

# Accelerate Performance at Scale: Best Practices for Trino with Amazon S3

**Dai Ozaki**

Cloud Support Engineer, AWS



# Dai Ozaki

Cloud Support Engineer, AWS Support Engineering, Amazon Web Services



- Responsible for solving the most complex technical issues related to AWS big data services such as Amazon Athena, AWS Glue, and Amazon EMR
- Athena subject matter expert



# Agenda

- Why Amazon S3 with Trino?
- Common challenges in scaling Trino workload
- Best practices to scale workload with Amazon S3

# Why Amazon S3 with Trino?

# Amazon S3



Durable

Highly available

Scalable

Cost effective

Secure

# Use Case of Amazon S3 with Trino

- Trino is a powerful tool to query data from data lakes
- Amazon S3 is the best place to build a data lake



Amazon S3

**Cost effective** to store data

Unmatched **durability, availability,** and **scalability**

Best **security, compliance,** and **audit** capabilities

# Common challenges in scaling Trino workload

# Common challenges in scaling Trino workload



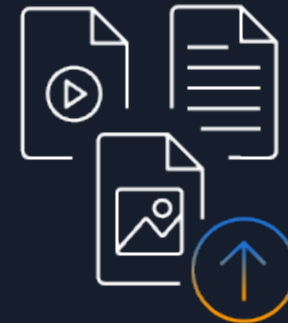
