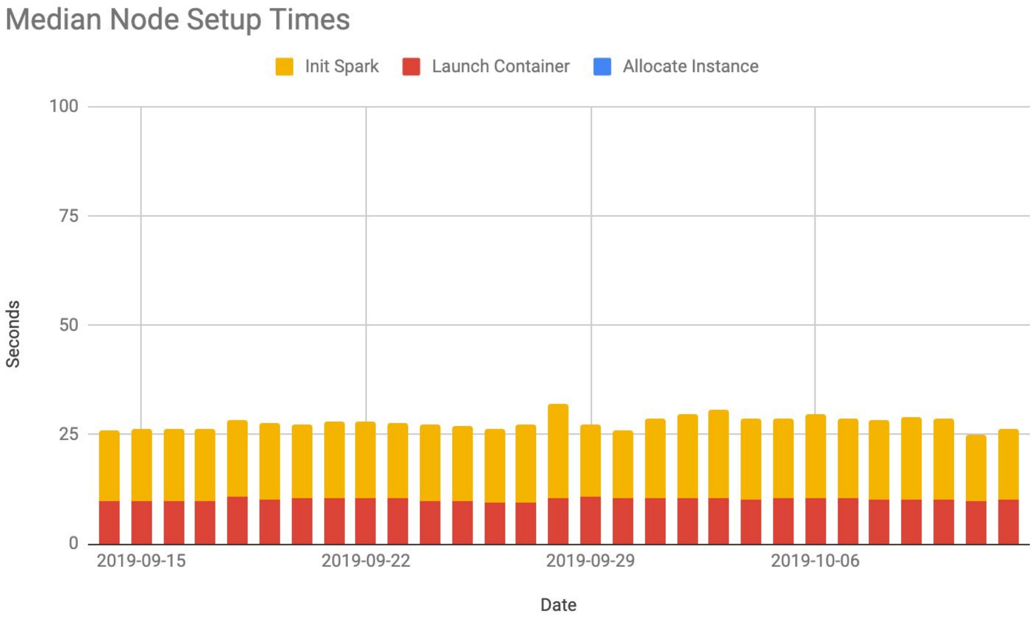


databricks
APACHE Spark
Cluster Manager
1. Scheduler Info
5. Launch Containers and start Spark (25 secs)
2. Pool Instance Allocation Requests
4. Cluster Expansion (~7ms)
Pool Manager
3. Pool Instance Allocation
Idle Instances
Cloud Instance Allocation Requests (background)
Cloud Provider
Pool Expansion (55 seconds) (background)

