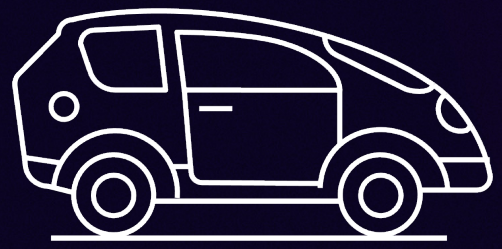
SVS217-NEW

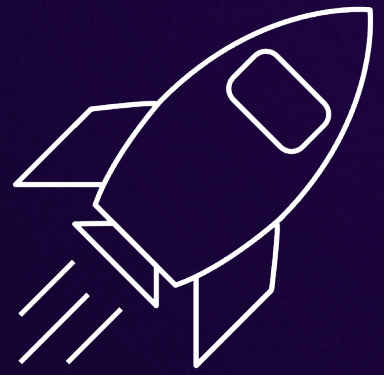
# Improve throughput and monitoring of serverless streaming workloads

**Anton Aleksandrov**
Principal Solutions Architect, Serverless
AWS

# Let’s talk about cars

# Let’s talk about cars

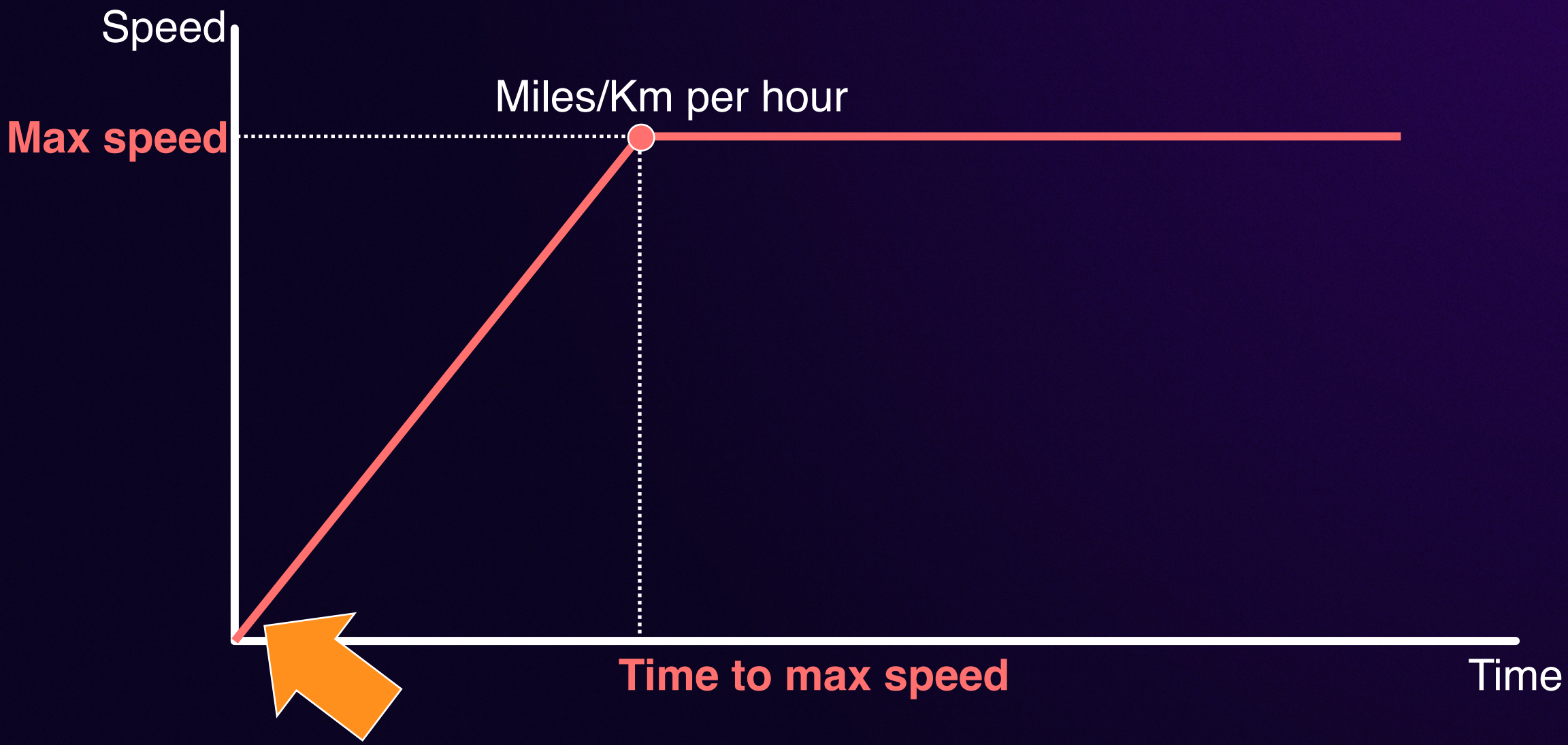




**Max speed**

**Acceleration**

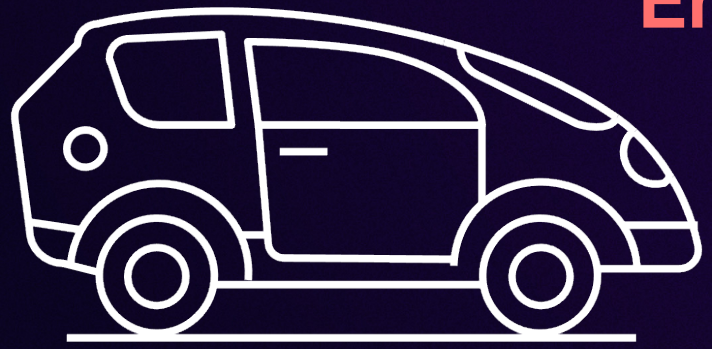
# Let’s talk about cars



# Let’s talk about cars

Speed

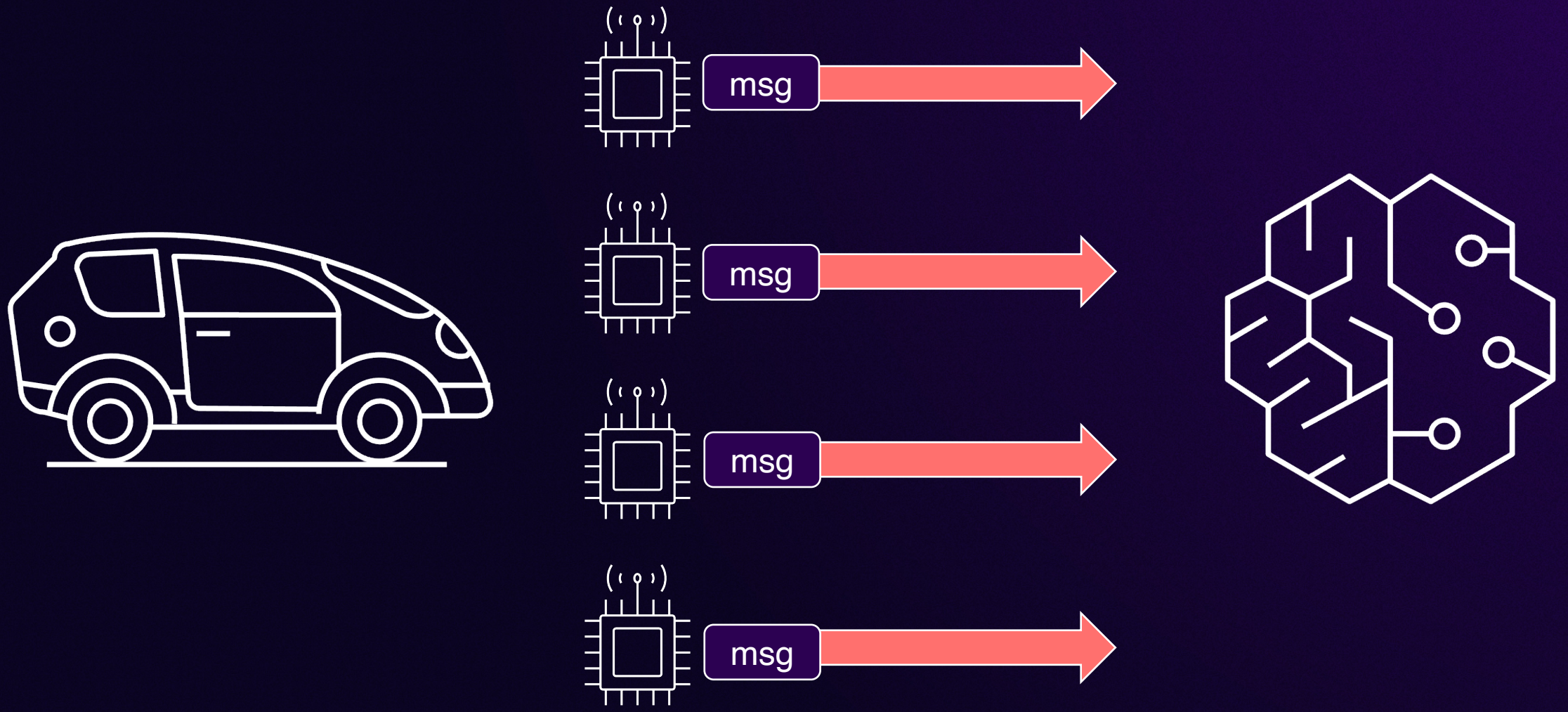
Engine RPM



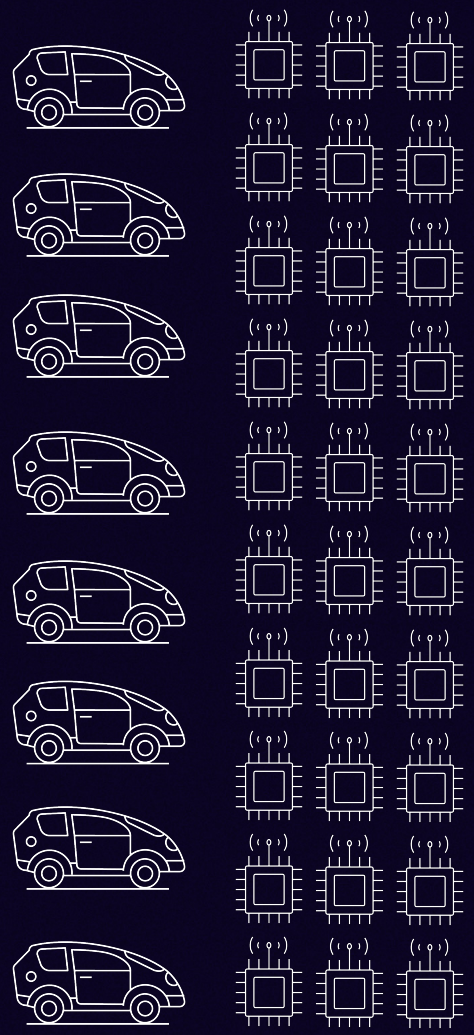
Gas remaining

Tire pressure

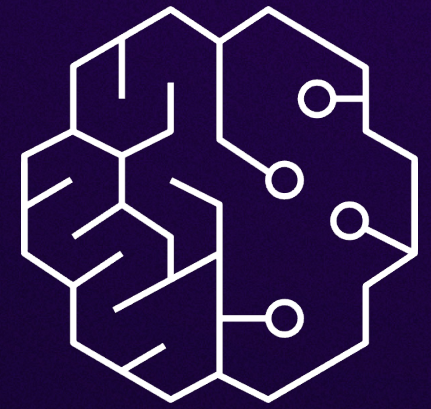
# Let’s talk about cars



# Let’s talk about streaming data processing

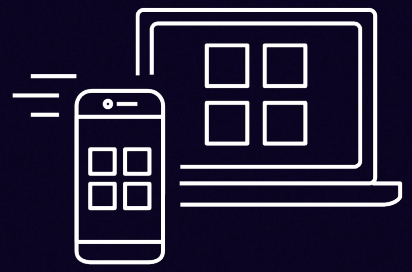


?