Huge data scan



HTTP Slow Down error

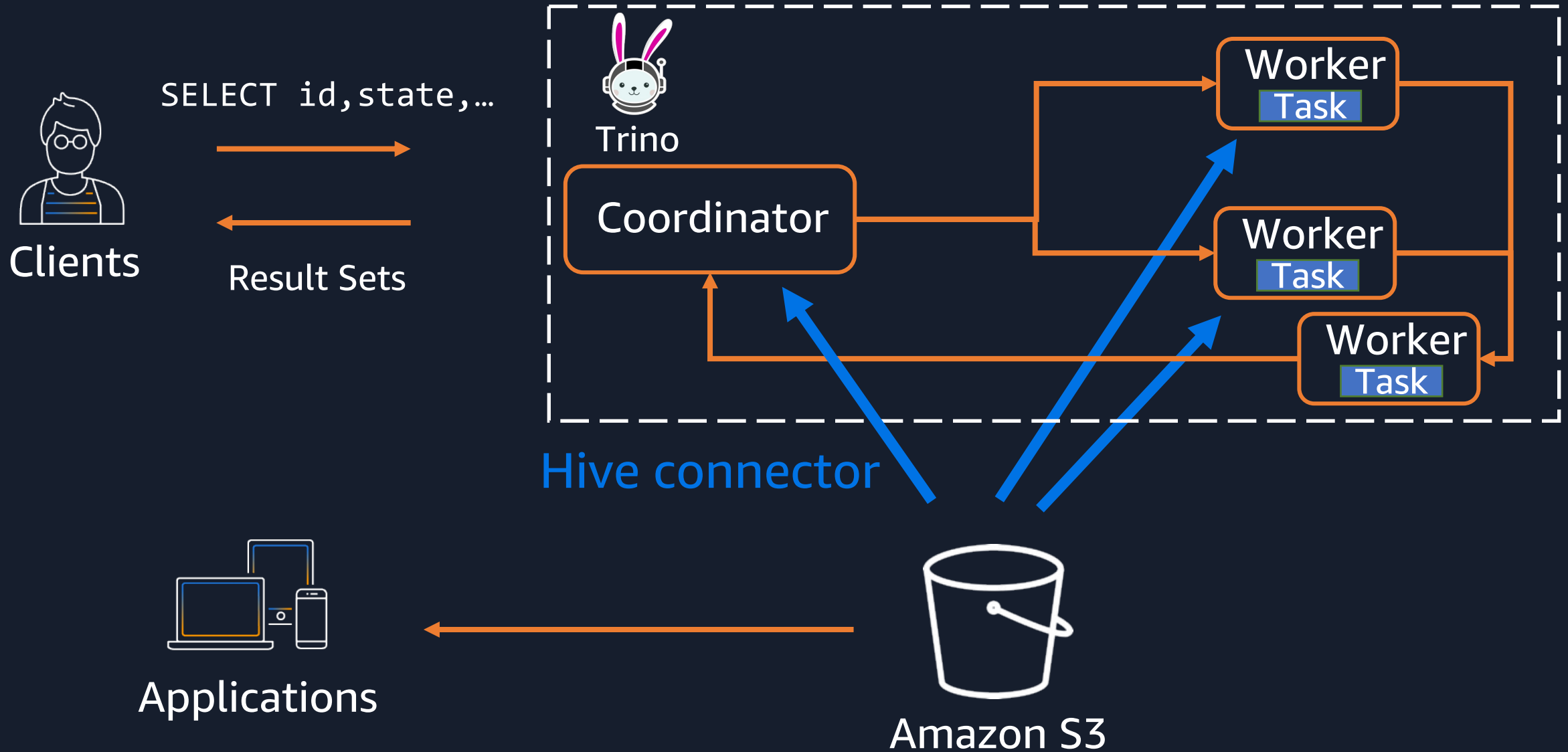


Many small files issue

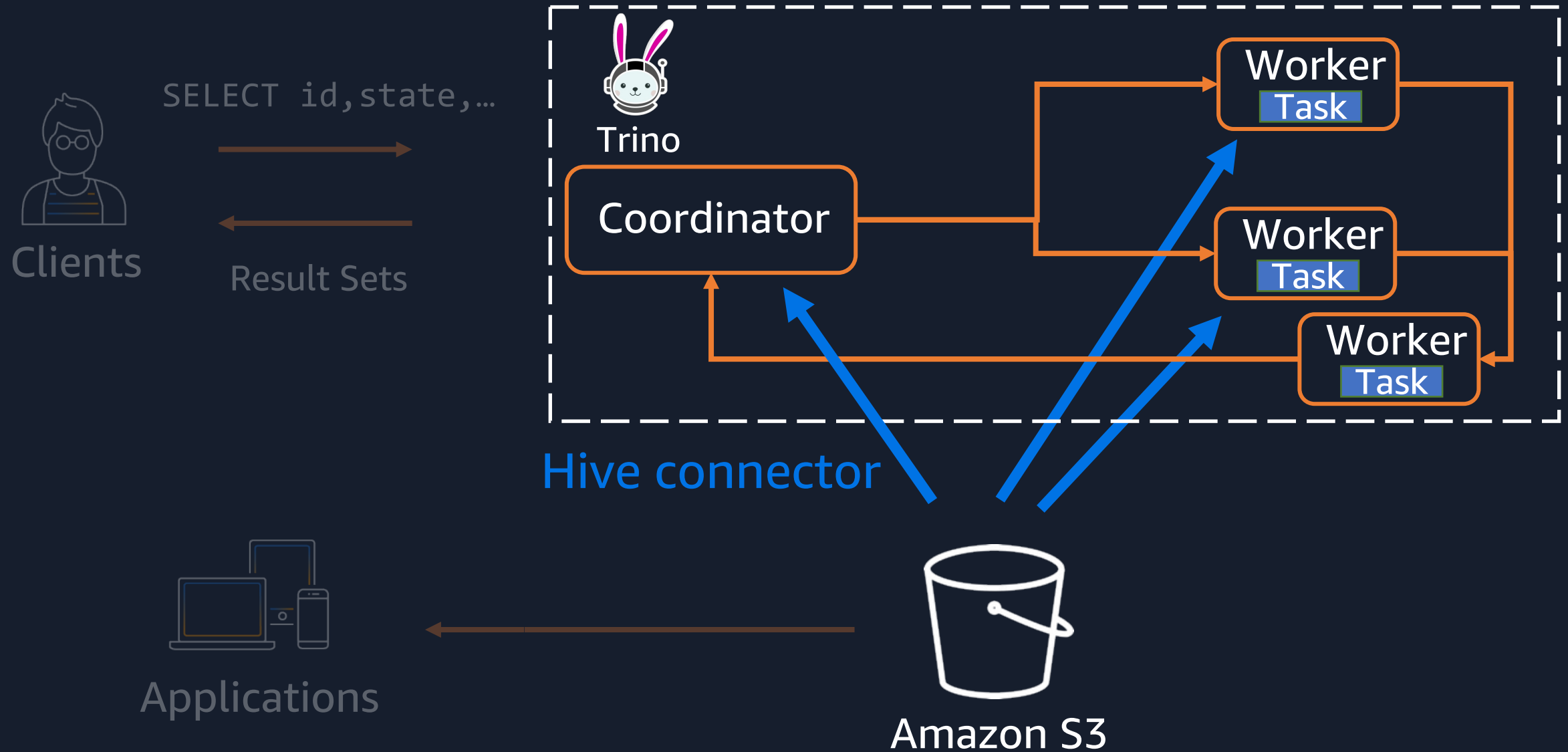


Unneeded data is stored

# Common Architecture



# Challenge 1: Huge data scan



# Challenge 1: Huge data scan



Slower query

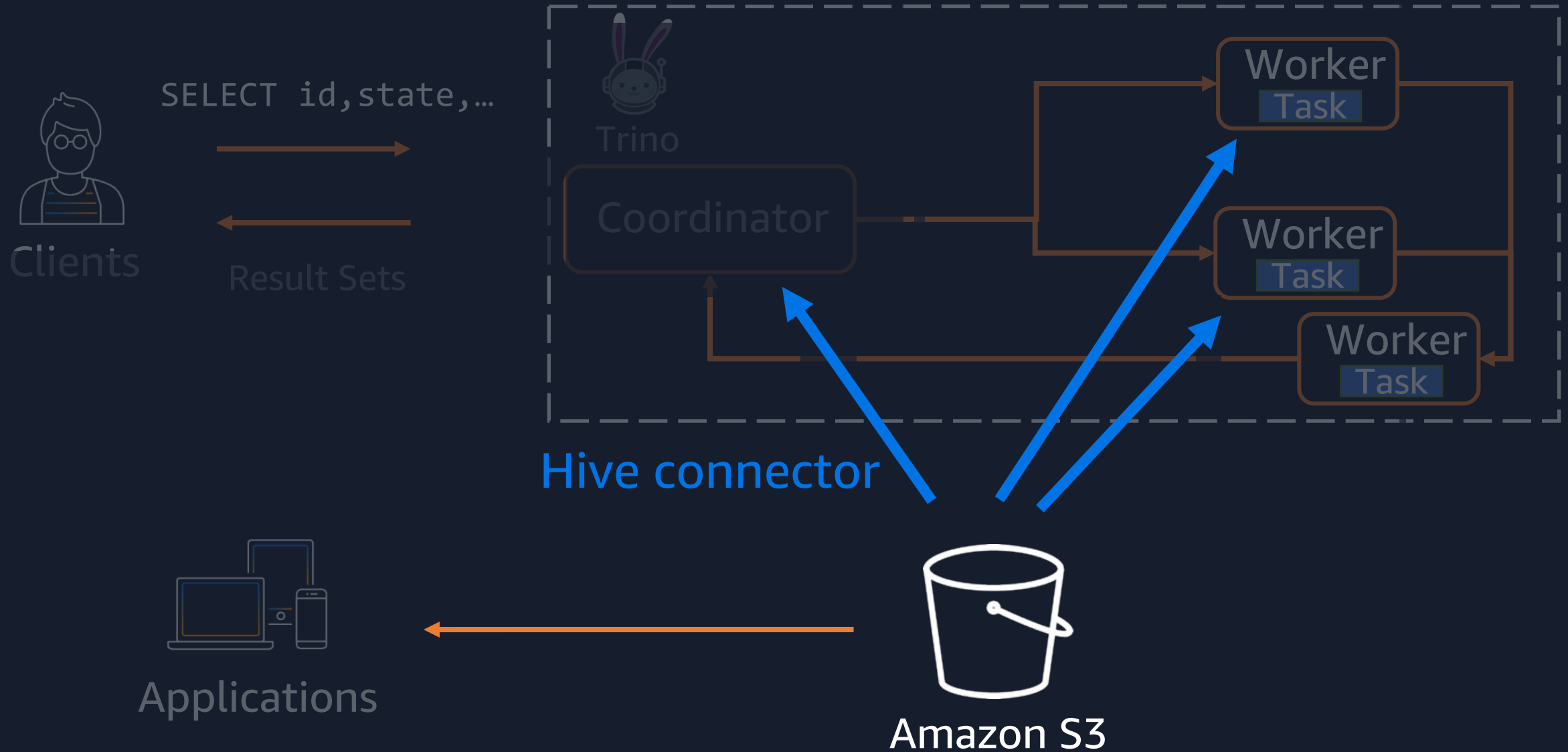


Worker OOM



High cost

# Challenge 2: HTTP Slow Down error



# Challenge 2: HTTP Slow Down error

- Amazon S3 performance is defined per **prefix**

s3://bucket/daily-uploads/20240613/drive-data.csv  
Prefix

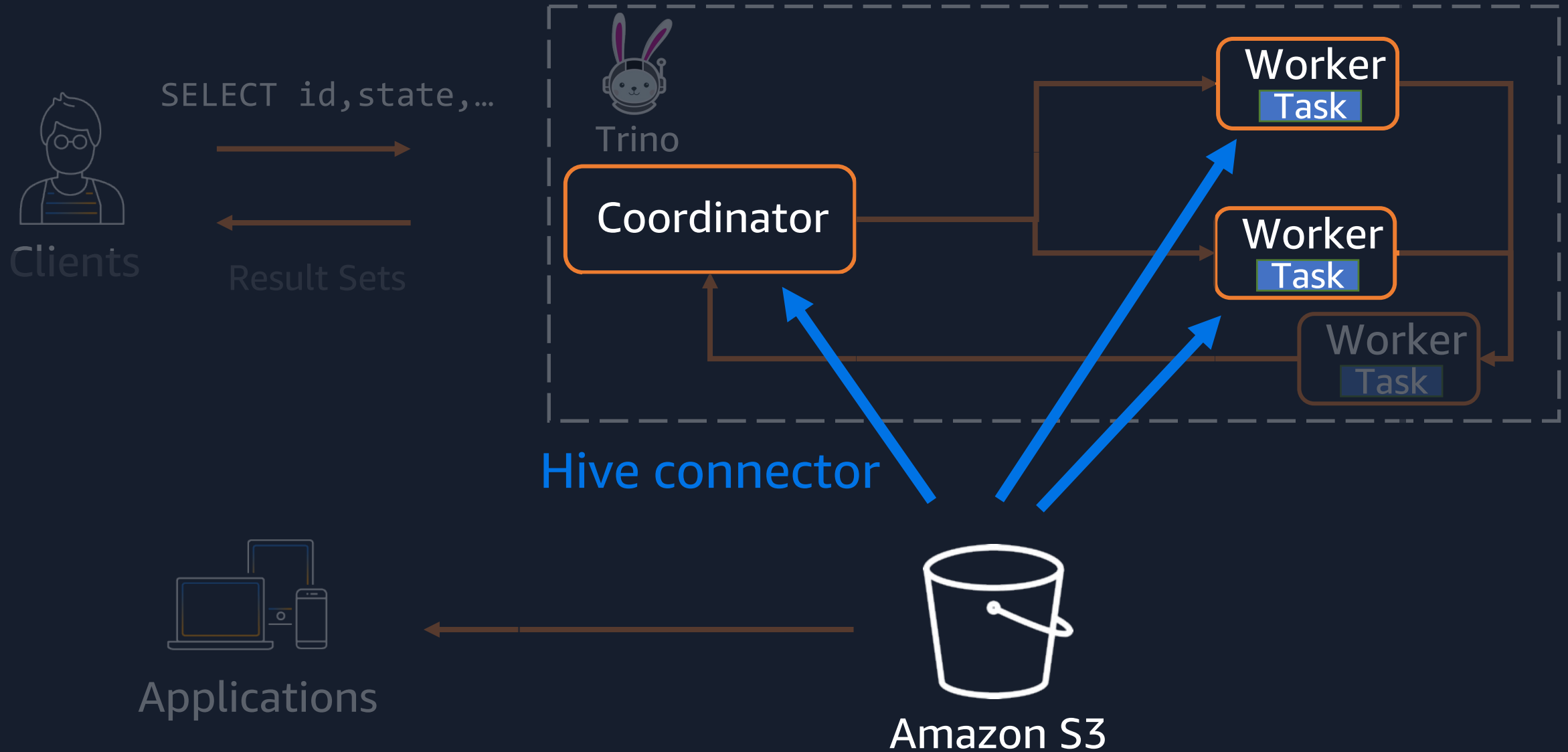
- You can achieve 3,500 PUT/COPY/POST/DELETE requests or 5,500 GET/HEAD requests per second per prefix in a bucket

# Challenge 2: HTTP Slow Down error

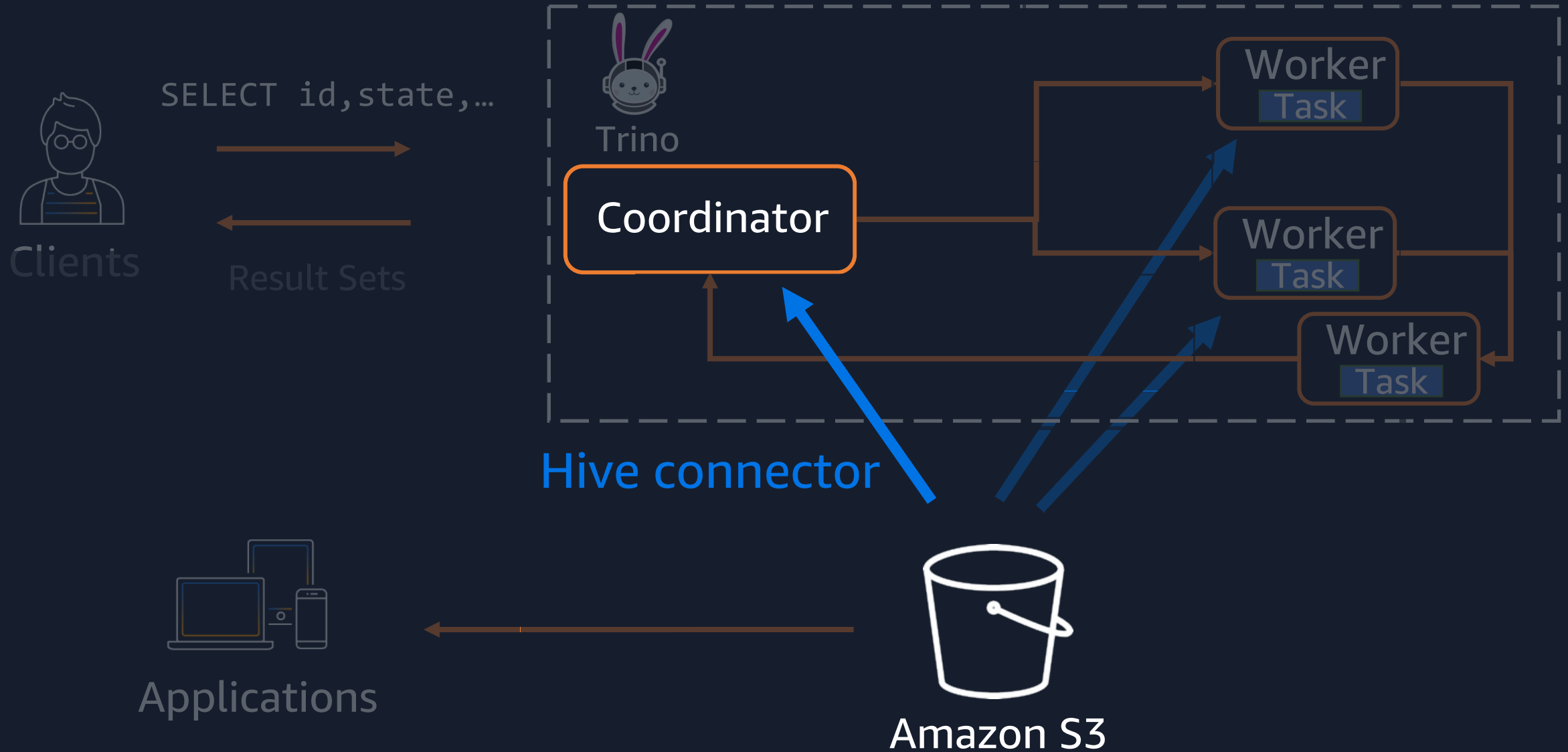
- If your requests exceed threshold, you will face HTTP 503 Slow Down error

```
AmazonS3Exception: Please reduce your request  
rate. (Service: Amazon S3; Status Code: 503;  
Error Code: SlowDown)
```