Down-scaling with Instance Pools



Node Start Time with Pools

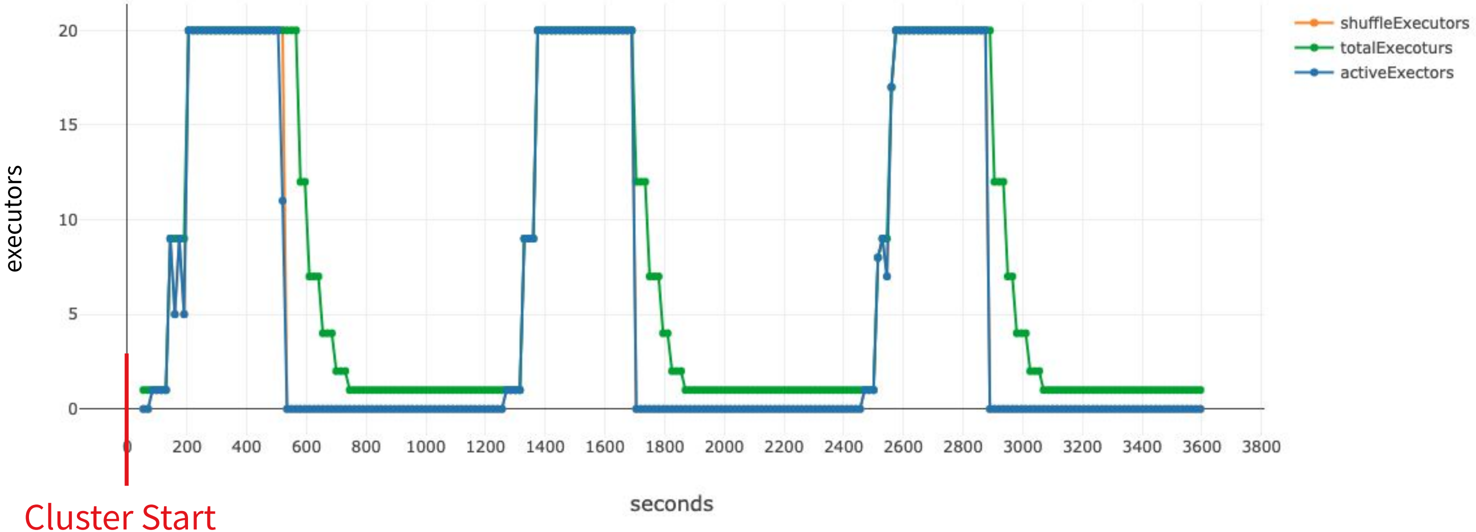


- Instance allocation is mostly gone (milliseconds).
- ~10 second container start (DBR already downloaded by the pool).
- ~15 seconds to start master/worker
- Median cluster starts is 50 seconds. 2.84x faster.

Optimized Autoscaling with Fixed Pool

TPCDS q80-v2.4 every 20 minutes

Optimized Autoscaling w/ Fixed Pool, 20 Worker Cluster

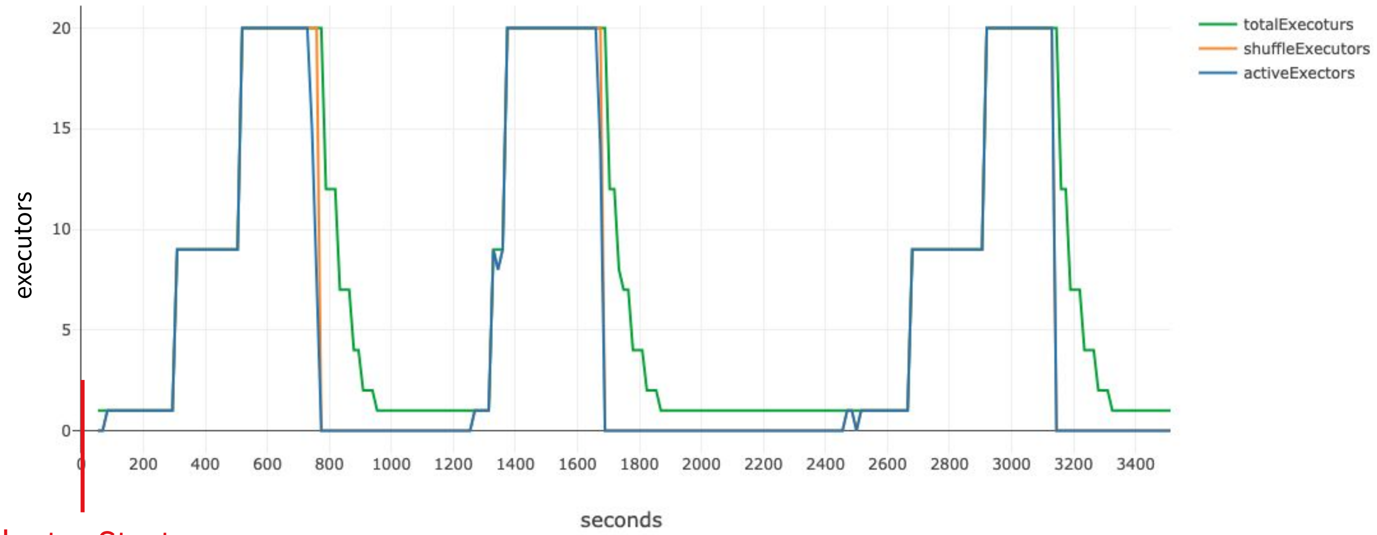


Cluster Start

“spark.databricks.aggressiveWindowDownS” -> “40”