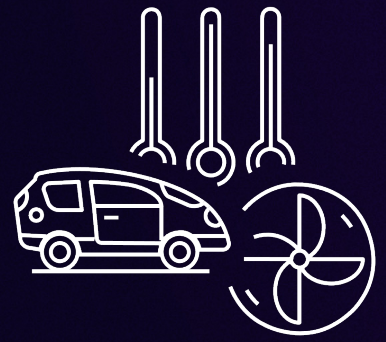


# Let’s talk about streaming data processing

# Streaming workloads use cases



Application click streams



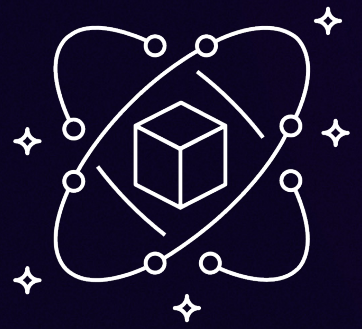
Connected devices, IoT

Financial data, stock tickers

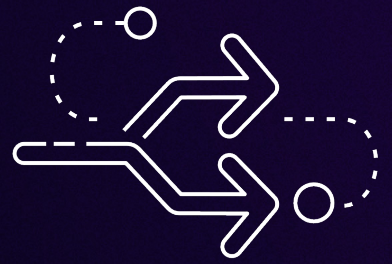
Real-time anomaly and fraud detection

# Streaming workloads characteristics

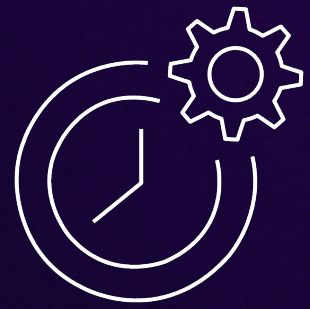
High volume



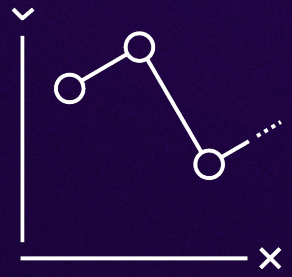
Continuous



Ordered

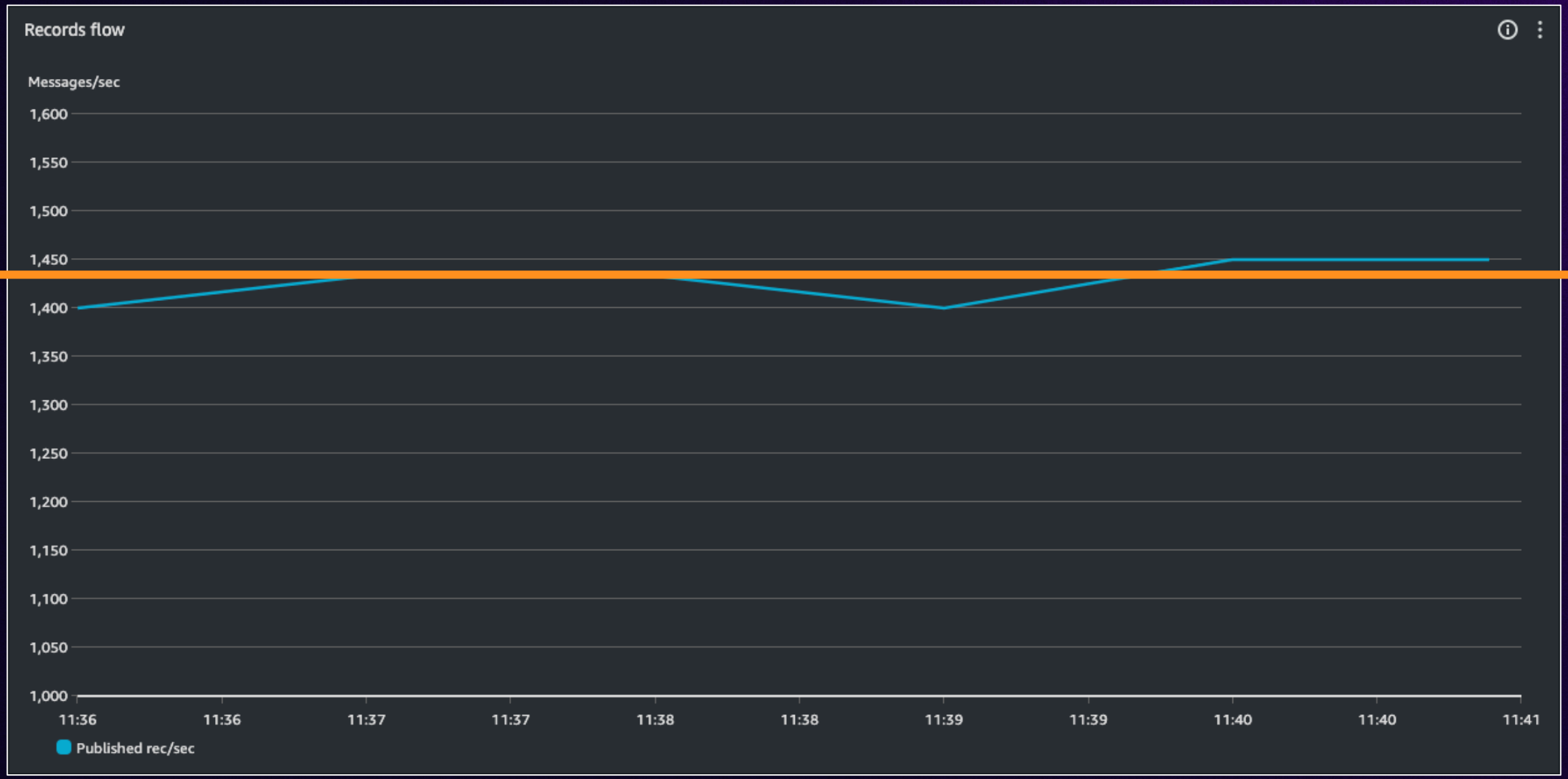


Time-sensitive

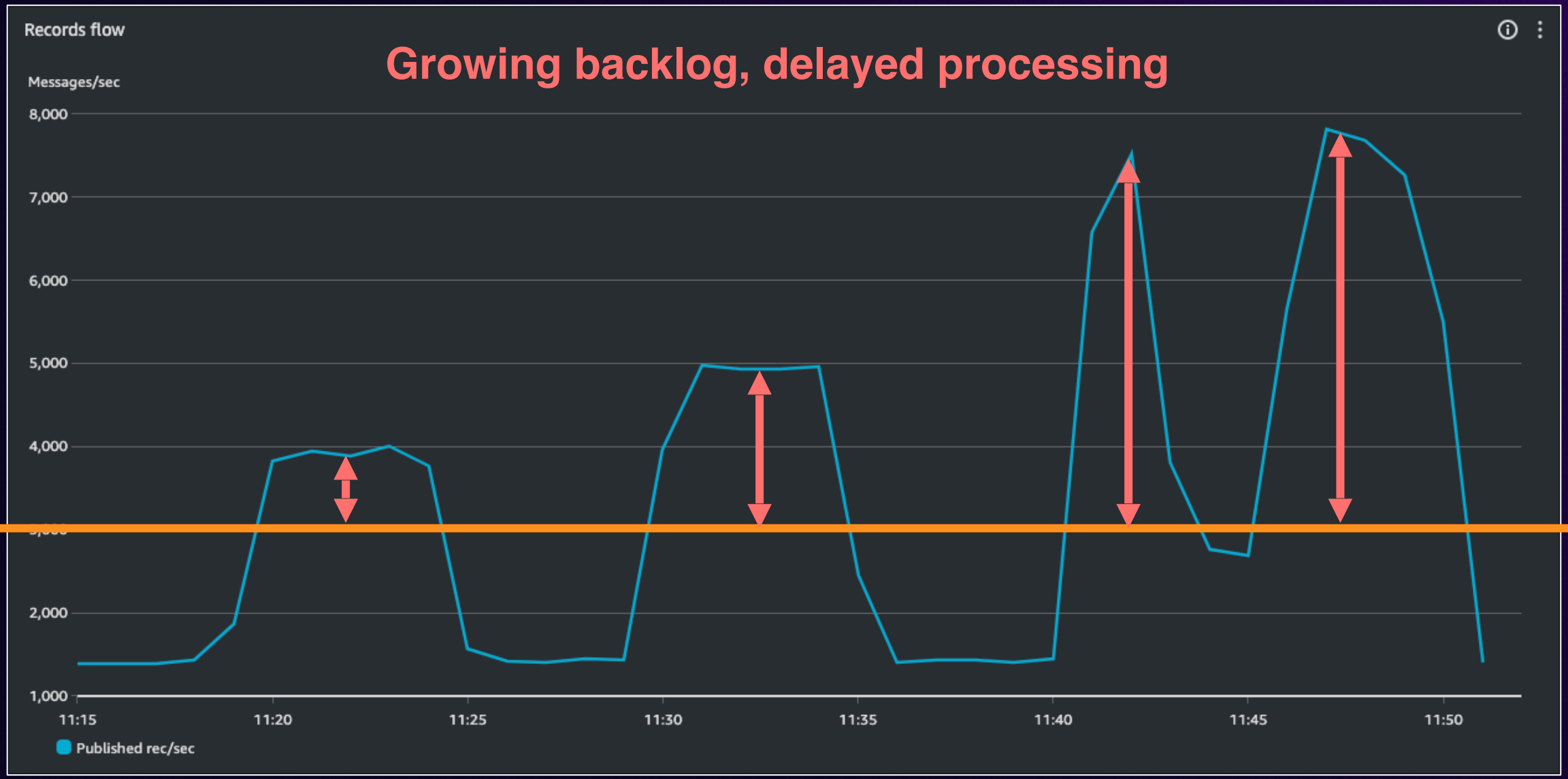


Spiky

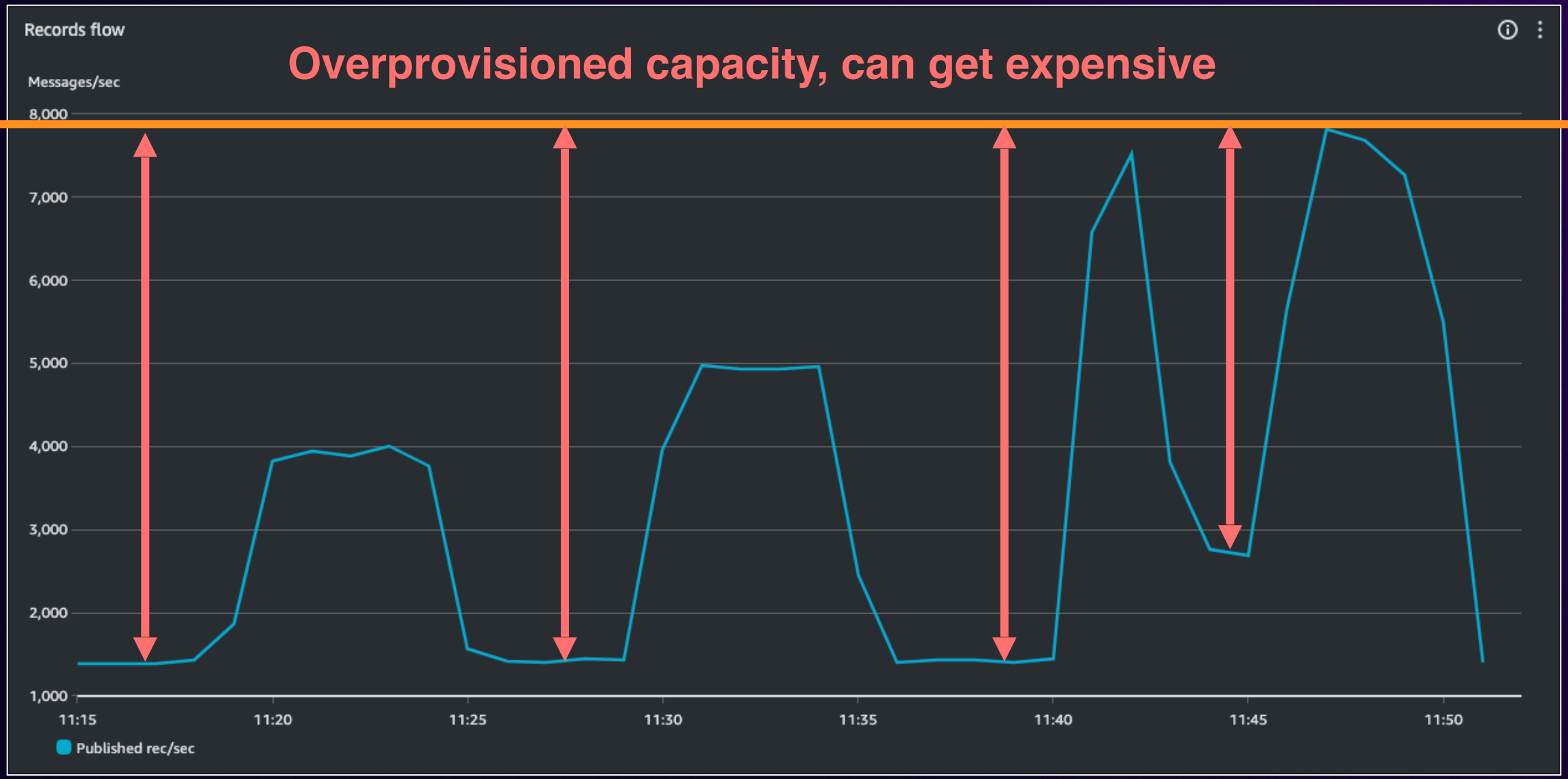
# Consistent workloads



# What is a spiky workload?



# What is a spiky workload?



# Serverless streaming on AWS

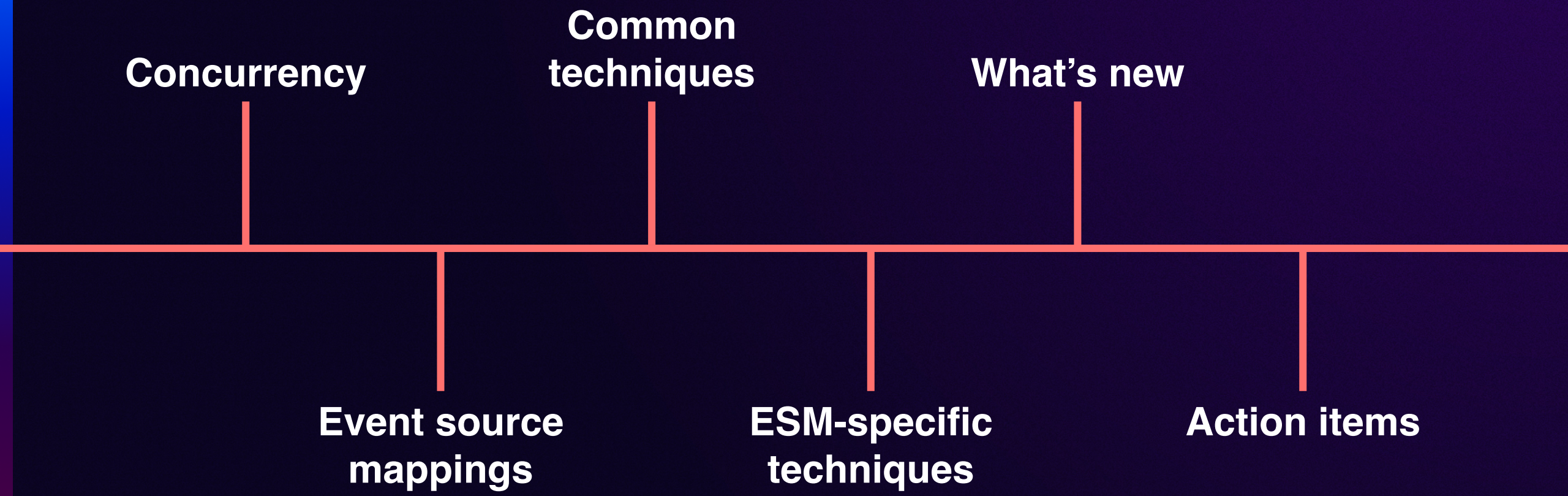
Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Apache Kafka

Amazon Kinesis Data Streams

AWS Lambda

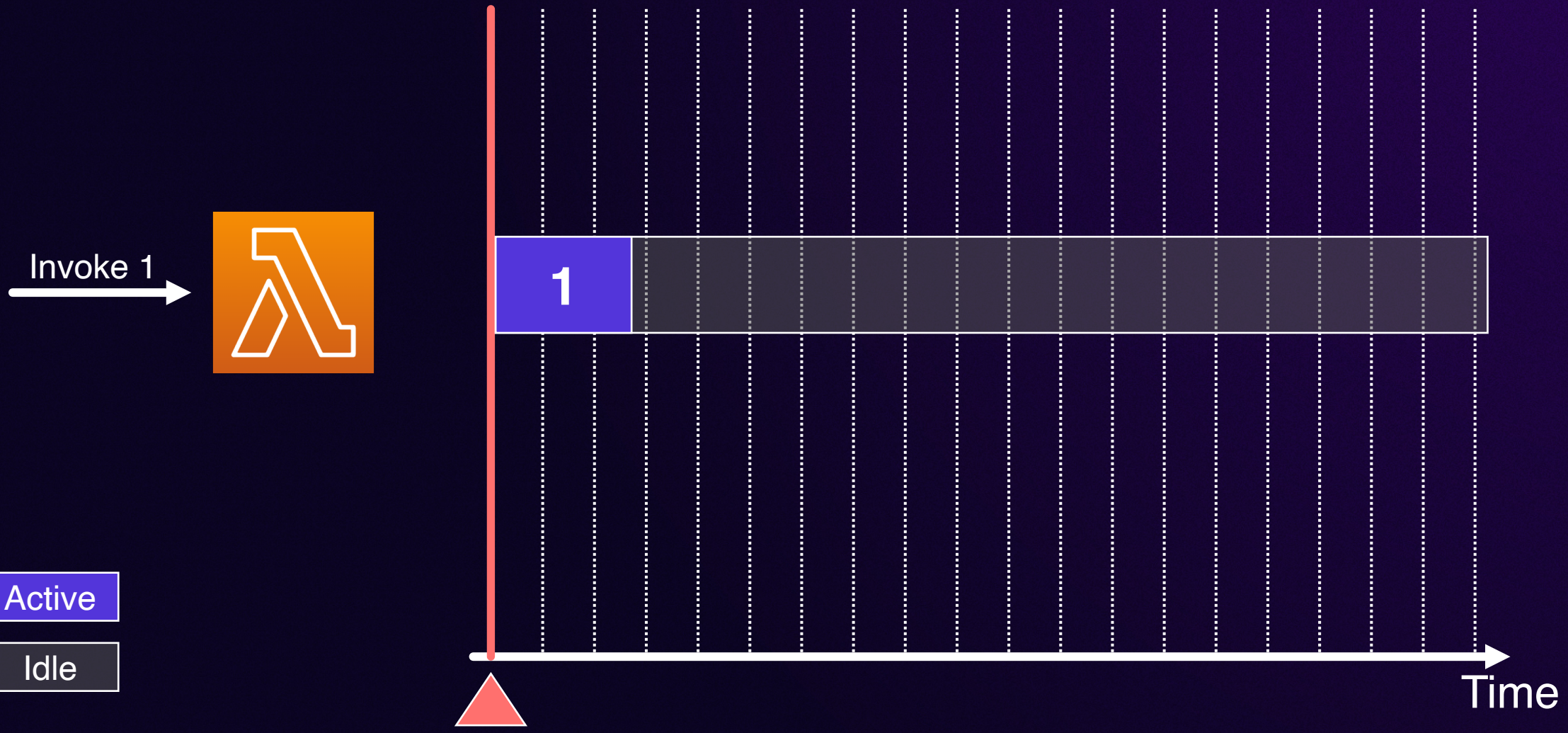
# Let’s talk about streaming data processing