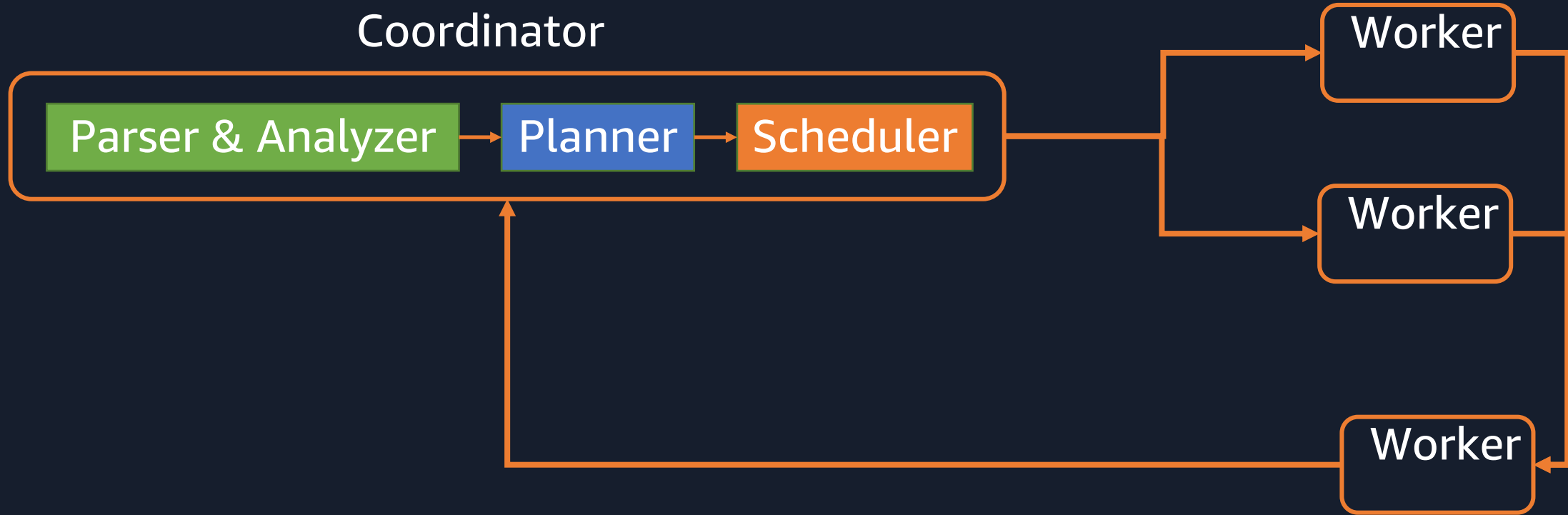
# Challenge 2: HTTP Slow Down error



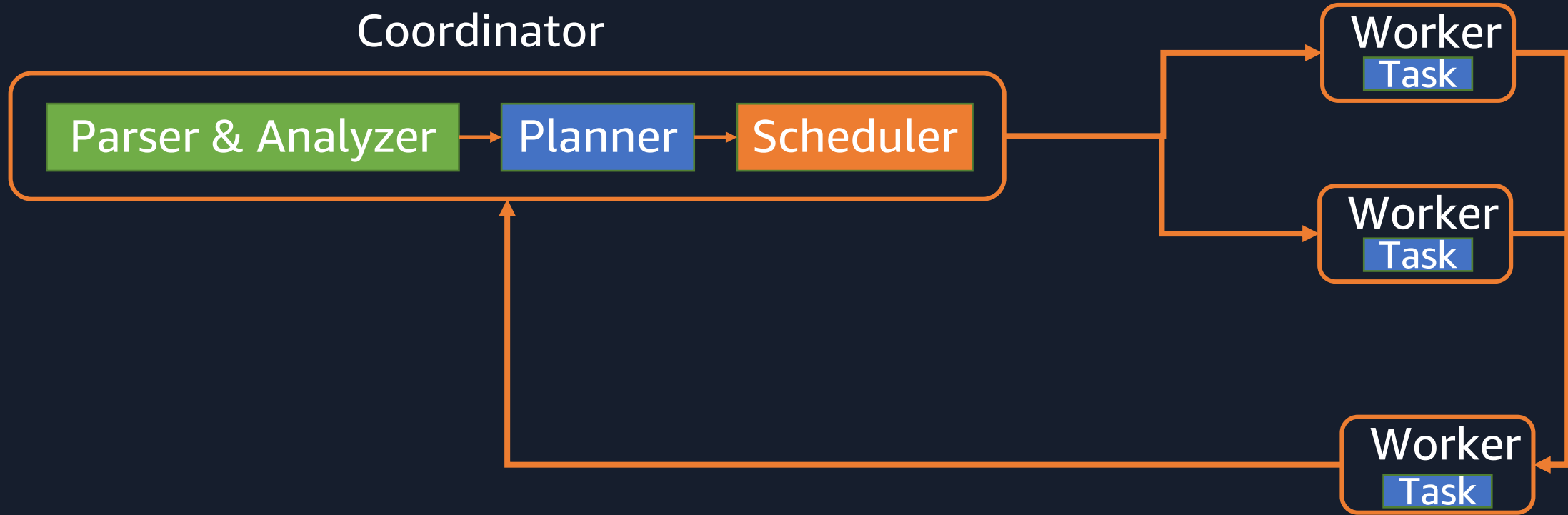
# Challenge 2: HTTP Slow Down error



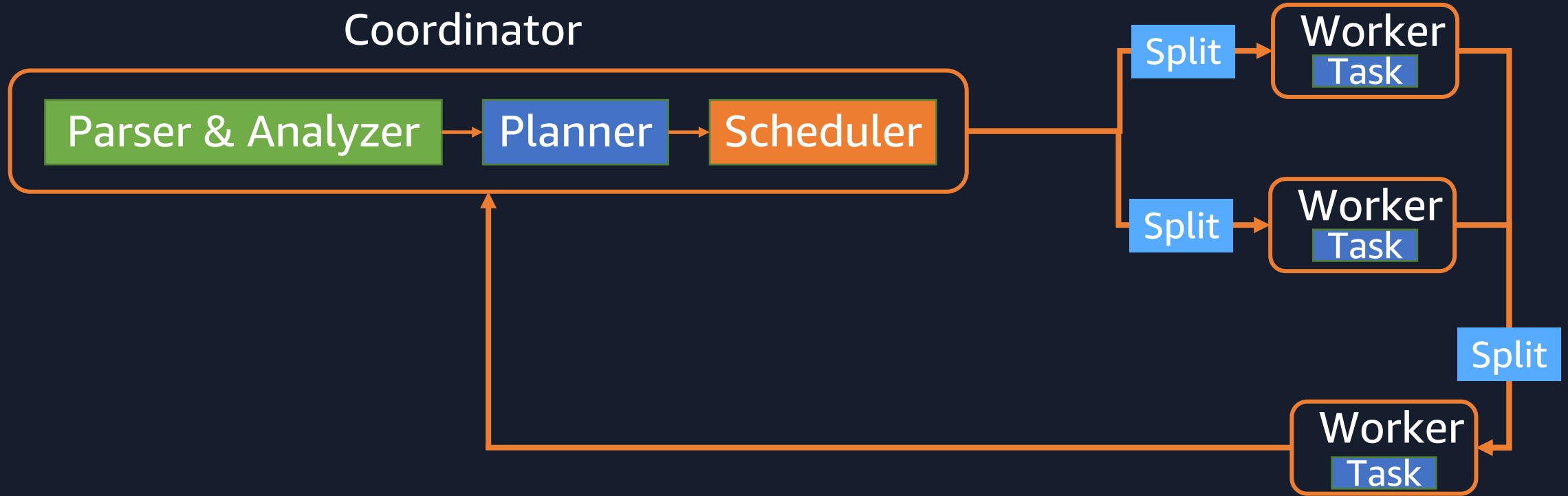
# Challenge 2: HTTP Slow Down error



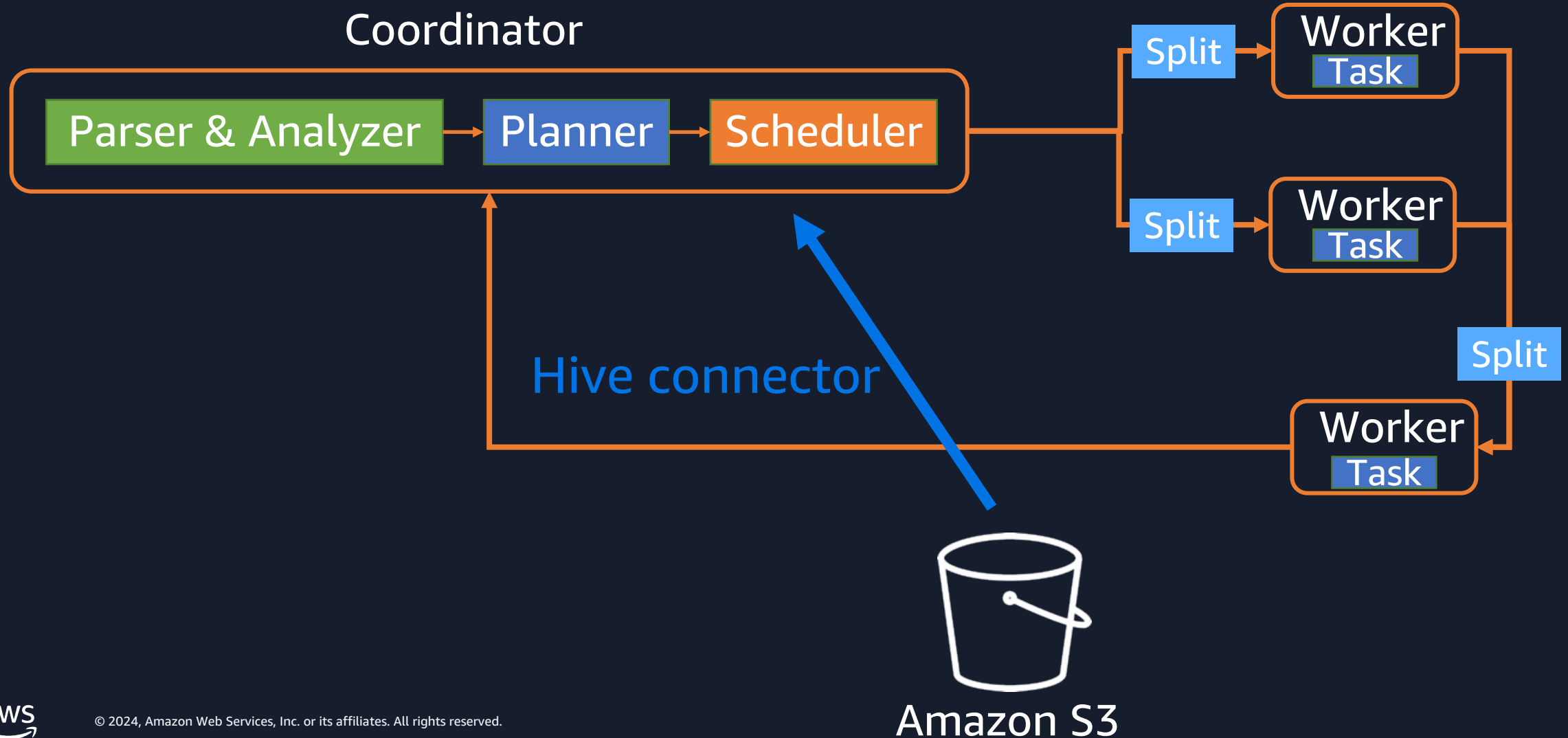