Execution Time: 427 seconds
Cost: $10.05 per hour

Optimized Autoscaling with Dynamic Pool



TPCDS q80-v2.4 every 20 minutes

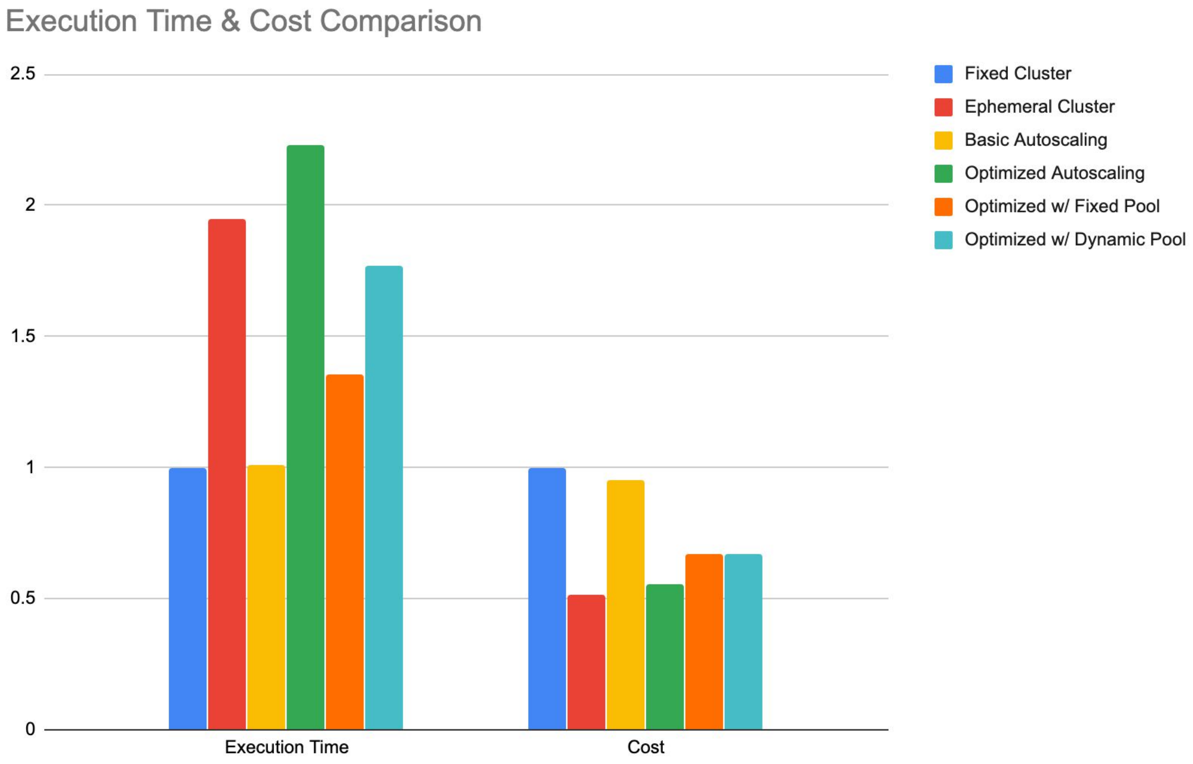
Optimized Autoscaling w/ Dynamic Pool, 20 Worker Cluster

Cluster Start

“spark.databricks.aggressiveWindowDownS” -> “40”
Pool has 5 min idle with 10 minute idle timeout.

Execution Time: 557 seconds
Cost: $10.05 per hour

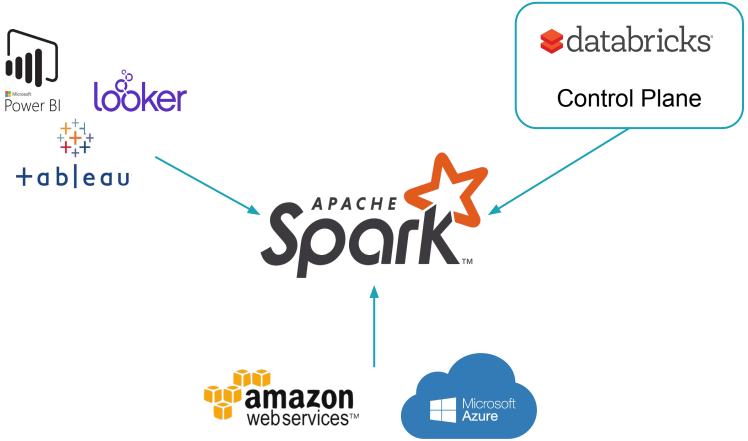
Cost vs Execution Time



Conclusions

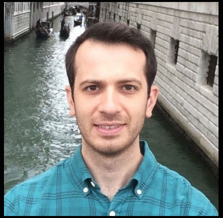
Spark is at the core but there's a lot more around it to bring it into production

1. Latency-aware integration
2. Efficient resource usage
3. Fast provisioning of resources



Thanks!

Chris Stevens - linkedin.com/in/chriscstevens



Bogdan Ghit - linkedin.com/in/bogdanghit

databricks

Comparison

	Fixed Cluster	Ephemeral Cluster	Standard Autoscaling	Optimized Autoscaling	Fixed Pool	Dynamic Pool
Start Latency	218s	-4s	+499s	+434s	-27s	+280s
Repeat Latency	15s	+200s	+164s	+411s	+71s	+251s
Start Duration	407s	-8s	+323s	+278s	+75s	+283s
Repeat Duration	315s	+84s	+3s	+352s	+123s	+186s
EC2 Cost	$6.55	-$3.19	-$1.04	-$2.44	-$0	-$0.38
DBU Cost	$8.40	-$4.10	-$1.33	-$3.13	$-3.83	-$3.35