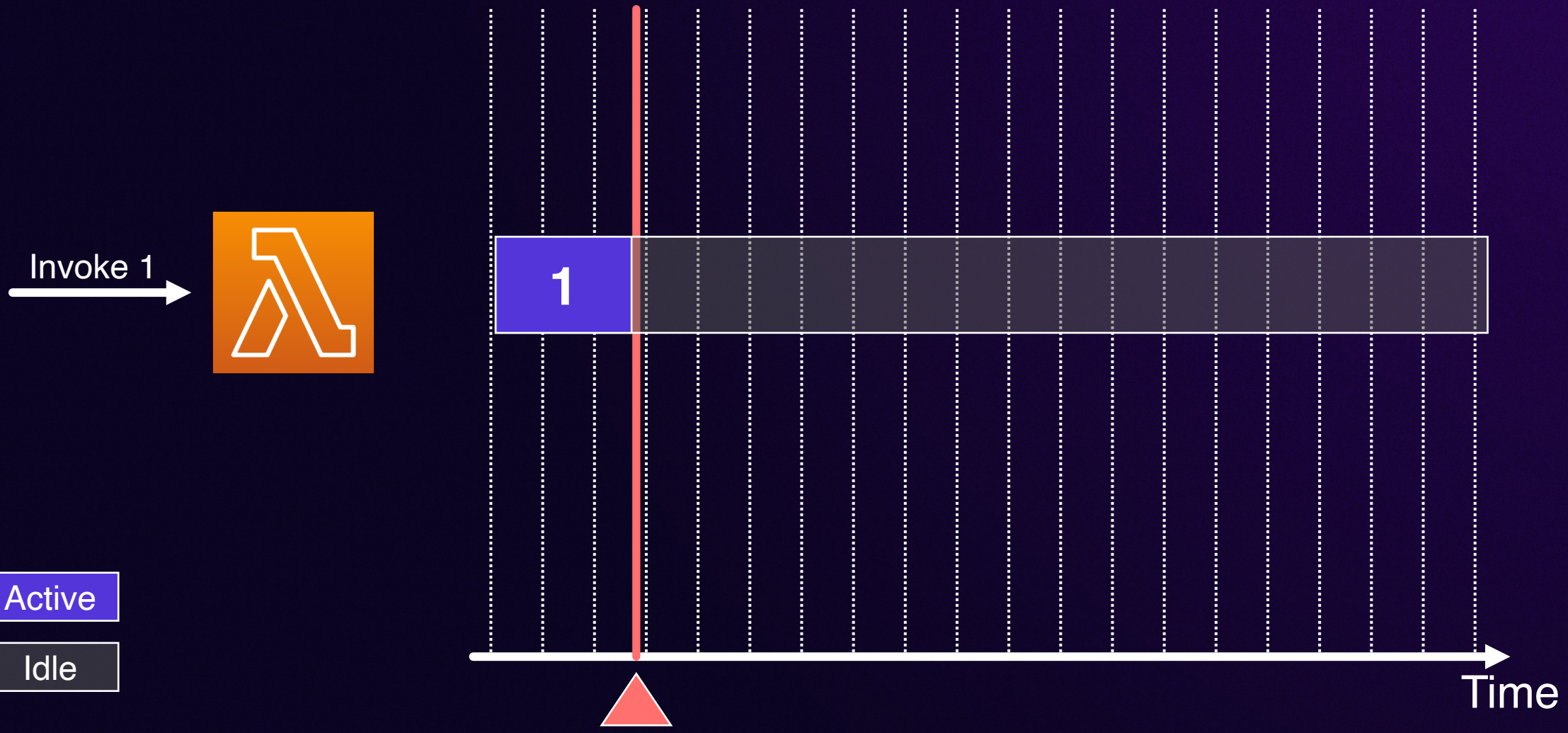
# Let’s dive deeper

# Understanding Lambda concurrency

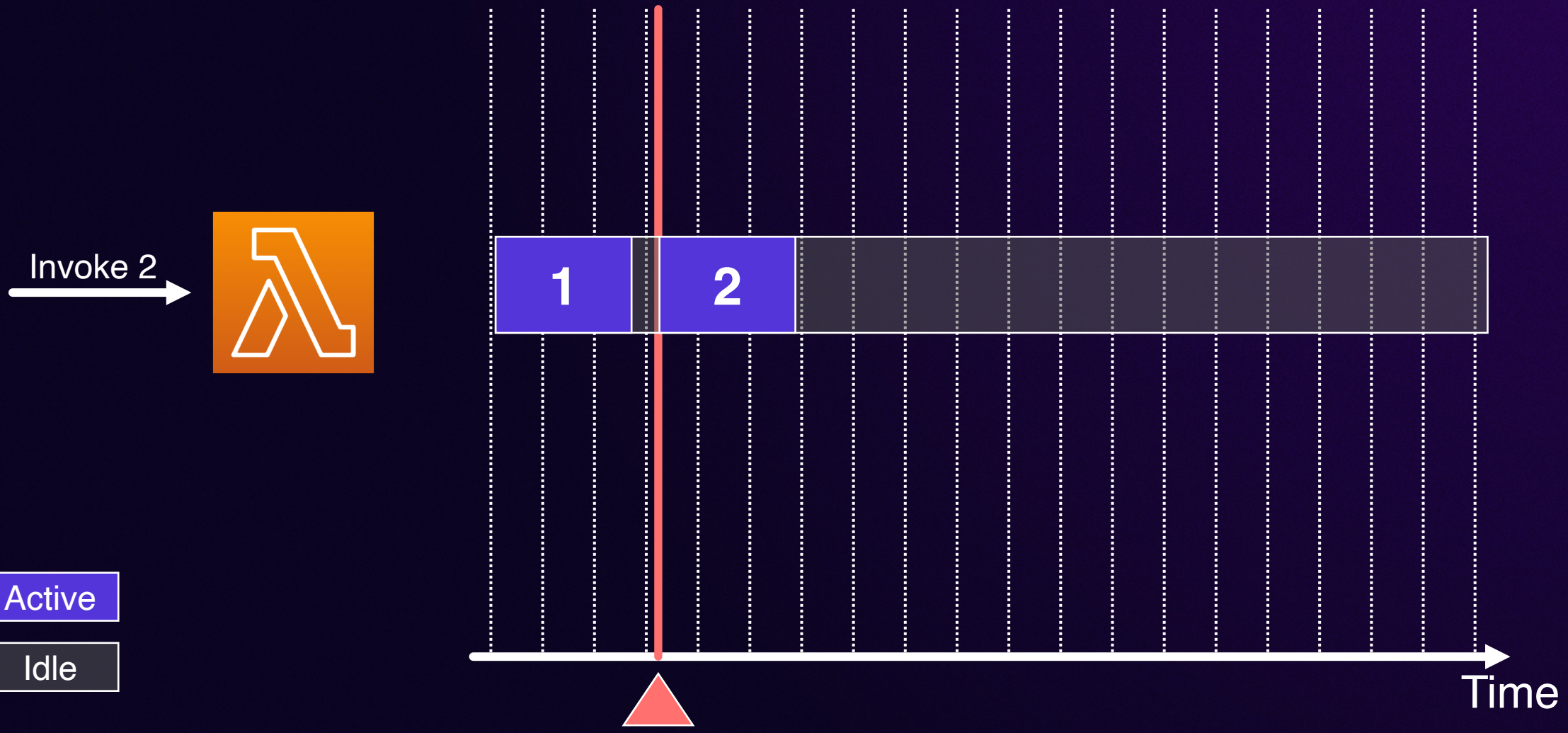
# Understanding Lambda function concurrency



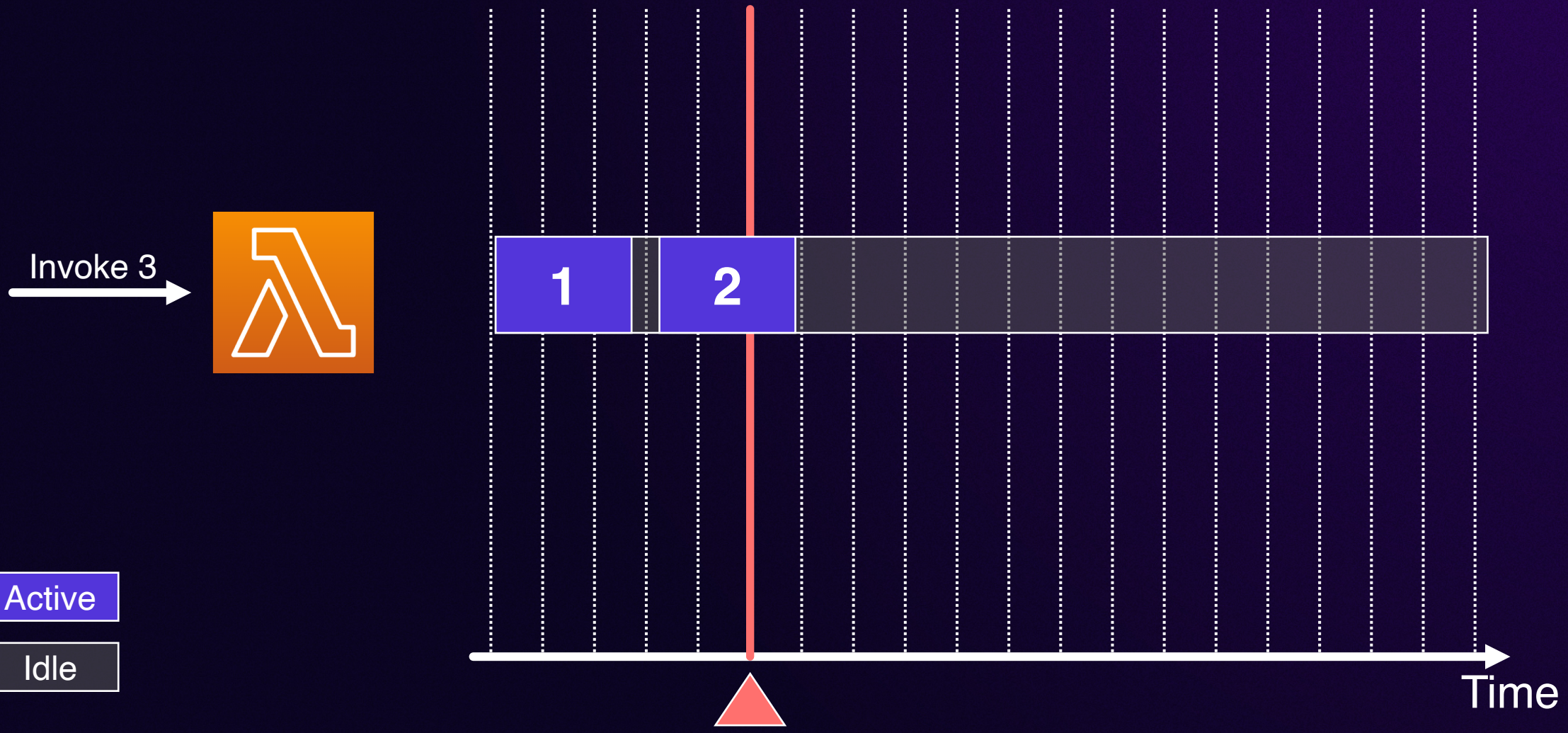
# Understanding Lambda function concurrency



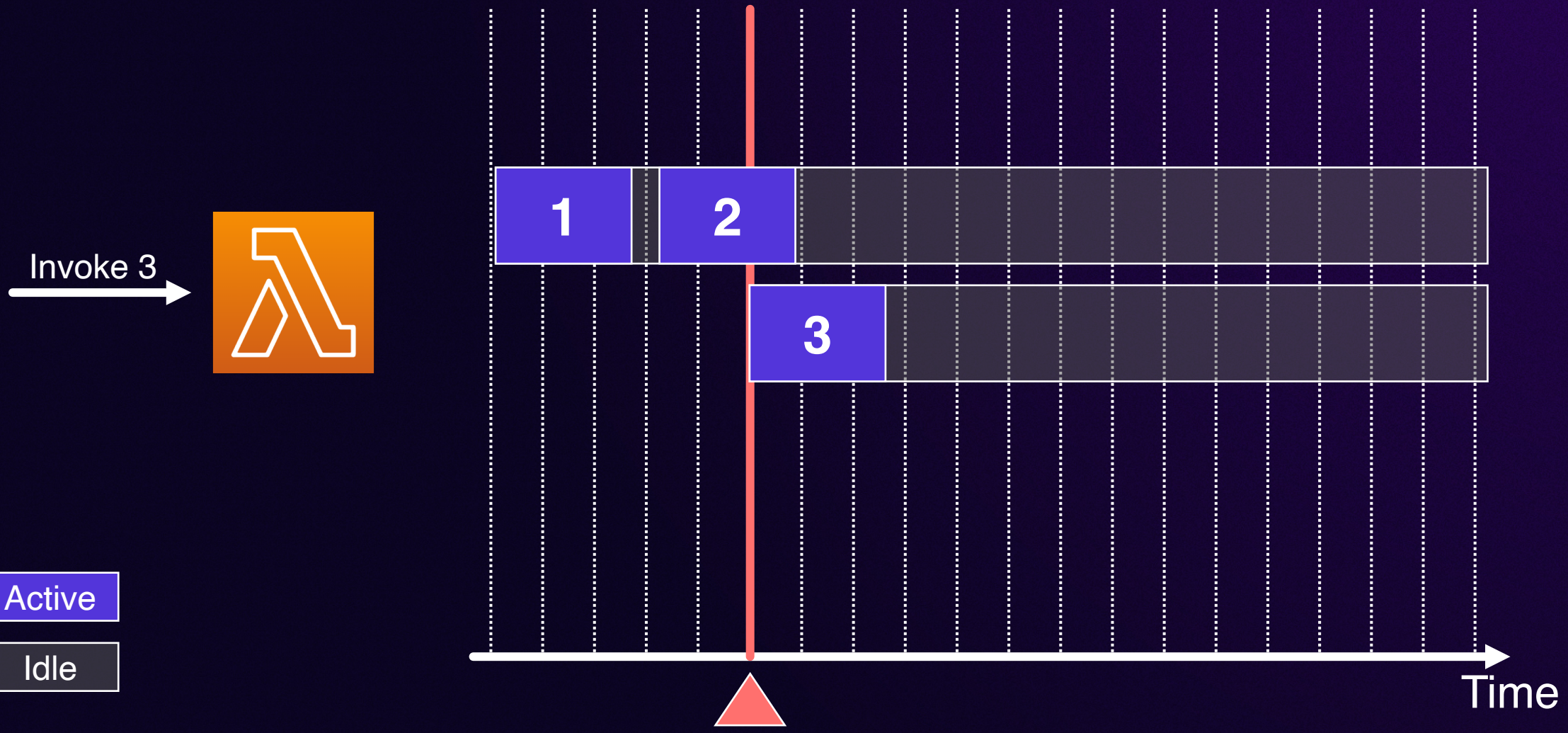
# Understanding Lambda function concurrency



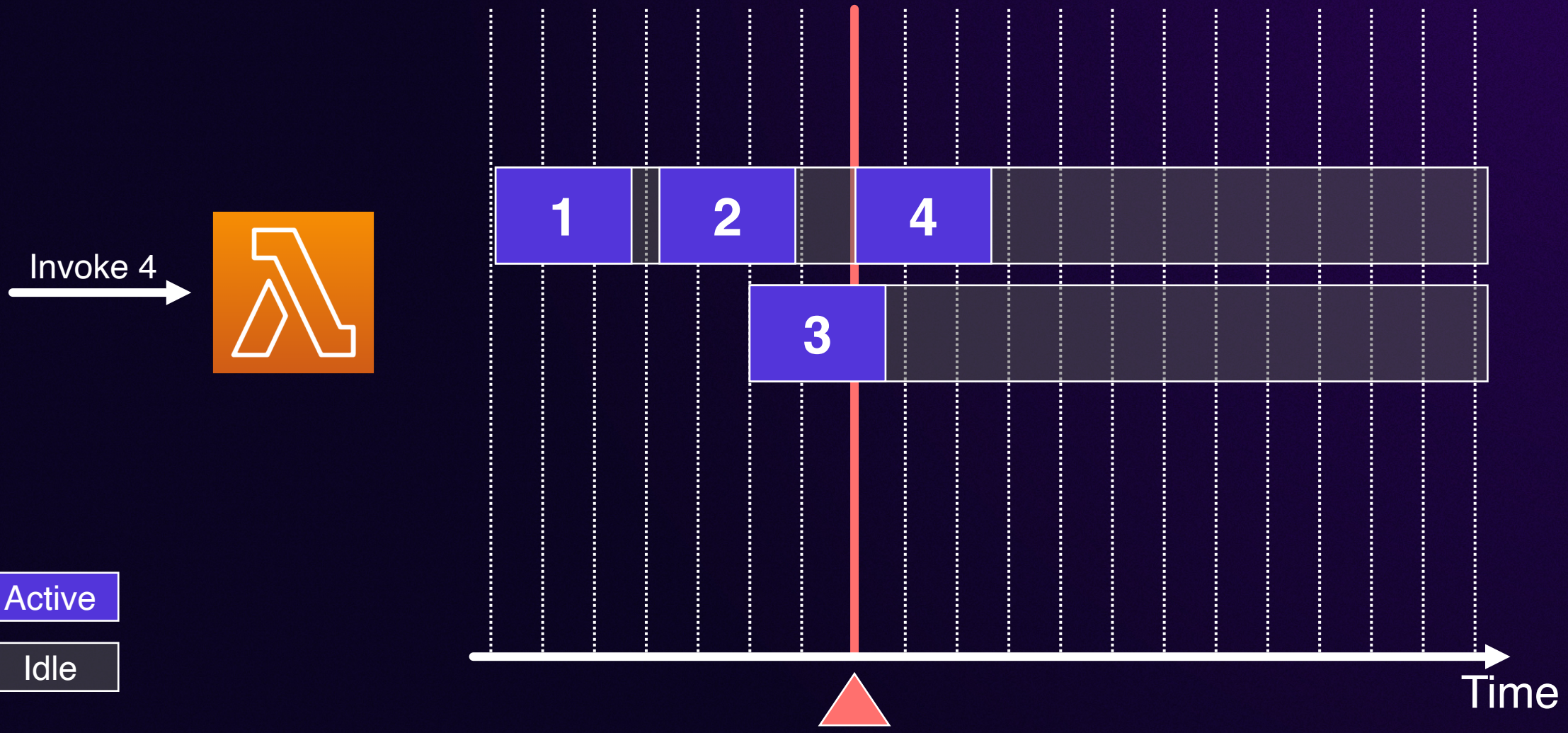
# Understanding Lambda function concurrency