# Challenge 2: HTTP Slow Down error



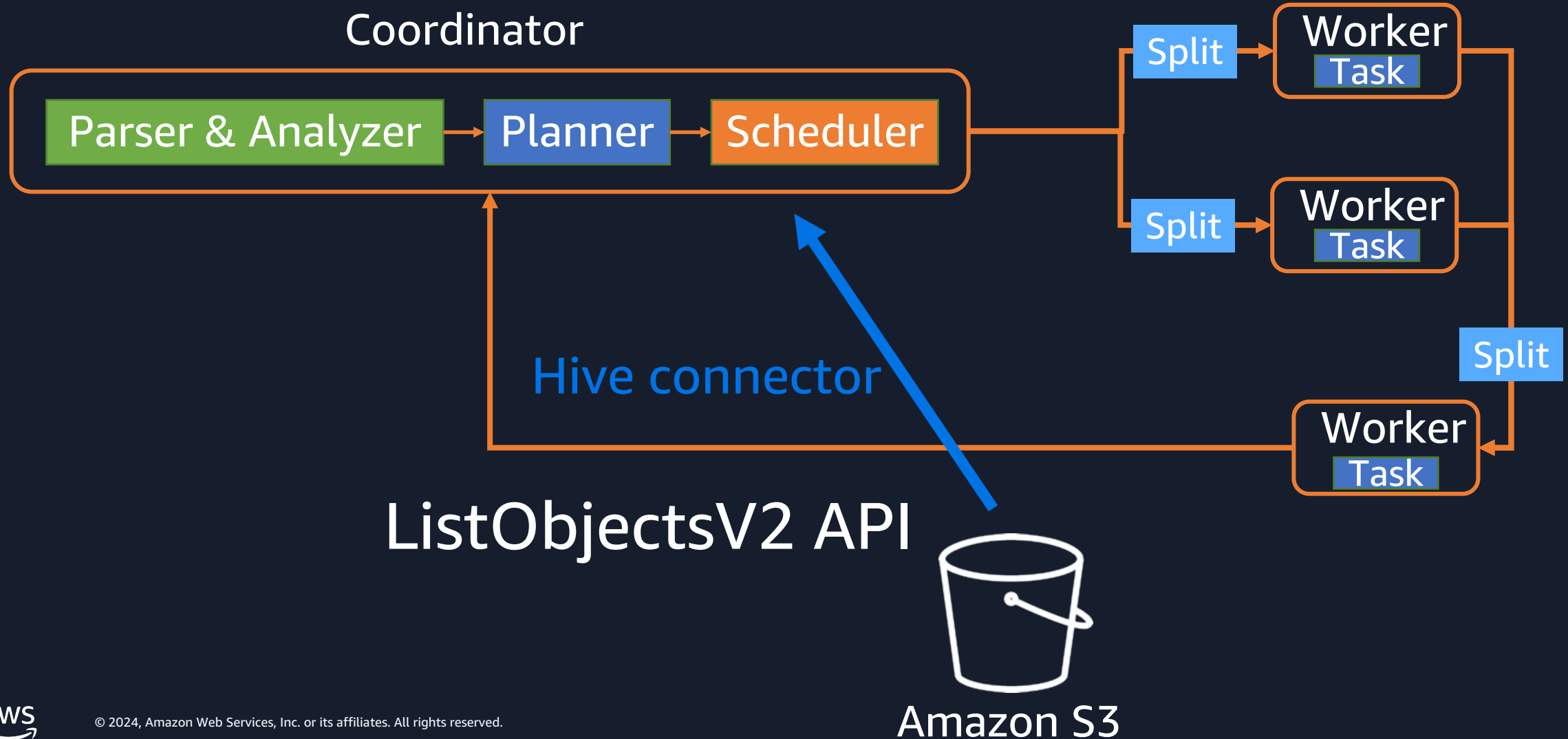
# Challenge 2: HTTP Slow Down error



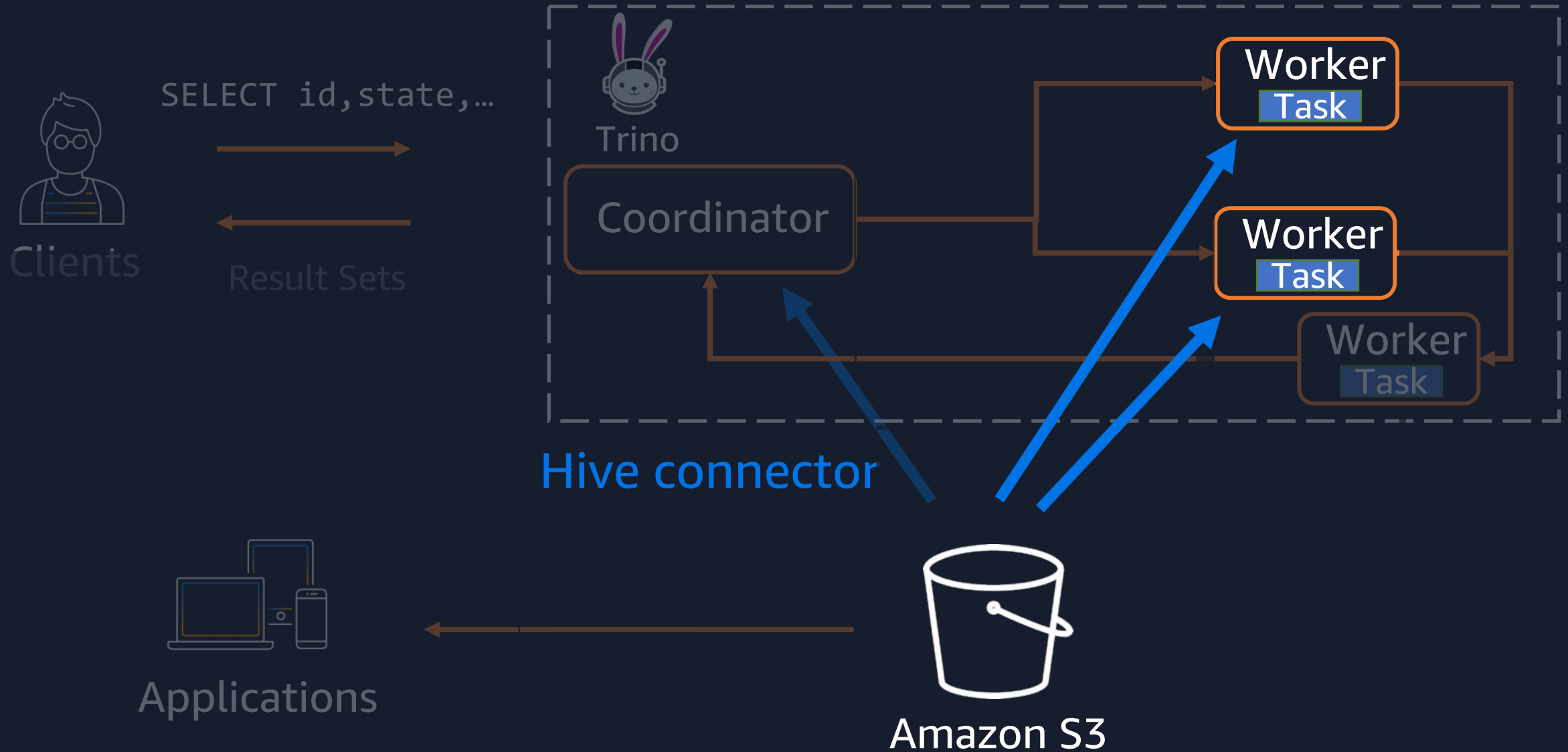
# Challenge 2: HTTP Slow Down error



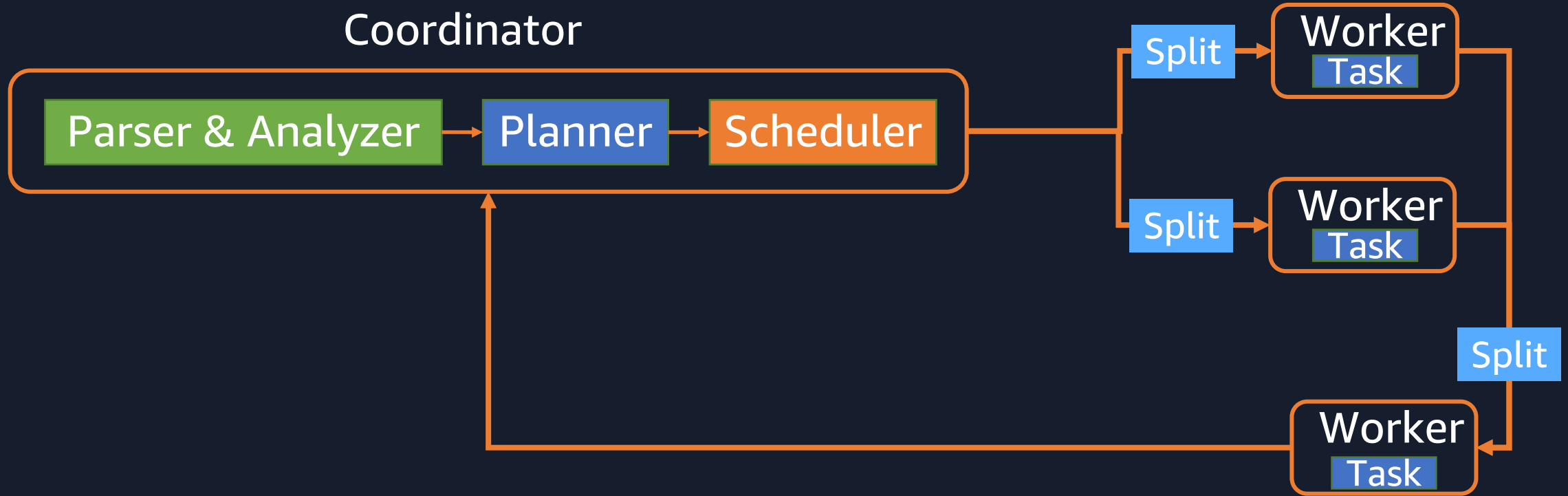
# Challenge 2: HTTP Slow Down error



# Challenge 2: HTTP Slow Down error

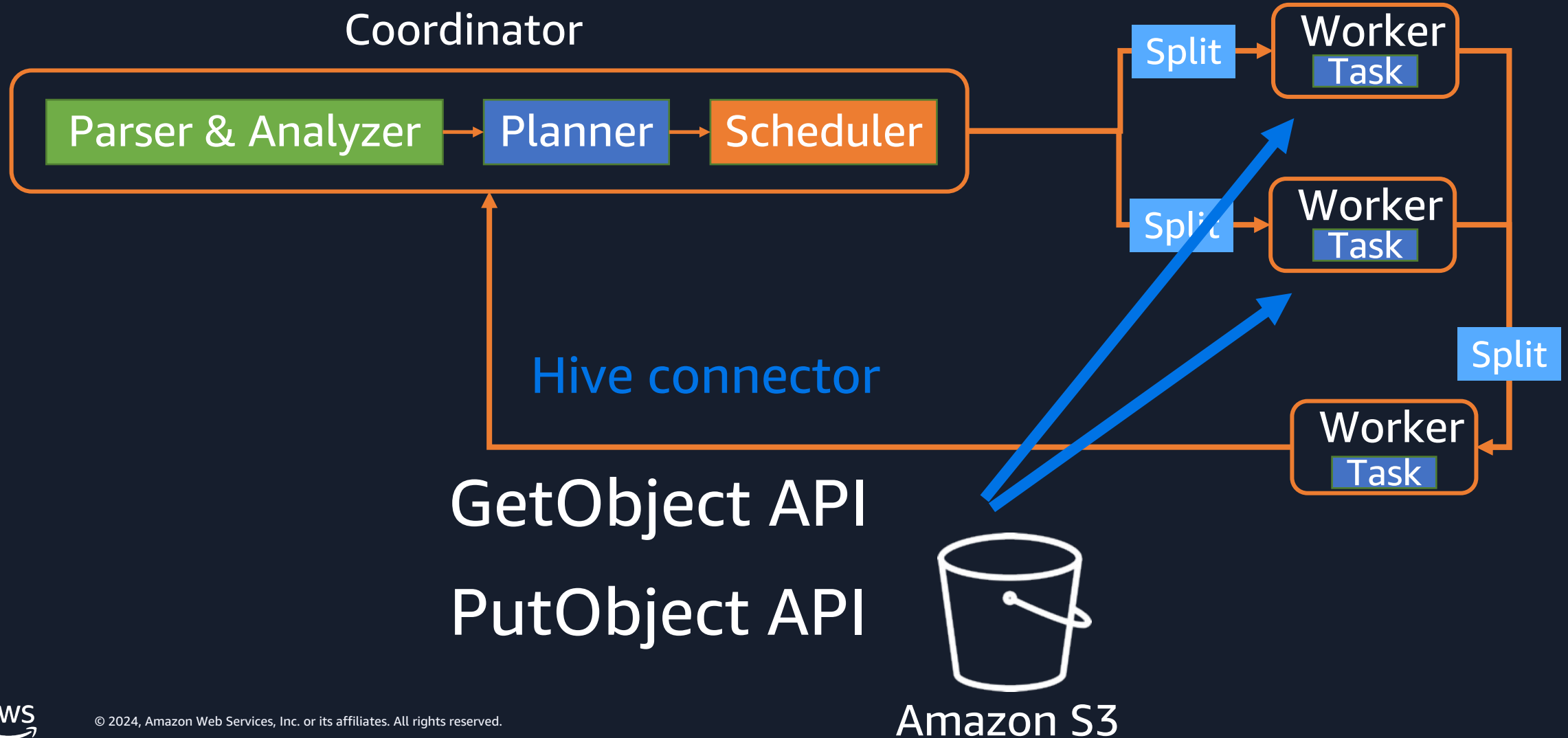