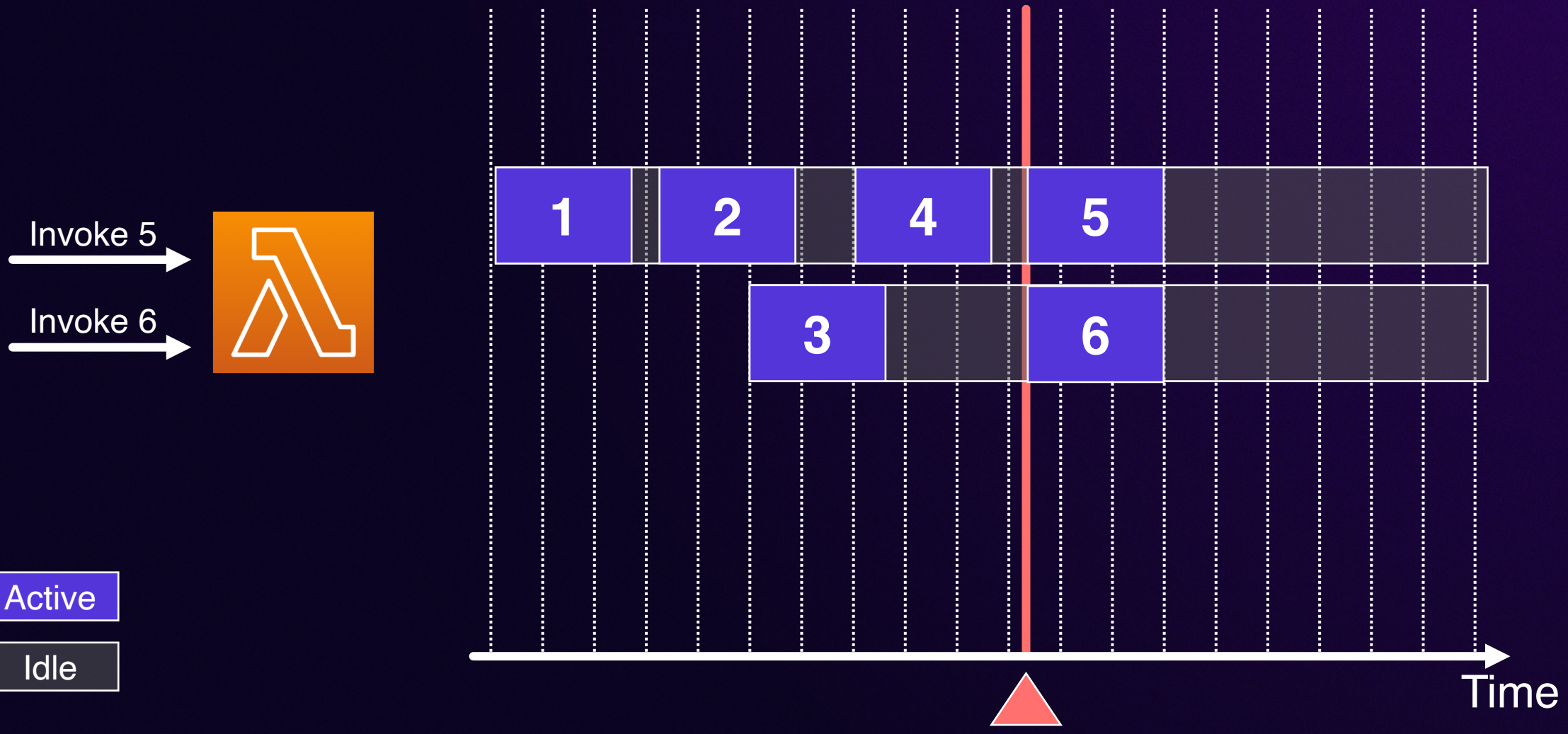
# Understanding Lambda function concurrency



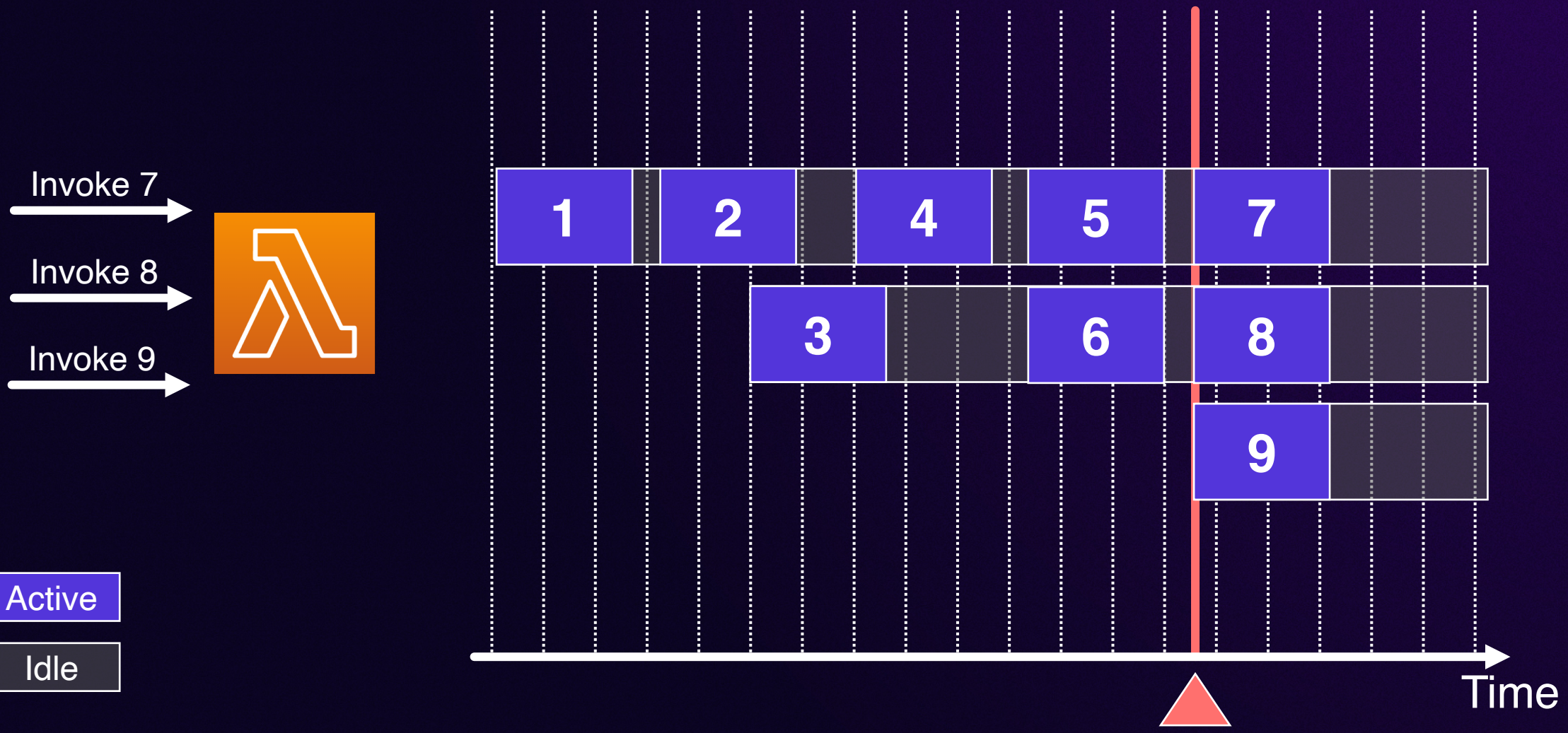
# Understanding Lambda function concurrency



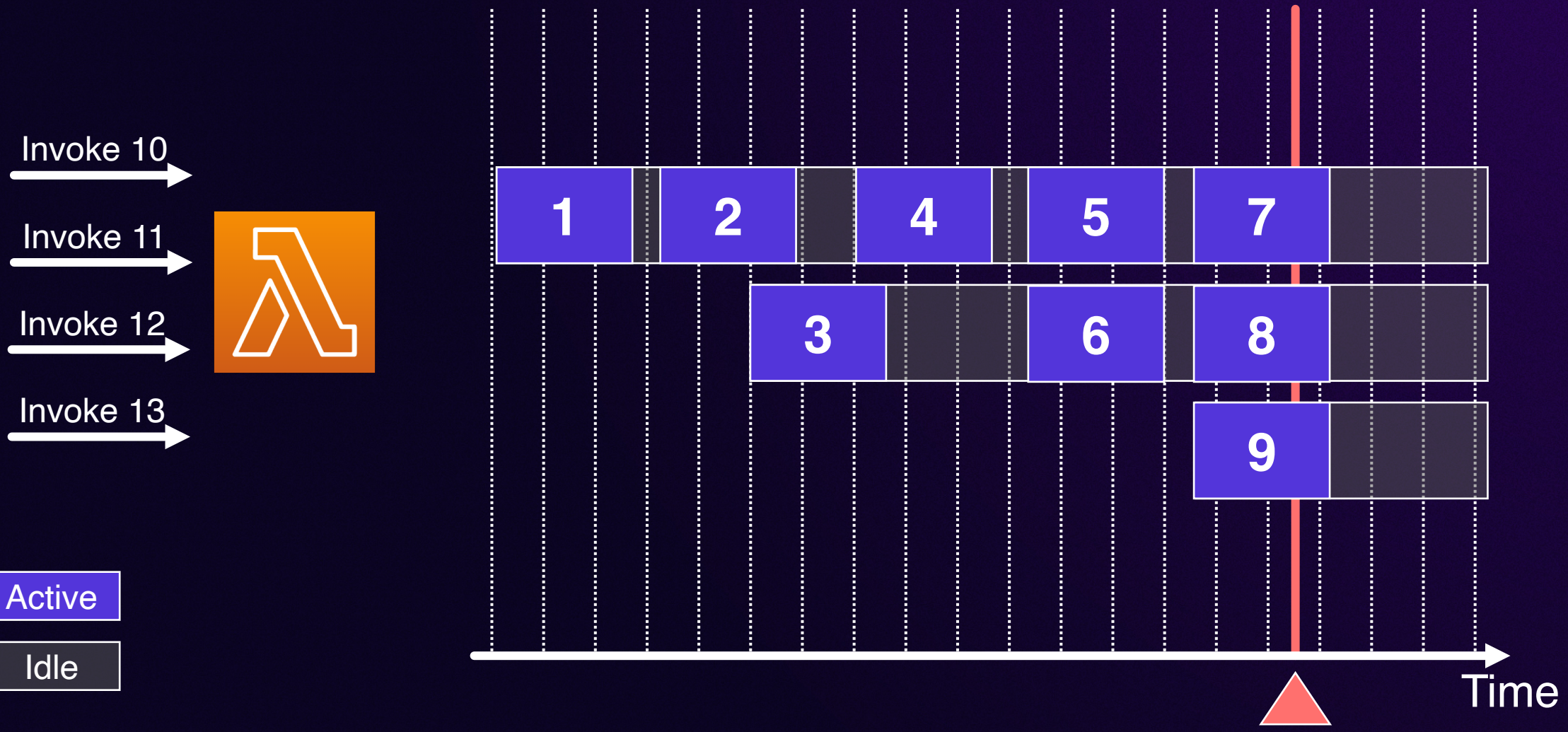
# Understanding Lambda function concurrency



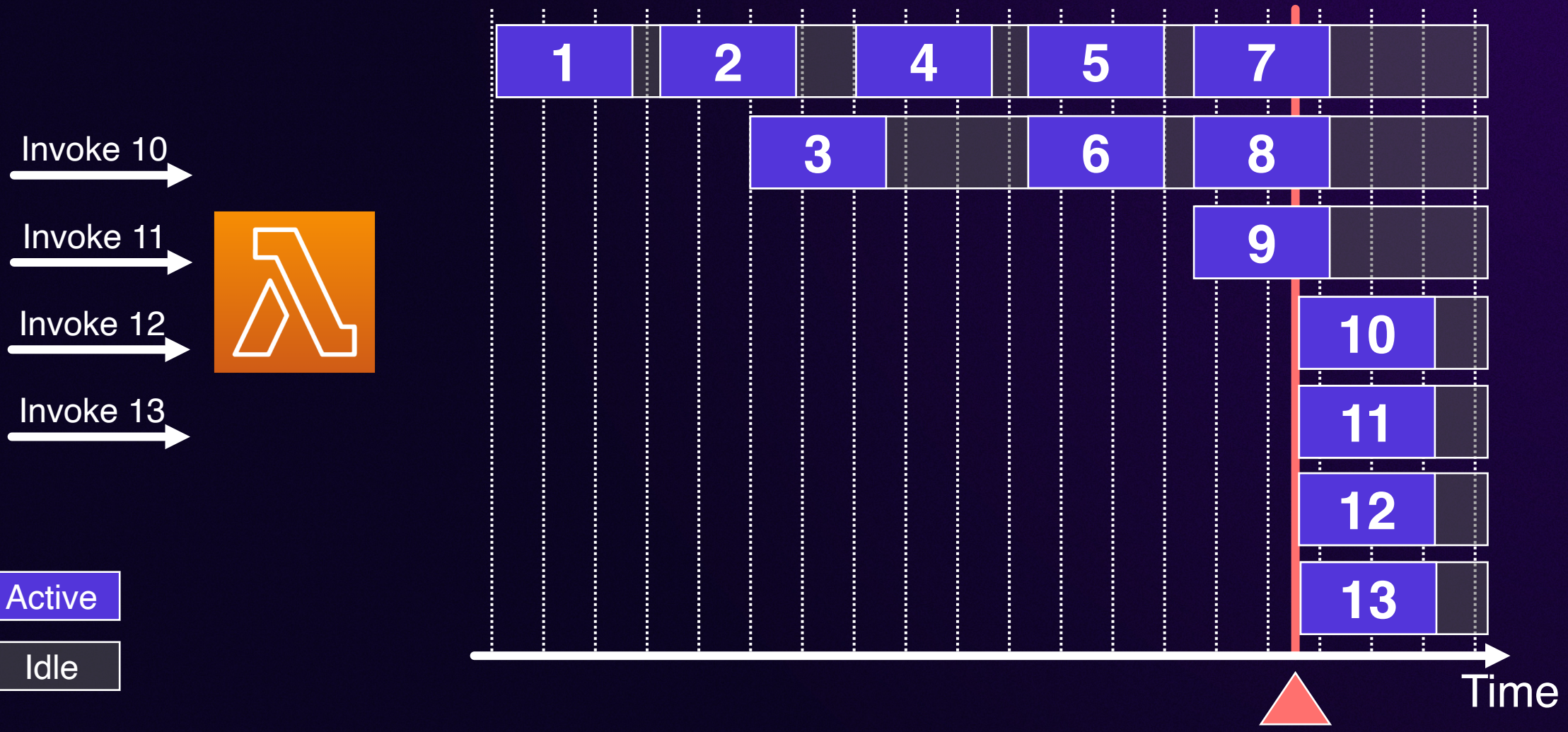
# Understanding Lambda function concurrency