# Challenge 2: HTTP Slow Down error

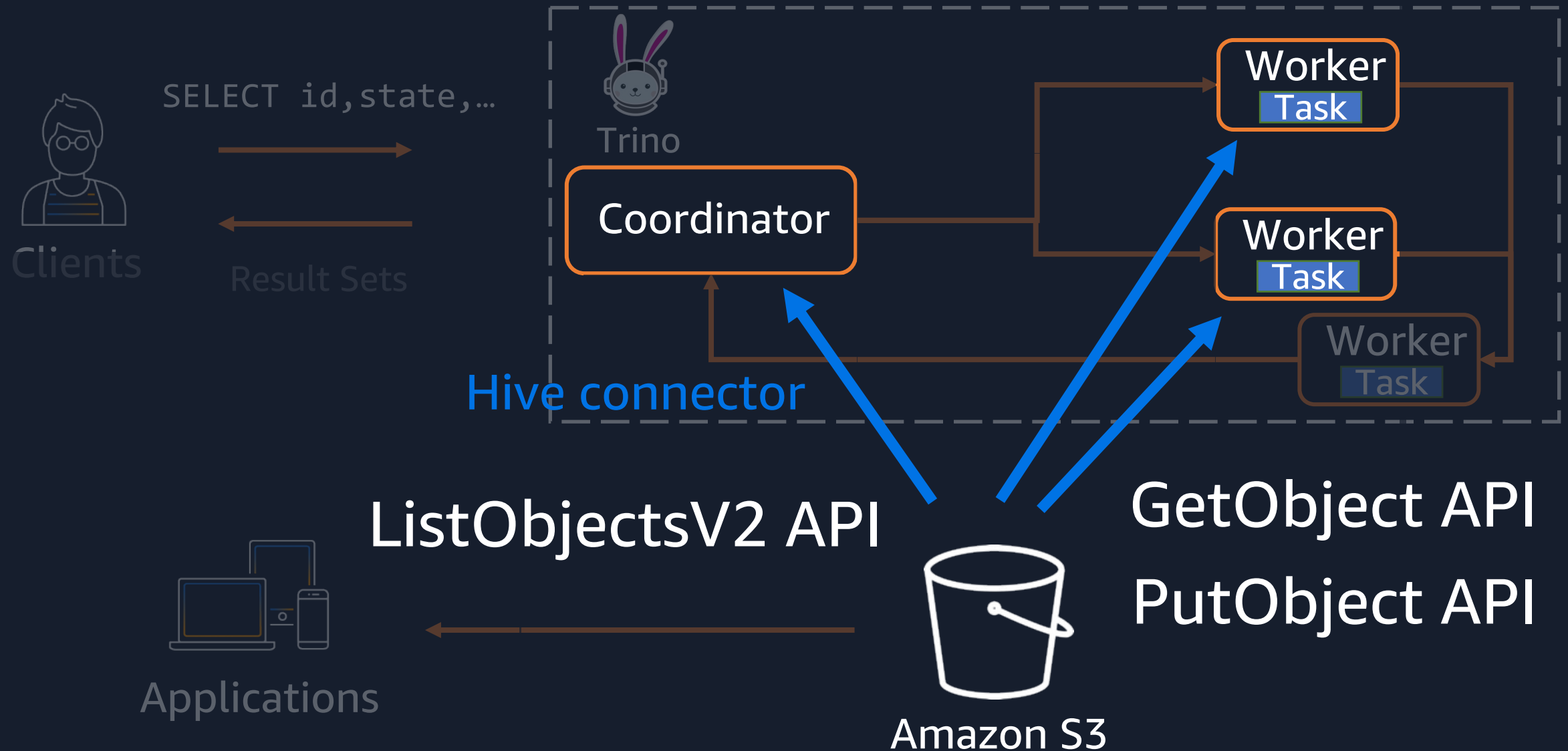


Amazon S3

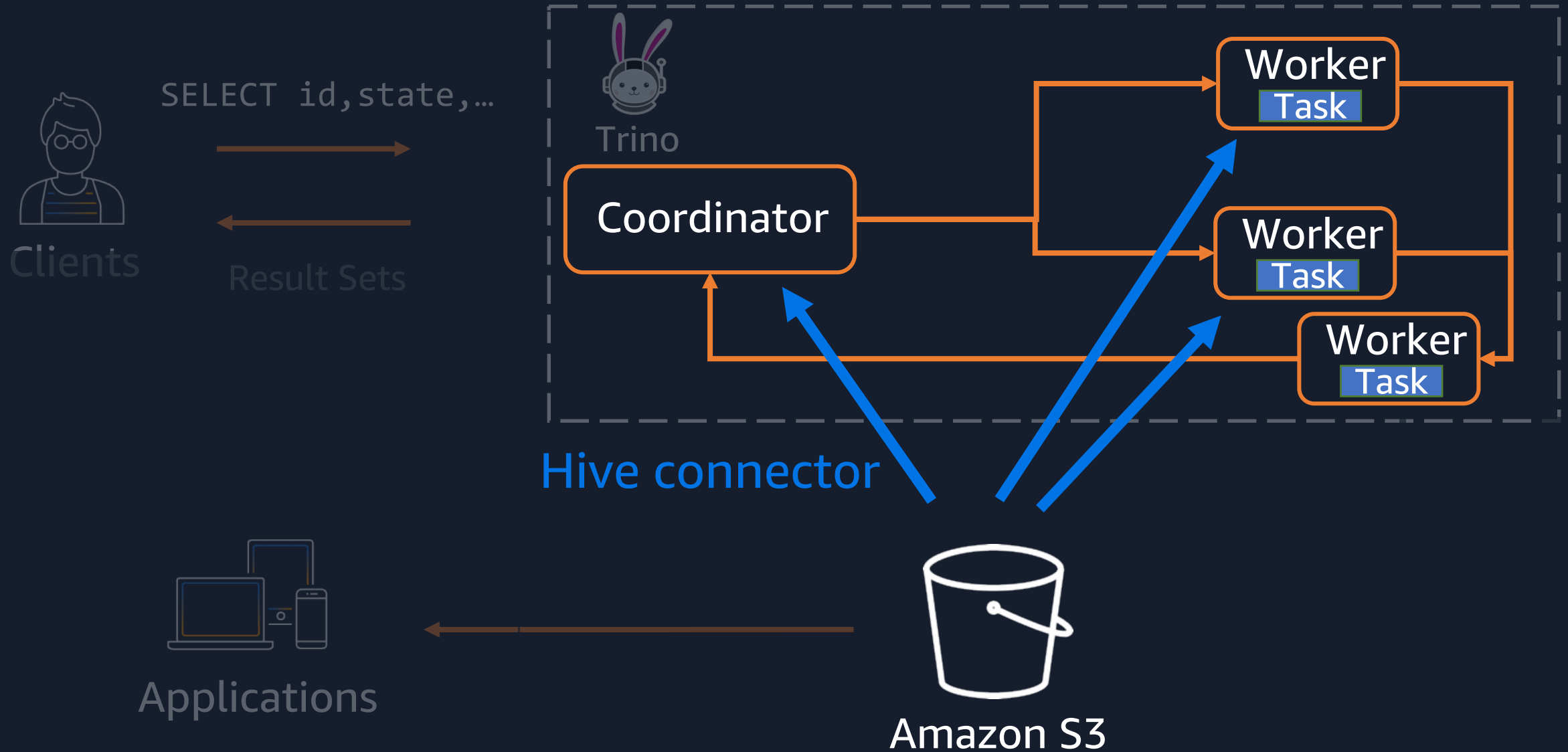
# Challenge 2: HTTP Slow Down error



# Challenge 2: HTTP Slow Down error



# Challenge 3: Many small files issue



# Split



Amazon S3



Split

Split

Split

Split

. . .