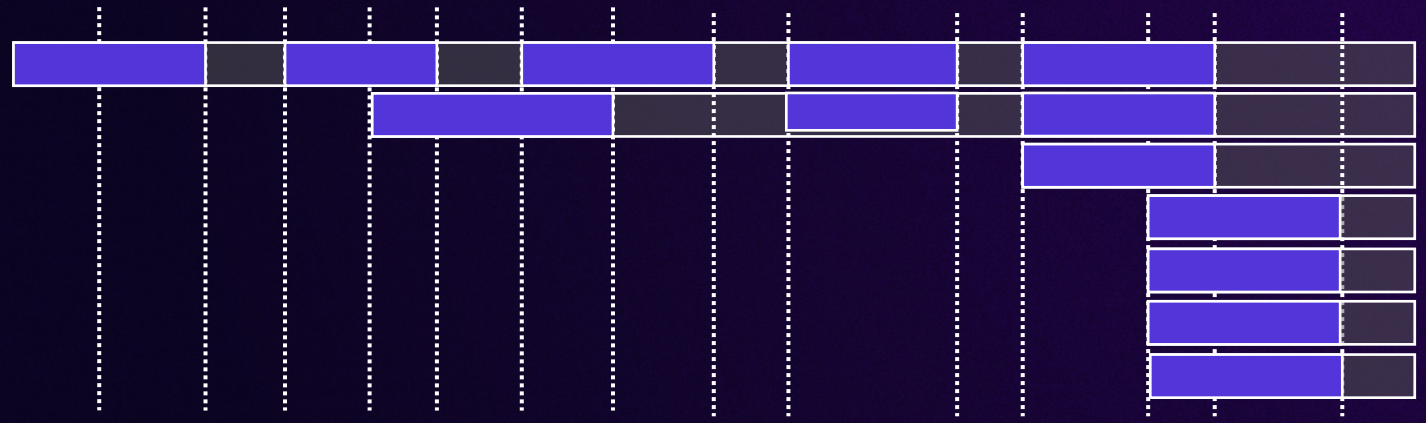
# Understanding Lambda function concurrency

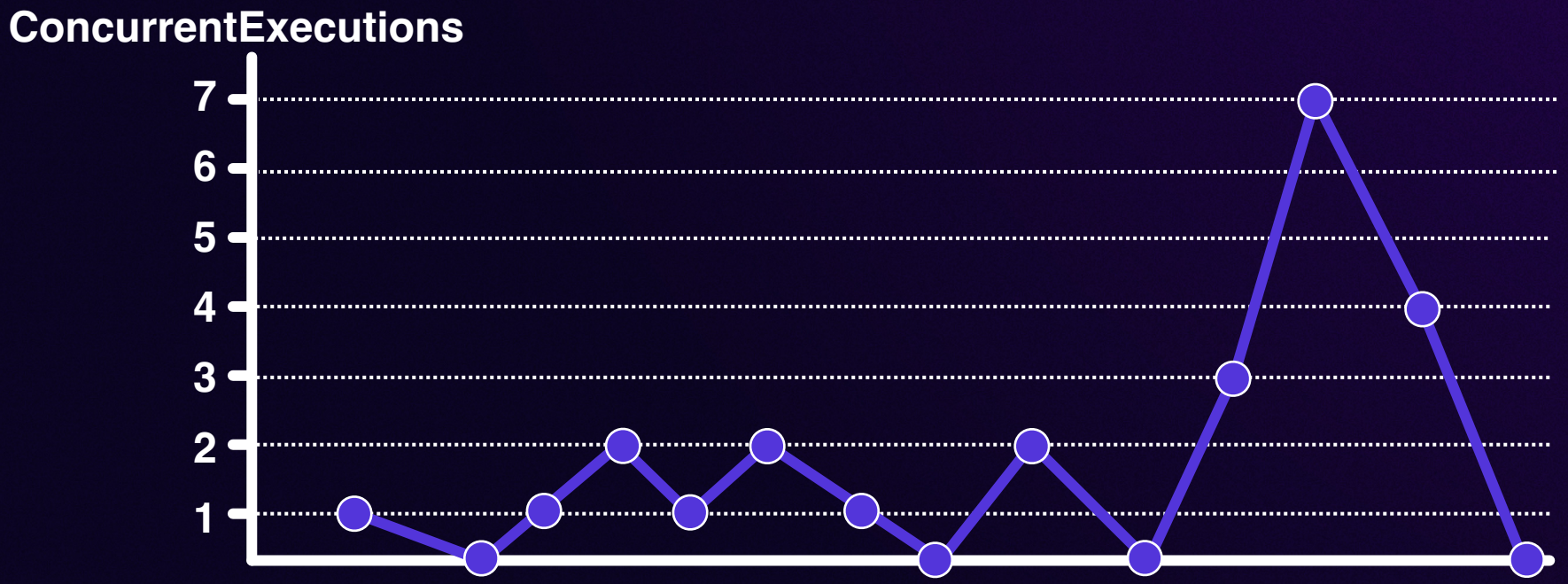


# Understanding Lambda function concurrency

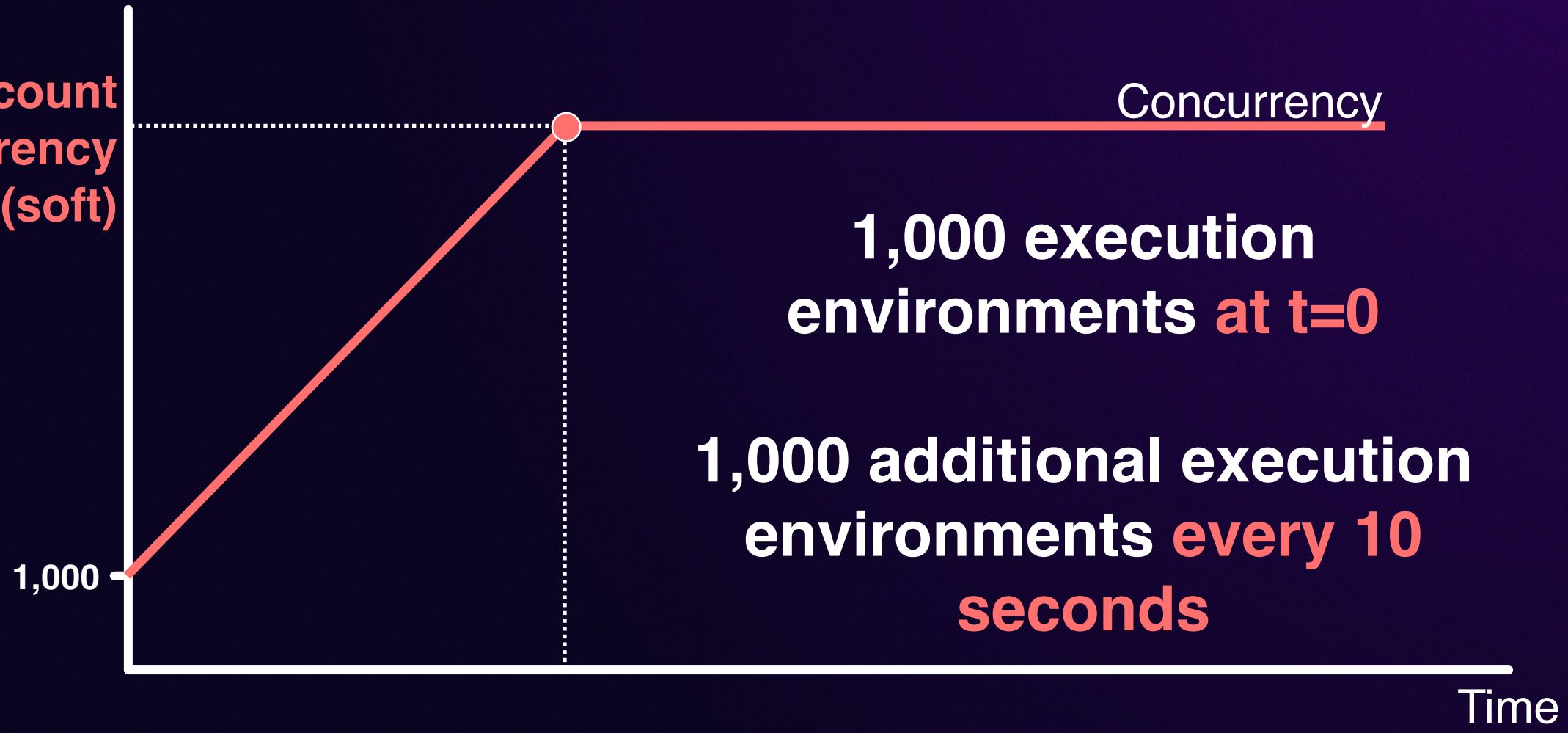


# Understanding Lambda function concurrency





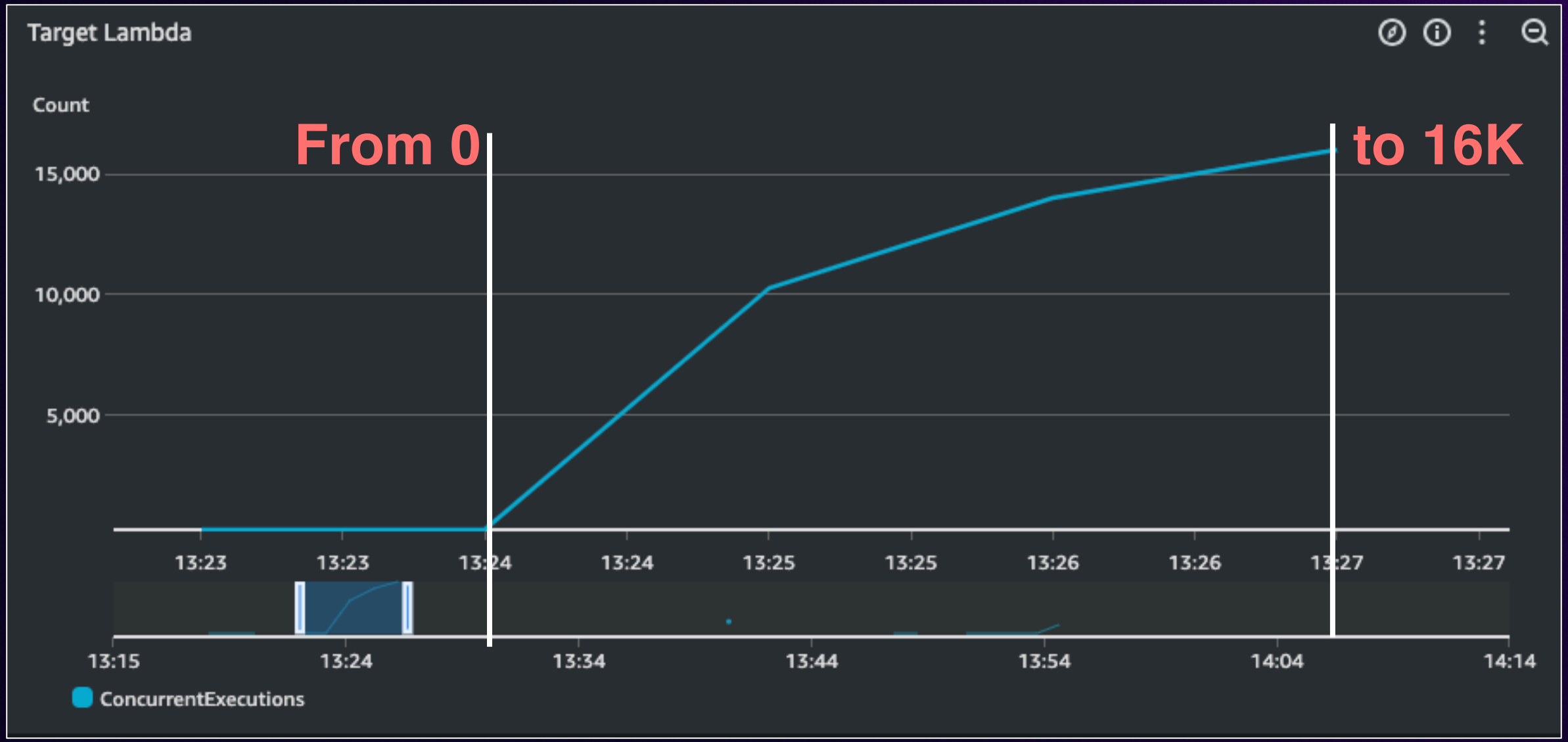
# Concurrency scaling rate – per function



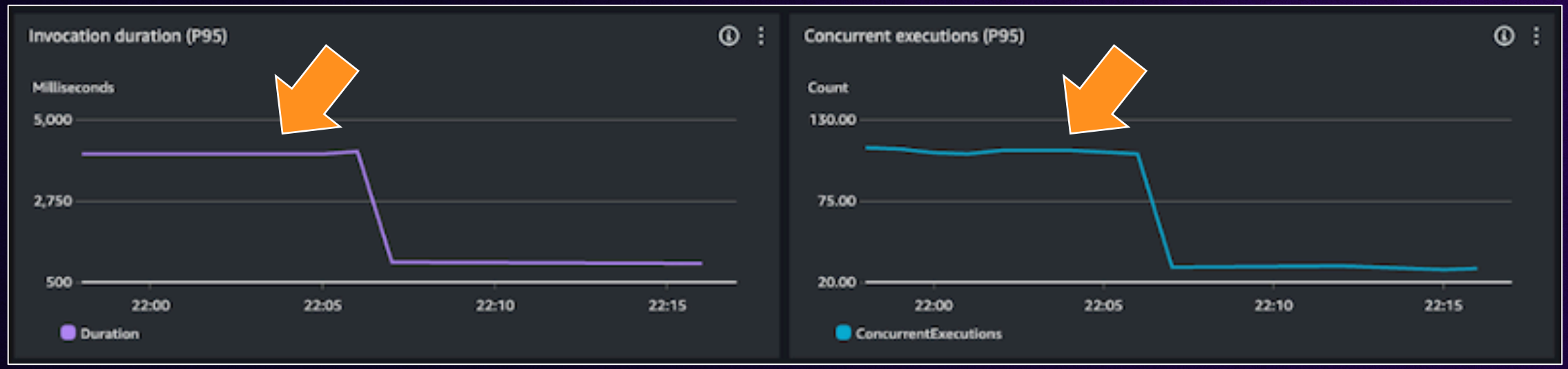

1,000 execution environments at t=0

1,000 additional execution environments every 10 seconds

# Concurrency scaling rate – per function



# Lambda monitoring




- Invocations
- Errors
- Throttles

- Duration
- ConcurrentExecutions
- ClaimedAccountConcurrency

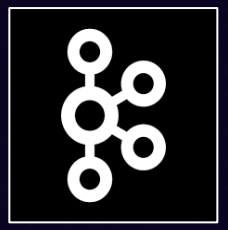


https://docs.aws.amazon.com/lambda/latest/dg/monitoring-metrics.html

# Event sources

Amazon Managed Streaming for Apache Kafka


Apache Kafka



Amazon Kinesis Data Streams

Invoke

Invoke

Invoke

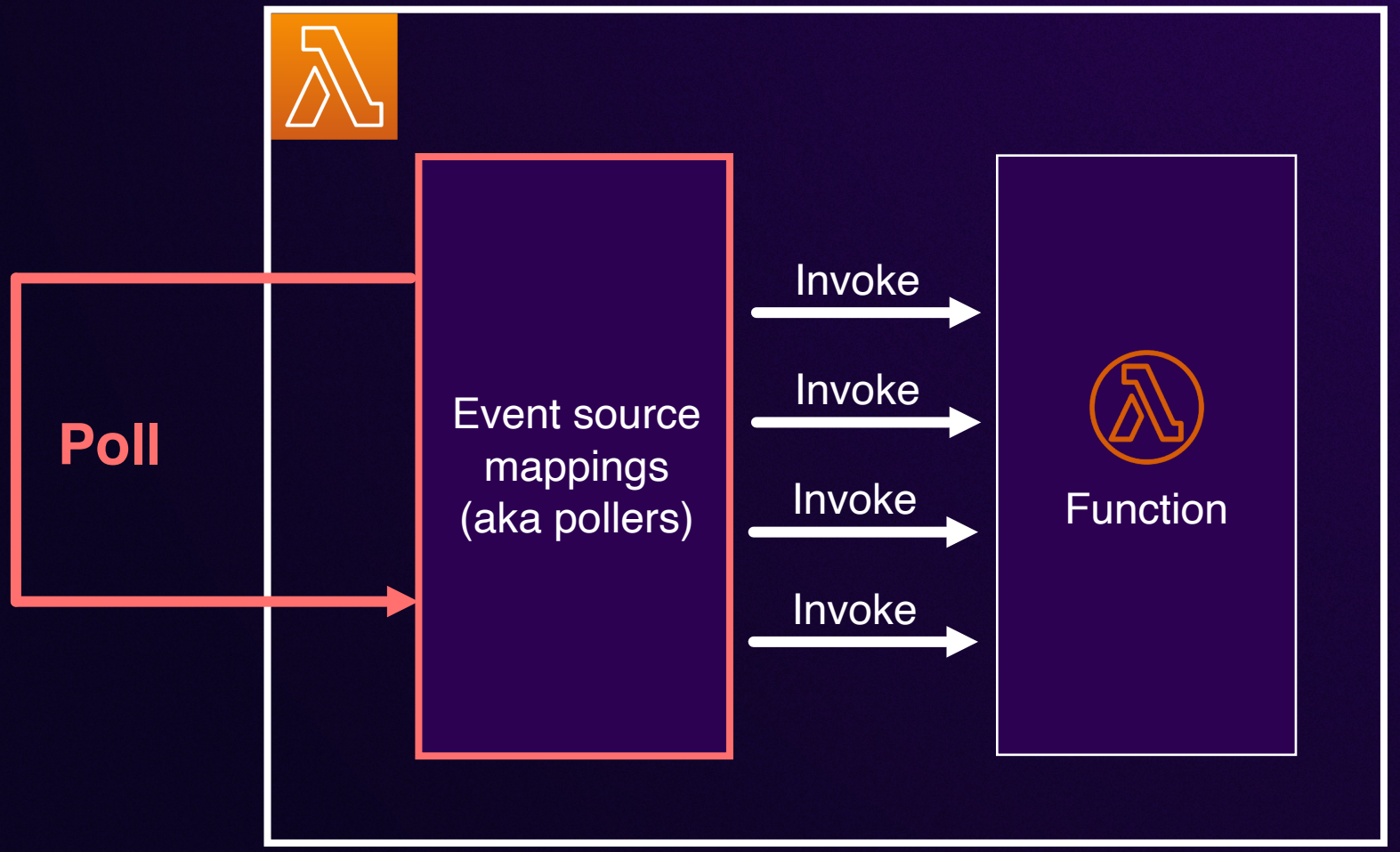
Invoke

# Event source mappings

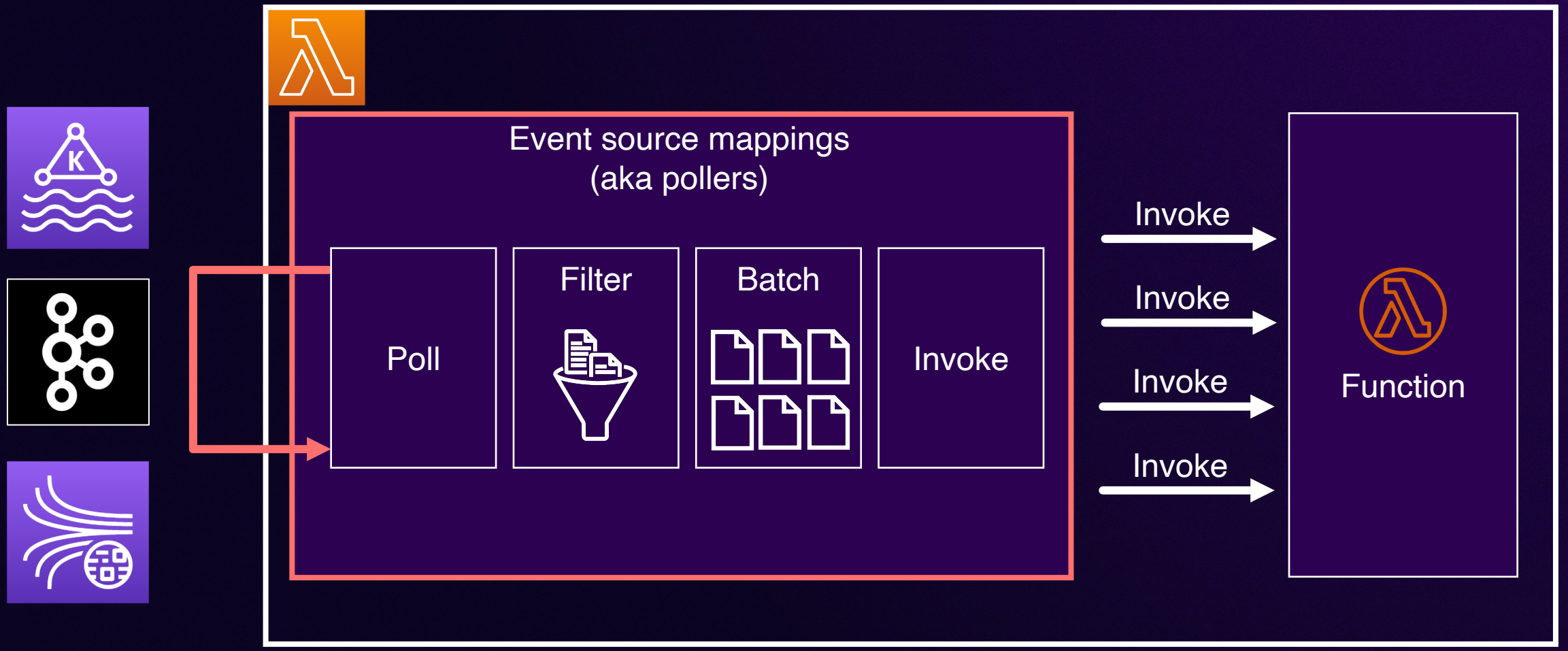
Amazon Managed Streaming for Apache Kafka