- Splits are the smallest unit of work assignment
- Number of splits are related to parallelism
- For query performance, number of splits are important
- Roughly, number of splits can be calculated by below parameters

<code>hive.max-initial-splits</code>	Default: 200
<code>hive.max-initial-split-size</code>	Default: 32MB
<code>hive.max-split-size</code>	Default: 64MB

# How to estimate parallelism

Example1: 1000 files, each file size is 10 KB

1. Initial 200 files are smaller than `hive.max-initial-split-size`

First 200 files are 200 splits

2. Each of the remaining 800 files are smaller than `hive.max-split-size`

Remaining 800 files are 800 splits

Total: 1000 splits

<code>hive.max-initial-splits</code>	Default: 200
<code>hive.max-initial-split-size</code>	Default: 32MB
<code>hive.max-split-size</code>	Default: 64MB

# How to estimate parallelism

Example2: 10 files, each file size is 1000 KB

1. 10 files are smaller than `hive.max-initial-split-size`

Total: 10 splits

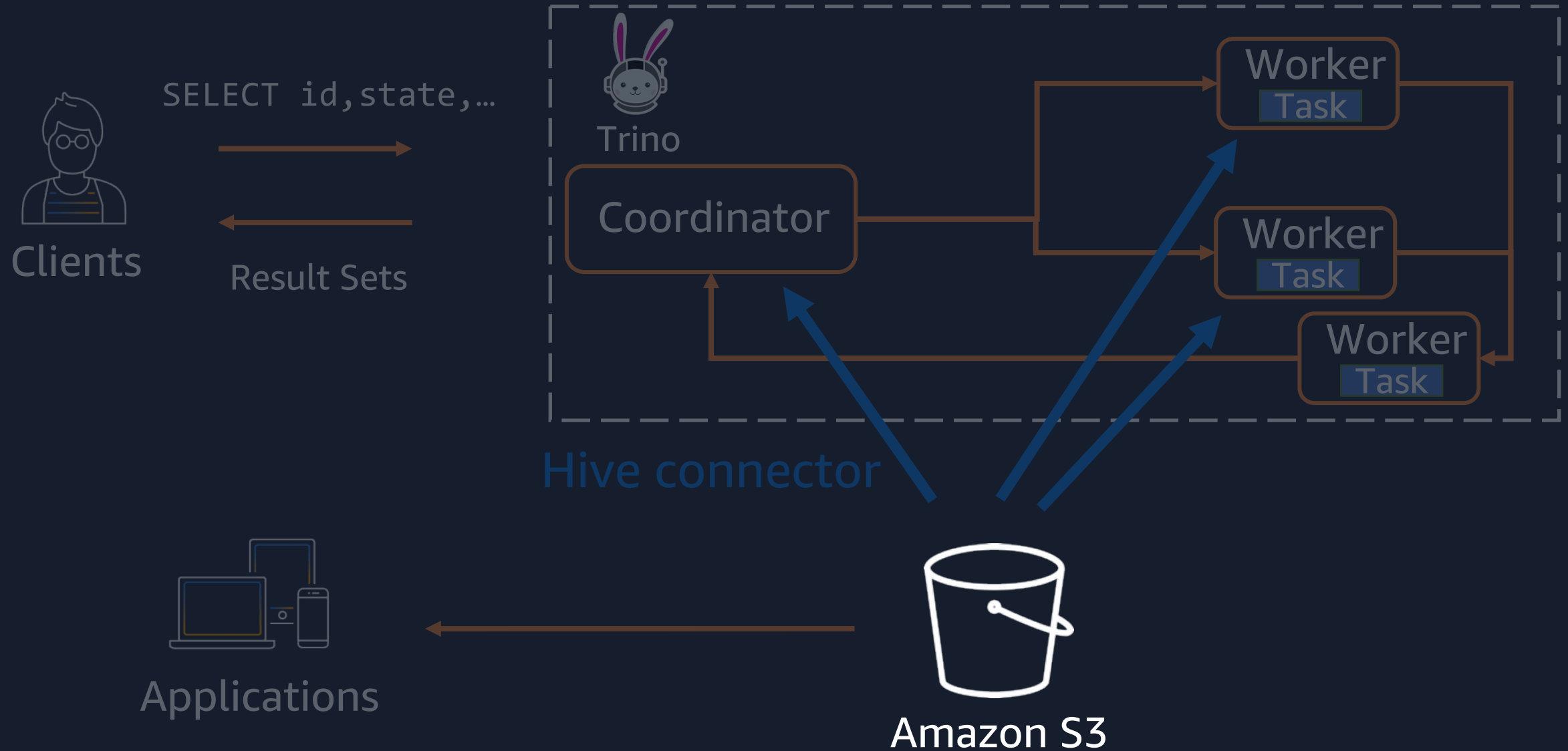
<code>hive.max-initial-splits</code>	Default: 200
<code>hive.max-initial-split-size</code>	Default: 32MB
<code>hive.max-split-size</code>	Default: 64MB

# Challenge 3: Many small files issue

What happen when reading many small files?

- Heavy I/O load to Amazon S3 due to LIST/GET requests
- Generates many splits and it generates computational overhead

# Challenge 4: Unneeded data is stored



# Challenge 4: Unneeded data is stored

Data is growing

- The storage cost is getting higher
- There are data with know or predictable access patterns and data with unknow or changing access patterns.
- How to delete irrelevant data ?

# Common challenges in scaling Trino workload



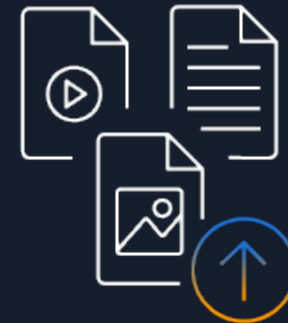
Huge data scan



HTTP Slow Down error














Many small files issue







Unneeded data is stored












# Best practices to scale workload with Amazon S3





# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Partitioning

■ Huge data scan    ■ HTTP Slow Down error

- Partitioning divides your table into parts and keeps the related data together based on column values
- By using partitioning, you can reduce the amount of data scanned per query

```
partitioned_by = ARRAY['view_date']
```

```
SELECT * FROM example.web.request_logs  
WHERE view_date=2024-06-13
```

Table



Partition

view\_date=2024-06-13

...

Partition

view\_date=2024-06-14

s3://mybucket/daily\_uploads/

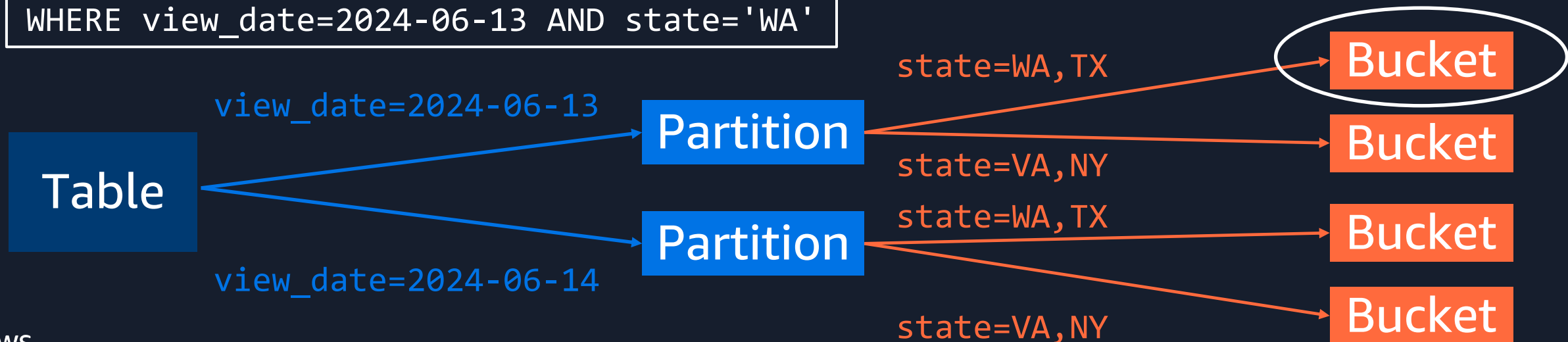
# Bucketing

■ Huge data scan   ■ HTTP Slow Down error

- With bucketing, you can specify one or more columns containing rows that you want to group together, and put those rows into multiple buckets.

```
partitioned_by = ARRAY['view_date']  
bucketed_by = ARRAY['state'],  
bucket_count = 50
```

```
SELECT * FROM example.web.request_logs  
WHERE view_date=2024-06-13 AND state='WA'
```



# Partitioning / Bucketing



Huge data scan



HTTP Slow Down error

## Partition columns

- Pick partition keys based on common query pattern
- Partition keys should have a relatively low cardinality