Apache Kafka

Amazon Kinesis Data Streams

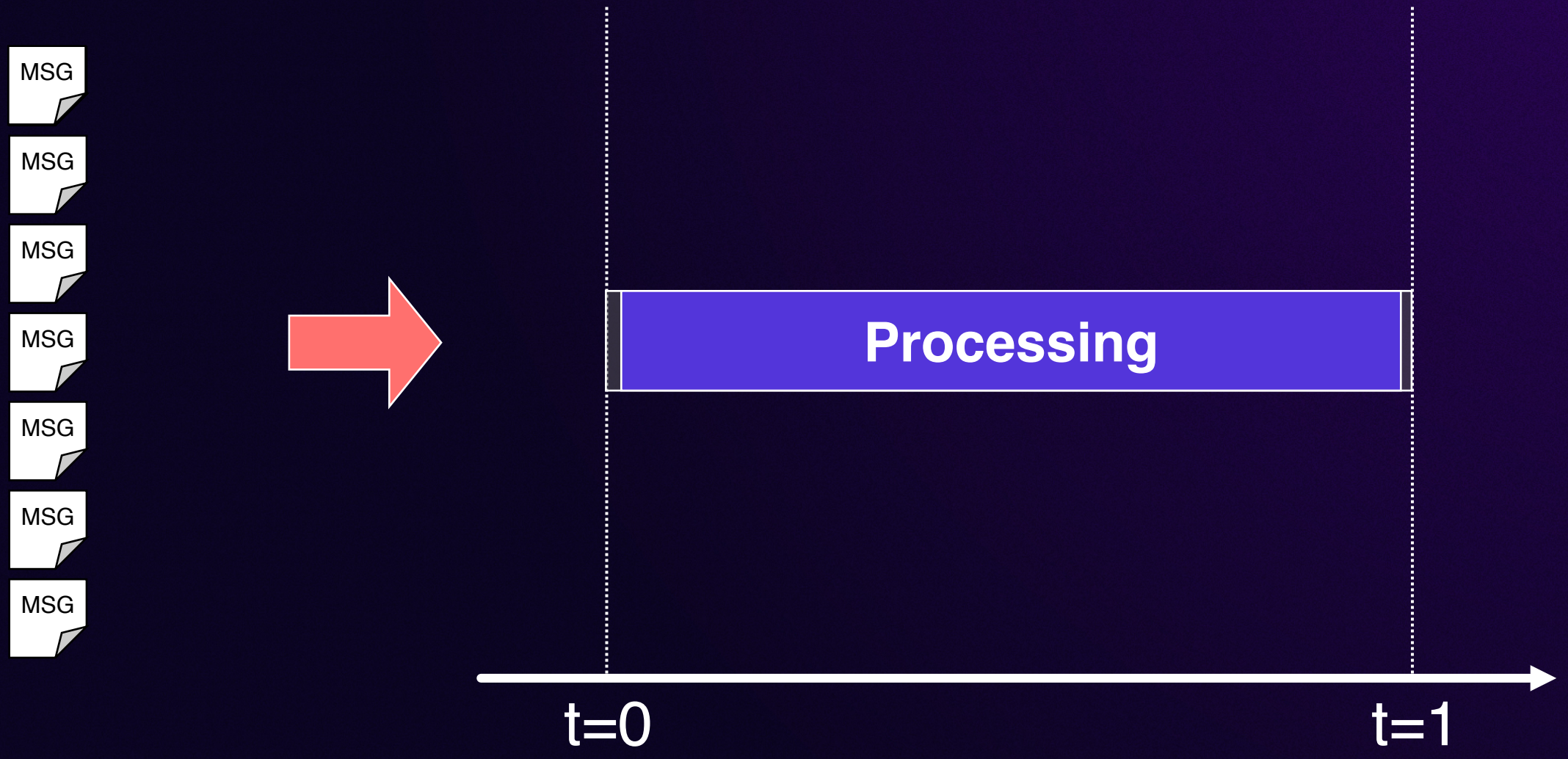


# Event source mappings

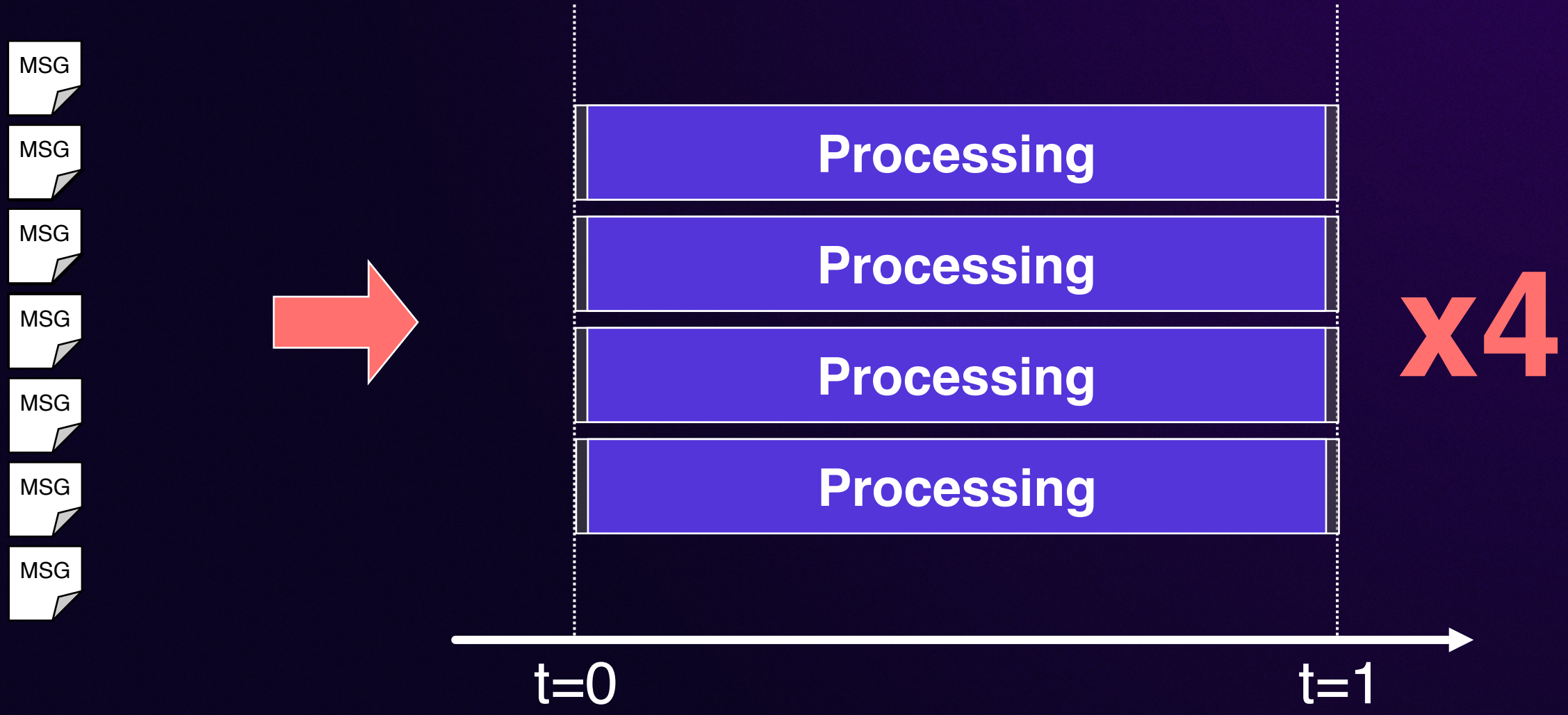


# Common techniques

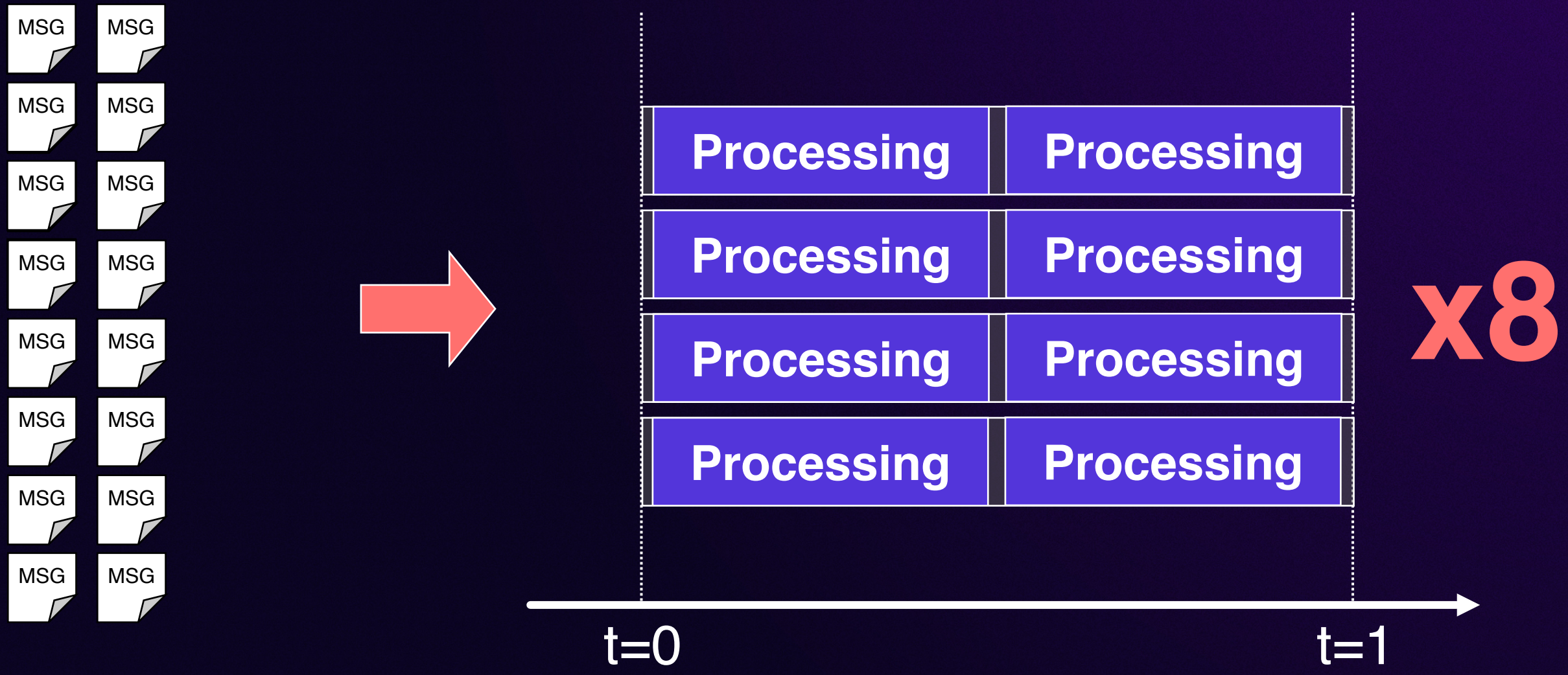
# Improving throughput



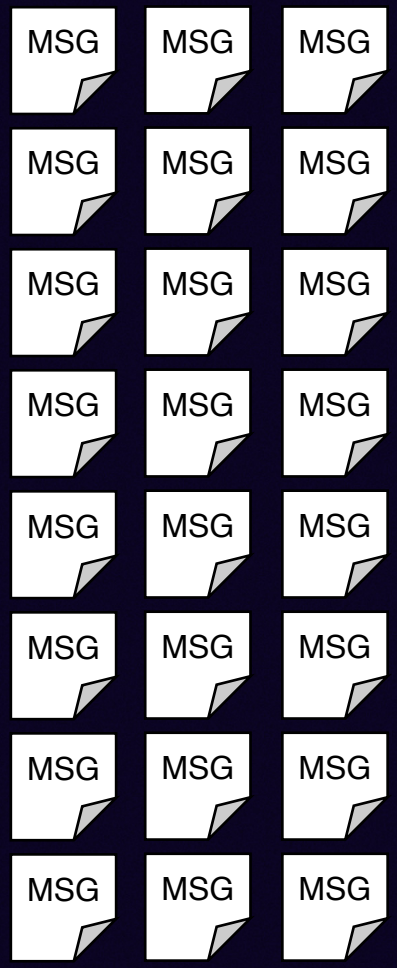
# Improving throughput - concurrency

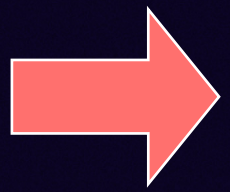


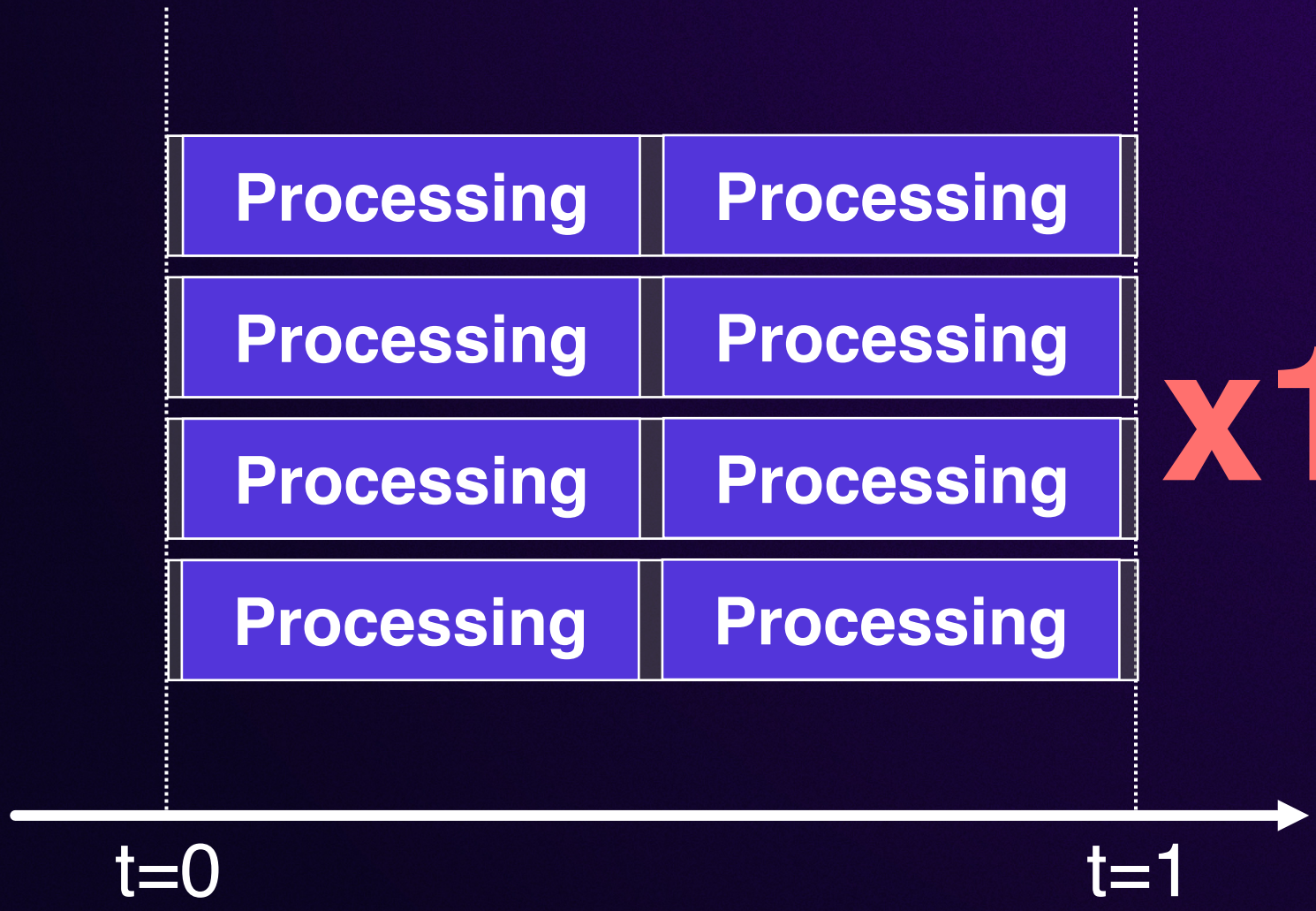
# Improving throughput - duration



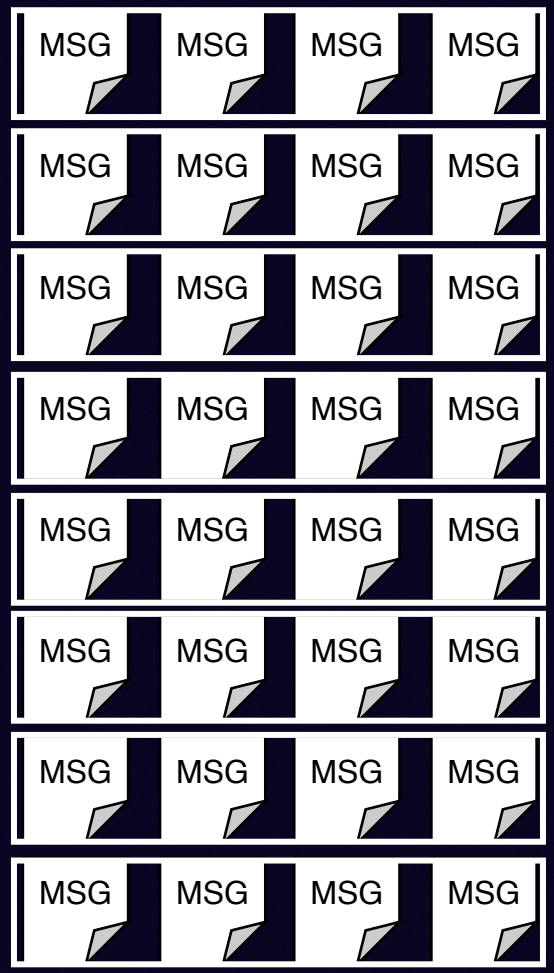
# Improving throughput - filtering

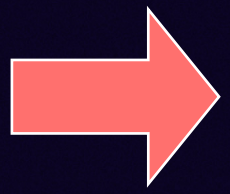


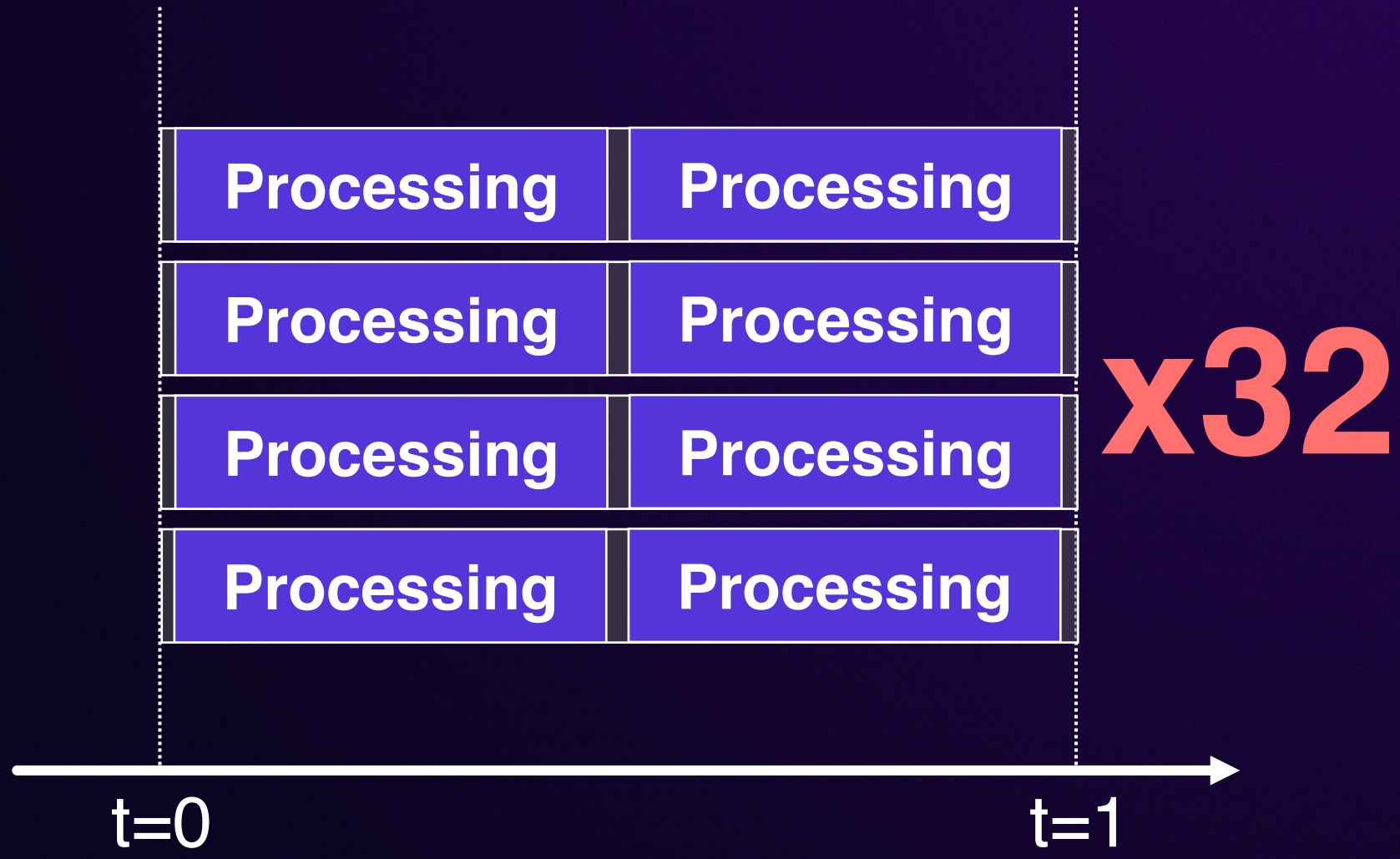





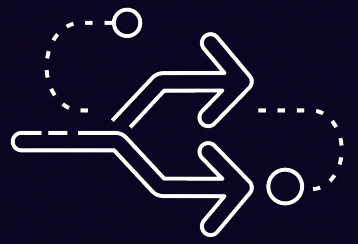
# Improving throughput - batching



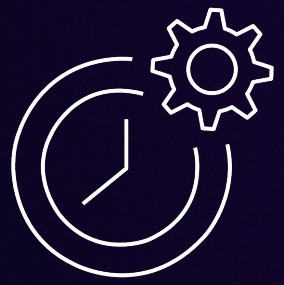





# Improving throughput



**Parallelize data processing**

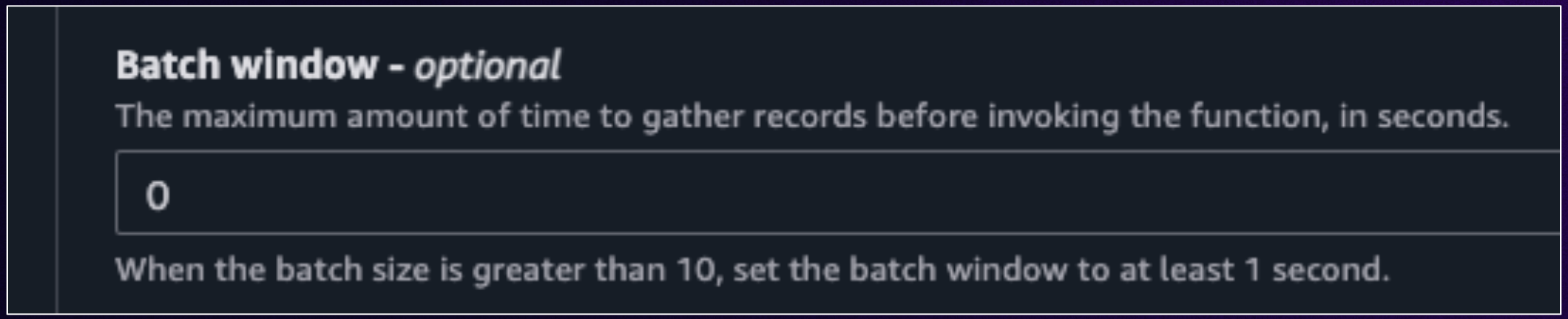


**Reduce processing duration**

**Filter irrelevant messages out**

**Batch messages**

# Event source mapping - polling




**Short polling**
**(0 seconds)**

**Long polling**
**(up to 300 seconds)**

# Event source mapping - filtering

```
if (message.data['fleet_id']==='fleet-452' && message.data['tire_pressure'<32]){
        processMessage(message);
} else {
    // Do nothing
}
```

**Filter criteria**

Define the filtering criteria to determine whether or not to process an event. Each filter must be in a valid JSON format. Lambda processes an event if any one of the filters are met. Otherwise, Lambda discards the event. **Learn more**.

```
{
   "data": {
      "fleet_id": ["fleet-452"],
      "tire_pressure": [{"numeric": ["<", 32]}]
   }
}
```

Remove

# Event source mapping - filtering

- 10,000 IoT sensors, emitting a telemetry message every minute
- Total **~450M messages/month**
- Lambda function with 256 MB, average duration 300ms, 50ms when doing nothing
- **~2.2% of messages result in action**

| | Without filtering | With filtering |
|---|---|---|
| Total messages to process | 450M | 10M |
| **Total charge for requests** | **$90** | **$2** |
| Actionable messages | 10M | 10M |
| Irrelevant messages | 440M | 0 |
| Processing compute duration | 25M milliseconds | 3M milliseconds |
| **Total compute cost** | **$200** | **$15** |

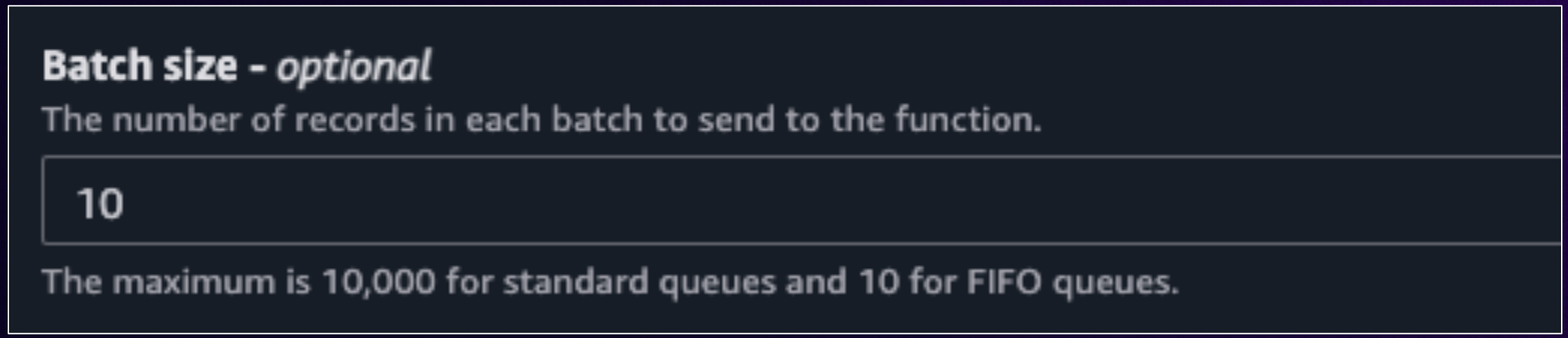
# Event source mapping - filtering

- 10,000 IoT sensors, emitting a telemetry message every minute
- Total **~450M messages/month**
- Lambda function with 256 MB, average duration 300ms, 50ms when doing nothing
- **~2.2% of messages result in action**