## Columns to bucket on

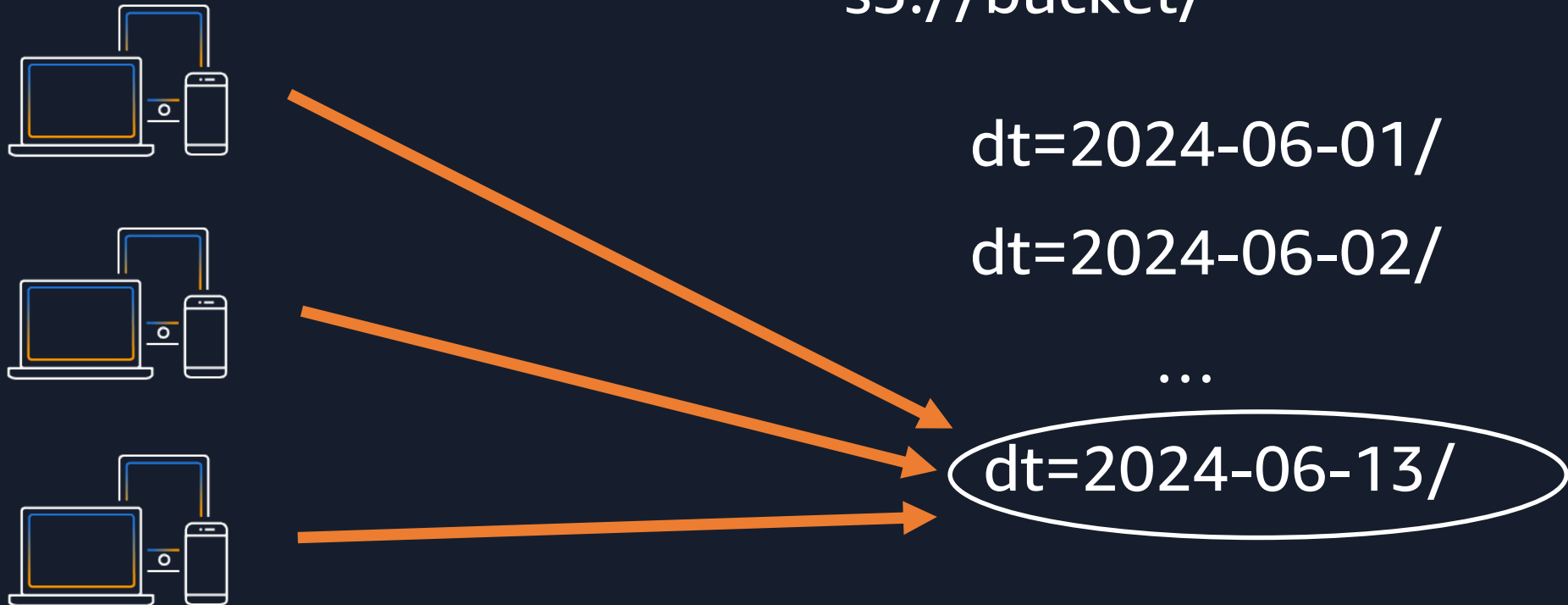
- Choose columns that have high cardinality
- Many of your queries look up specific values of the key

# Managing S3 prefixes

■ HTTP Slow Down error

Add S3 prefixes to scale S3 performance

Example:




# Managing S3 prefixes

■ HTTP Slow Down error

Add S3 prefixes to scale S3 performance

Example:

s3://bucket/



AmazonS3Exception: Please reduce your request rate. (Service: Amazon S3; Status Code: 503; Error Code: SlowDown)

...

dt=2024-06-13/



# Managing S3 prefixes

 HTTP Slow Down error

Add S3 prefixes to scale S3 performance

Example:

s3://bucket/dt=2024-06-13



s3://bucket/country=US/dt=2024-06-13

s3://bucket/country=CA/dt=2024-06-13

s3://bucket/country=JP/dt=2024-06-13

# Managing S3 prefixes

 HTTP Slow Down error

Add S3 prefixes to scale S3 performance

Example:

s3://bucket/dt=2024-06-13

} 5,500 requests per second



s3://bucket/country=US/dt=2024-06-13

s3://bucket/country=CA/dt=2024-06-13

s3://bucket/country=JP/dt=2024-06-13

} **16,500 requests per second**

# Managing S3 prefixes












 HTTP Slow Down error





Add S3 prefixes to scale S3 performance

Which columns from data should we select when adding s3 prefixes?

- Choose the columns which has multiple different values over recent records
- Choose the columns which are frequently used as a predicate in your queries

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Optimizing data size

■ HTTP Slow Down error ■ Many Small files issue

- Run OPTIMIZE command

( 0. If you run OPTIMIZE command against Hive external tables, set this parameter)

```
hive.non-managed-table-writes-enabled=true
```

1. Set session parameter

```
SET SESSION <catalog>.non_transactional_optimize_enabled=true
```

2. Run OPTIMIZE command

```
ALTER TABLE <catalog>.<schema>.<table> EXECUTE  
optimize(file_size_threshold => '128MB')
```

\* file\_size\_threshold is 100 MB by default

# Optimizing data size

■ HTTP Slow Down error ■ Many Small files issue












If you use Trino on Amazon Athena, and your table is a hive external table





- Migrate the table to Iceberg table

```
CREATE TABLE iceberg_table
WITH (table_type = 'ICEBERG',
      format = 'PARQUET',
      location = 's3:// bucket /iceberg/',
      is_external = false,
      partitioning = ARRAY['country_code'])
AS SELECT id, name, country_code FROM table1;
```

- Use automatic compaction feature on Glue Data Catalog

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Making well-designed retries

■ HTTP Slow Down error

Increase retry limit for Amazon S3 requests in Trino

Native implementation ( `fs.native-s3.enable=true` )

`s3.max-error-retries`

From Trino 449

Legacy version ( `fs.native-s3.enable=false` )

`hive.s3.max-client-retries`

# Making well-designed retries












 HTTP Slow Down error





## Increase retry limit in Trino on Amazon EMR

EMRFS

```
fs.s3.maxRetries
```

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Taking advantage of Amazon S3 Storage Class

Unneeded data is stored



# Taking advantage of Amazon S3 Storage Class

■ Unneeded data is stored



1

Data with known or predictable access patterns

2

Data with unknown or changing access patterns

# Taking advantage of Amazon S3 Storage Class

■ Unneeded data is stored



1

Data with known or predictable access patterns

2

Data with unknown or changing access patterns

# Data with known or predictable access patterns

Unneeded data is stored

S3 Express  
One Zone



Most frequently accessed data

Single-digit millisecond access

S3 Standard



Frequently accessed data

S3 Standard-IA



Infrequently accessed data

S3 One Zone-IA



Re-creatable, less accessed data

S3 Glacier Instant Retrieval



Rarely accessed data

S3 Glacier Flexible Retrieval



Archive data

S3 Glacier Deep Archive



Long-term archive data

Milliseconds access

Minutes to hours

# Data with known or predictable access patterns

■ Unneeded data is stored



- Storage cost for S3 Standard-IA is cheaper than S3 Standard
- S3 request cost for S3 Standard-IA is higher than S3 Standard

S3 Standard-IA is suitable for infrequently accessed data

# Data with known or predictable access patterns

Unneeded data is stored

S3 Express  
One Zone



Most frequently accessed data

Single-digit millisecond access

S3 Standard



Frequently accessed data

S3 Standard-IA



Infrequently accessed data

S3 One Zone-IA



Re-creatable, less accessed data

S3 Glacier Instant Retrieval



Rarely accessed data

S3 Glacier Flexible Retrieval



Archive data

S3 Glacier Deep Archive



Long-term archive data

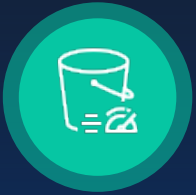
Minutes to hours



# Data with known or predictable access patterns

Unneeded data is stored

S3 Express  
One Zone



Most  
frequently  
accessed data

Single-digit  
millisecond  
access

- Lowest latency
- Most expensive for storage cost
- Cheapest for request cost
- Less available

# Data with known or predictable access patterns

■ Unneeded data is stored

S3 Express  
One Zone



Most  
frequently  
accessed data

Single-digit  
millisecond  
access

## Trade off

Latency

Storage cost

Request cost

Availability

## Data with known or predictable access patterns Unneeded data is stored

### Situation

- You create a daily report by using Trino
- The data is stored in S3 Standard
- You frequently access the data for a short period
- The data is rarely accessed again after a month or two

You can consider moving to another S3 class

# Taking advantage of Amazon S3 Storage Class

■ Unneeded data is stored



1

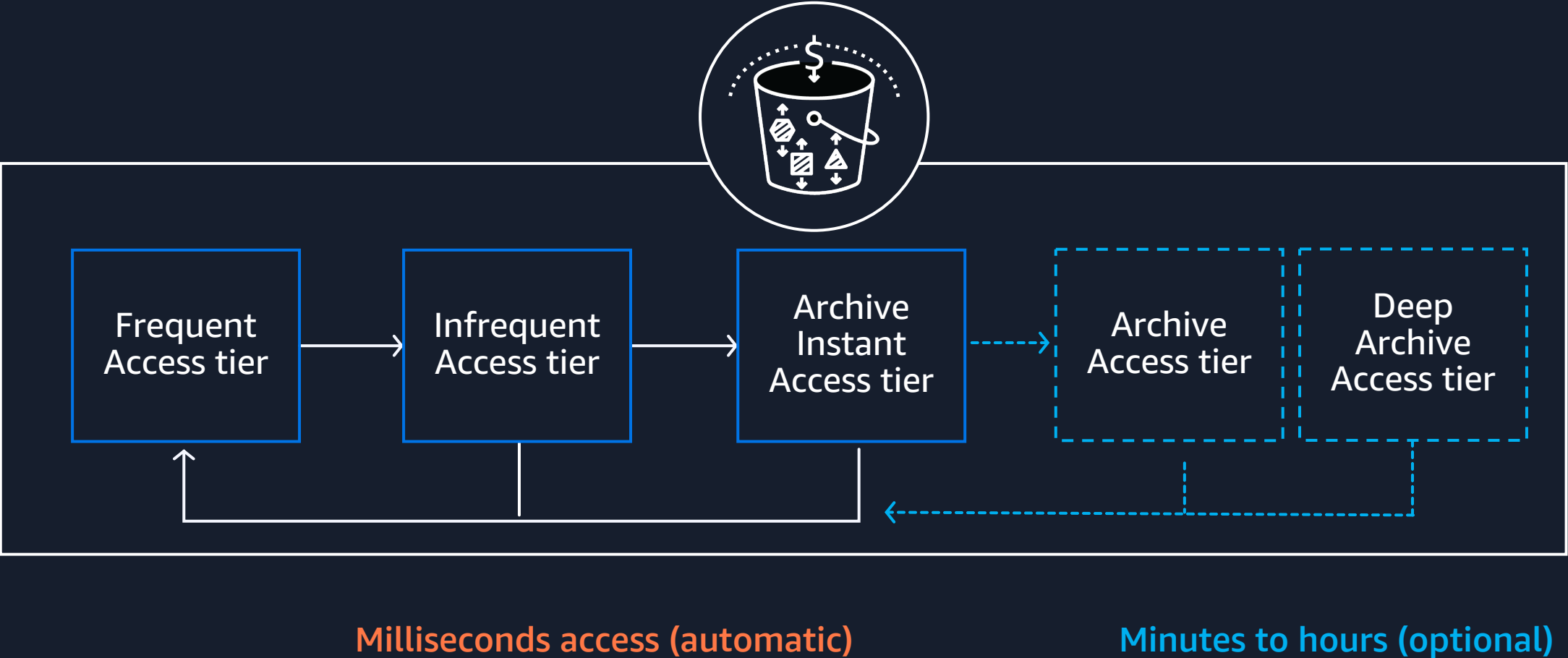
Data with known or predictable access patterns

2

Data with unknown or changing access patterns

# S3 Intelligent-Tiering storage class

Unneeded data is stored



## How to read/write data in different storage class in Trino ■ Unneeded data is stored

- You can read objects stored in S3 Standard/S3 Standard-IA/S3 Intelligent-Tiering/S3 Glacier Instant Retrieval storage class without additional parameters

### Native implementation ( `fs.native-s3.enable=true` )

- You can read restored glacier objects by default

### Legacy version ( `fs.native-s3.enable=false` )

- Skip glacier objects by setting `hive.s3.skip-glacier-objects`
- You can read restored glacier objects by default
- You can write data in Intelligent-Tiering by setting `hive.s3.storage-class`

# How to read/write data in different storage class in Trino ■ Unneeded data is stored












## Athena





- Skip glacier objects by default
- You can read restored glacier objects by setting `hive.restored_glacier_objects`

## EMR

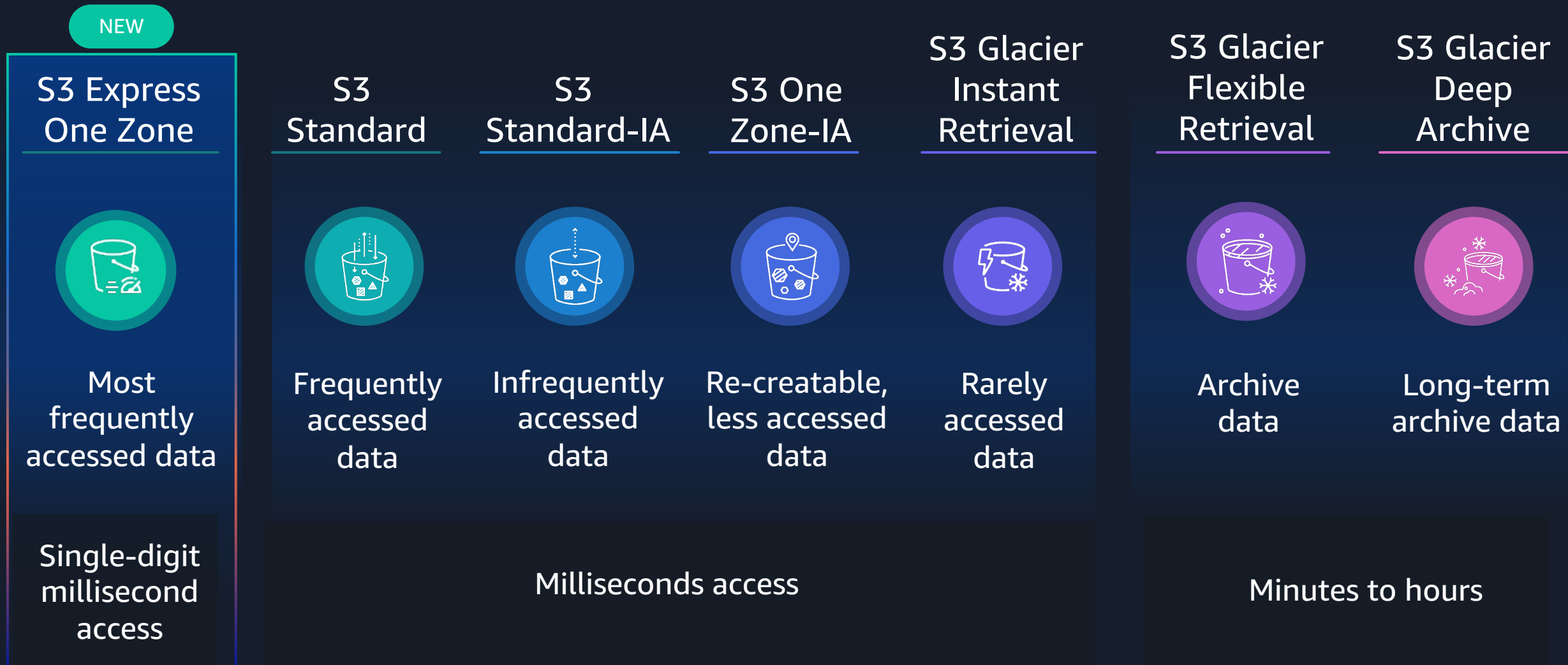
- You can read restored glacier objects by default

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

# Reducing latency with Amazon Express One Zone



# Reducing latency with Amazon S3 Express One Zone














## Scalable





- No per-prefix transaction limits
- Support hundreds of thousands of transactions per second (TPS)

### How to use Amazon Express One Zone

Application/Service	Parameter
Trino	fs.native-s3.enabled=true
Amazon Athena	Not required
Amazon EMR	fs.native-s3.enabled=true hive.s3-file-system-type=TRINO

# Best practices to scale workload with Amazon S3

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Reducing latency with Amazon Express One Zone 
- Taking advantage of Amazon S3 Storage Class 
- Managing data life cycle 

-  Huge data scan
-  HTTP Slow Down error
-  Many Small files issue
-  Unneeded data is stored

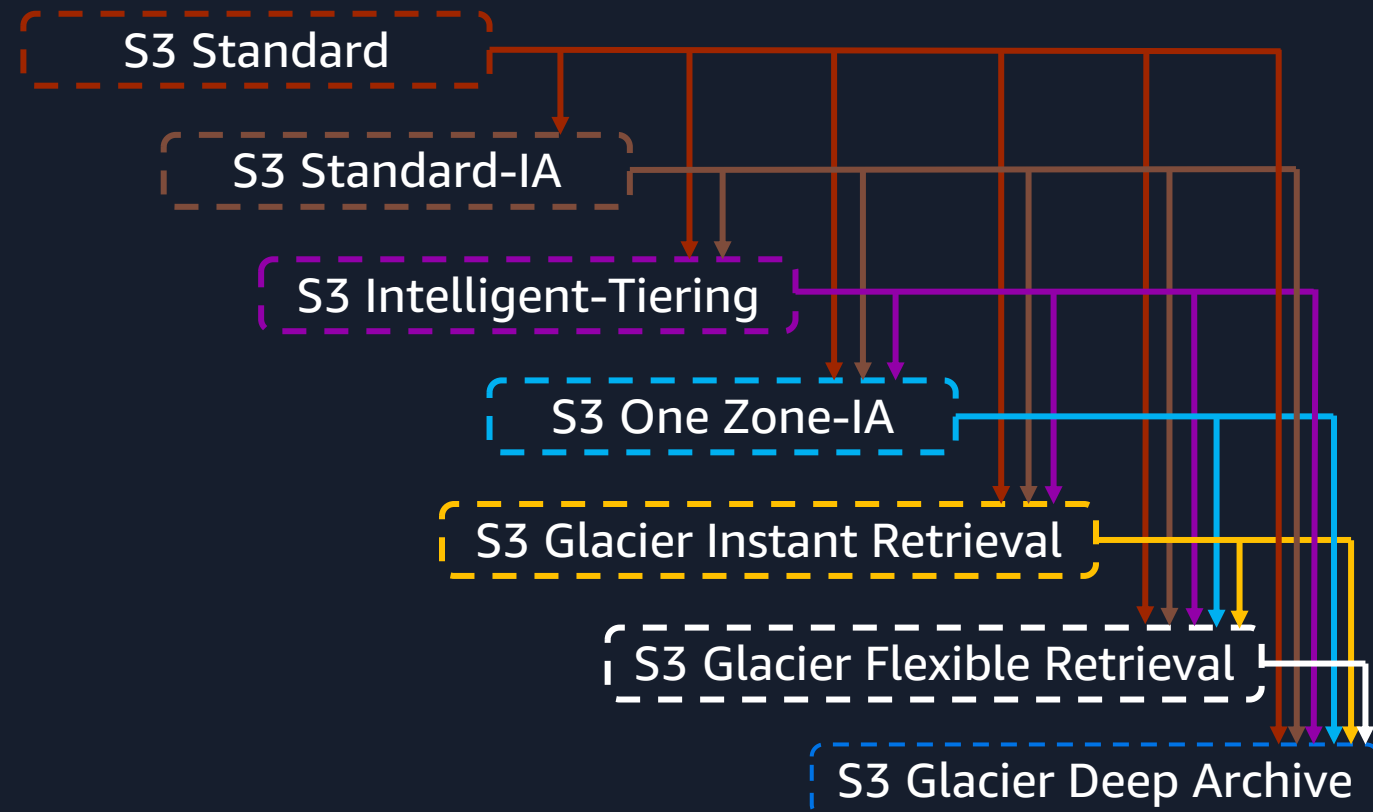
# Managing data life cycle

■ Unneeded data is stored

- Lifecycle rules take action based on object age

**Transition actions:** Define when objects transition to other Amazon S3 storage classes as they age





**Expiration actions:** Define when objects expire; Amazon S3 deletes expired objects on your behalf














# Summary

# Summary

## Common challenges

-  Huge data scan
-  HTTP Slow Down error
-  Many Small file issues
-  Unneeded data is stored

## Best practices

- Optimizing data layout
  - Partitioning  
  - Bucketing  
  - Managing S3 prefixes 
- Optimizing data size  
- Making well-designed retries 
- Taking advantage of Amazon S3 Storage Class 
- Reducing latency with Amazon Express One Zone 
- Managing data life cycle 

# Summary

For other challenges, contact AWS Support!

# Thank you!

Dai Ozaki

 [in/dai-ozaki/](https://www.linkedin.com/in/dai-ozaki/)