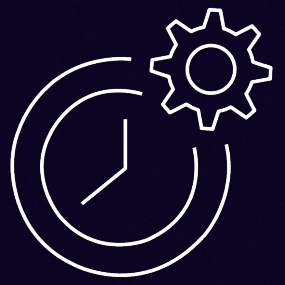
| | Without filtering | With filtering |
|---|---|---|
| Total messages to process | 450M | 10M |
| **Total charge for requests** | **$90** | **$2** |
| Actionable messages | 10M | 10M |
| Irrelevant messages | 440M | 0 |
| Processing compute duration | 6,250,000 seconds | 500,000 milliseconds |
| **Total compute cost** | **$21.60** | **$1.65** |

**92% cost reduction**

# Event source mapping - batching

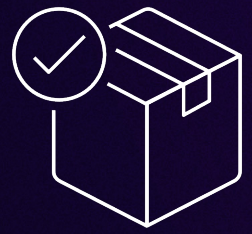


# Event source mapping - invoker



**Batching window has elapsed**

or



**Batch is full**

or

**Batch size is 6MB**

# Event source mapping - IaC

```
resource "aws_lambda_event_source_mapping" "example" {
  event_source_arn  = aws_msk_cluster.example.arn
  function_name     = aws_lambda_function.example.arn

  batch_size                         = 100
  maximum_batching_window_in_seconds = 20

  filter_criteria {
    filter {
      pattern = jsonencode({
        body = {
          Temperature : [{ numeric : [">", 0, "<=", 100] }]
          Location : ["New York"]
        }
      })
    }
  }
}
```

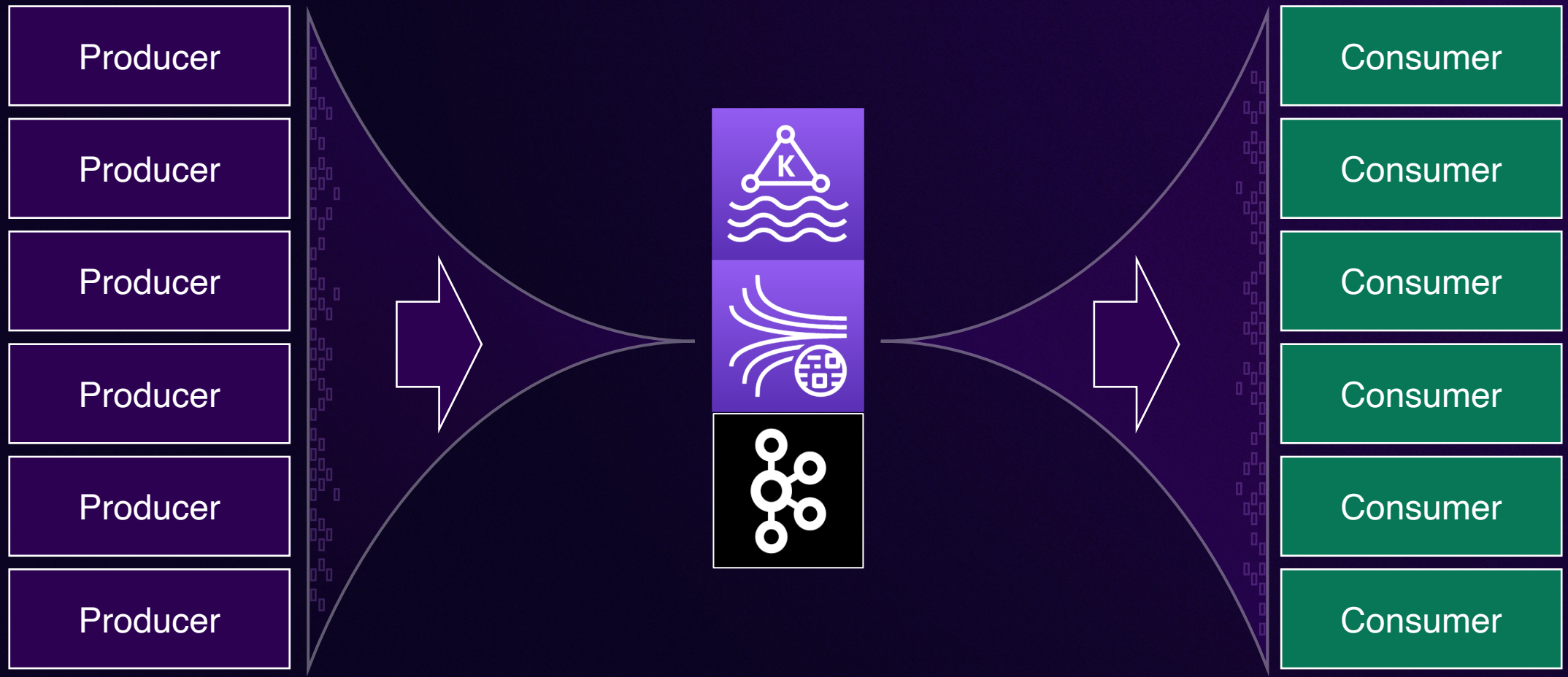
Event source

Batch size and window

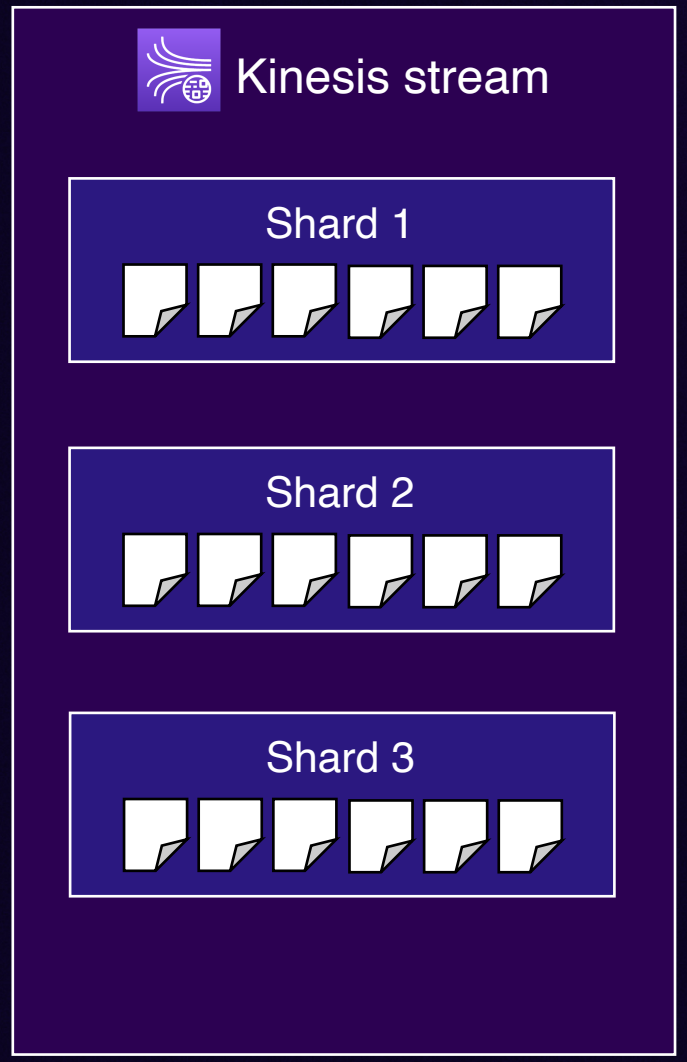
Filter criteria

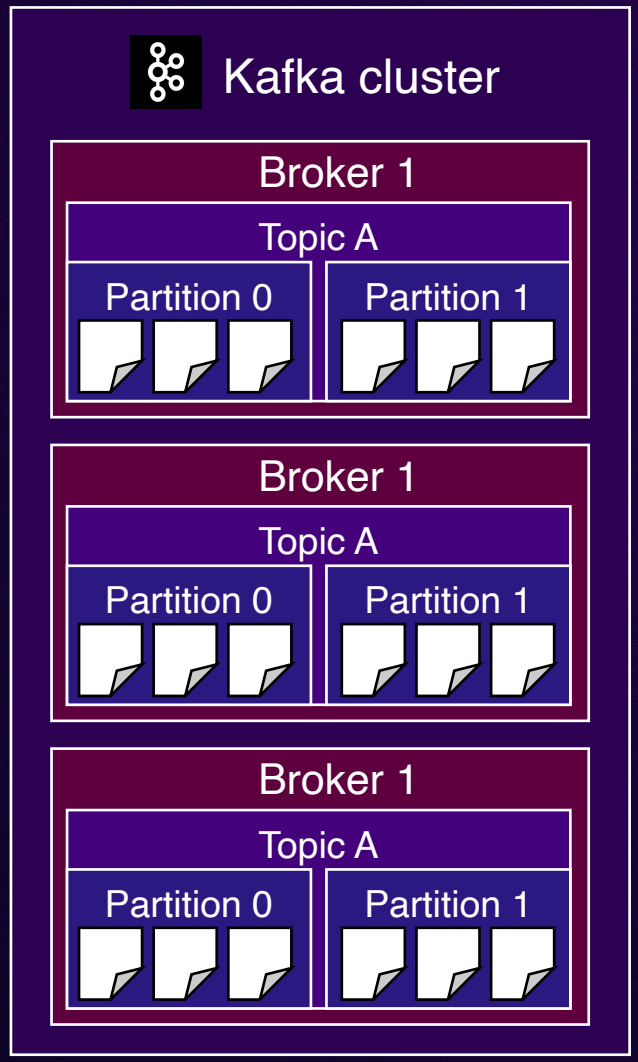
# Event source specific techniques

# Streaming event source types



# Streaming event source types



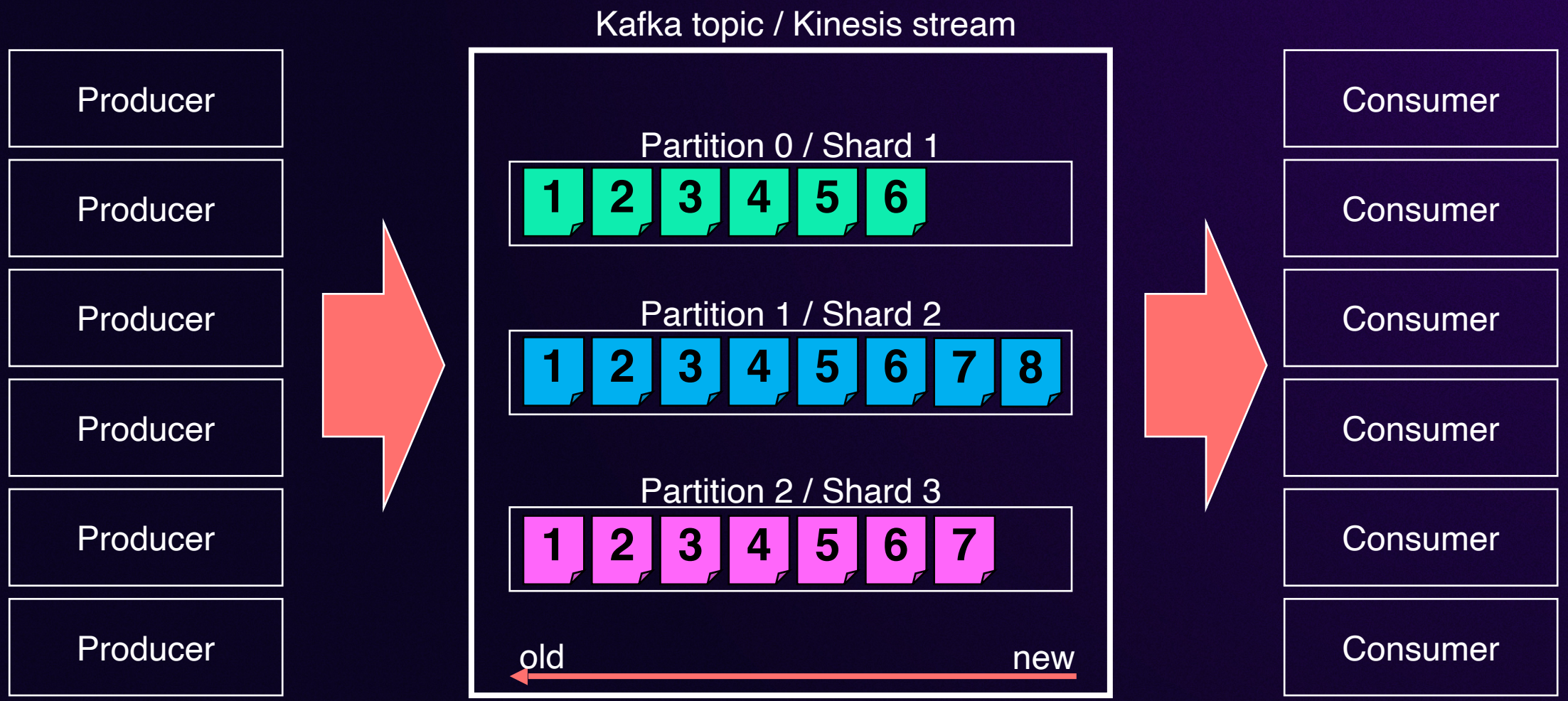



# Terminology

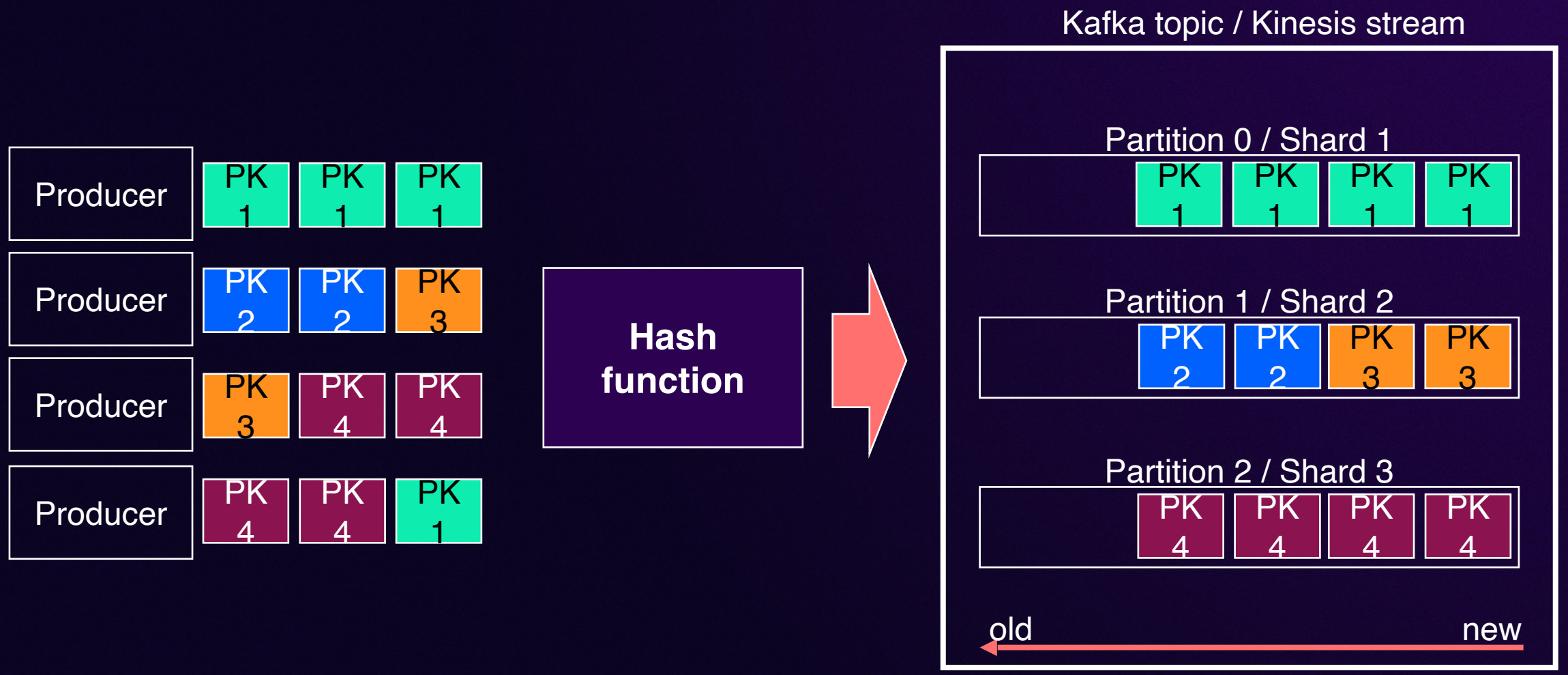
| Kinesis | Kafka |
| --- | --- |
| Stream | Topic |
| Shard | Partition |
| Iterator Age | Offset Lag |
| --- | Broker |
| --- | Cluster |

**Data “records”, ”events”, ”messages” are used interchangeably**

# Kafka partitions/Kinesis shards

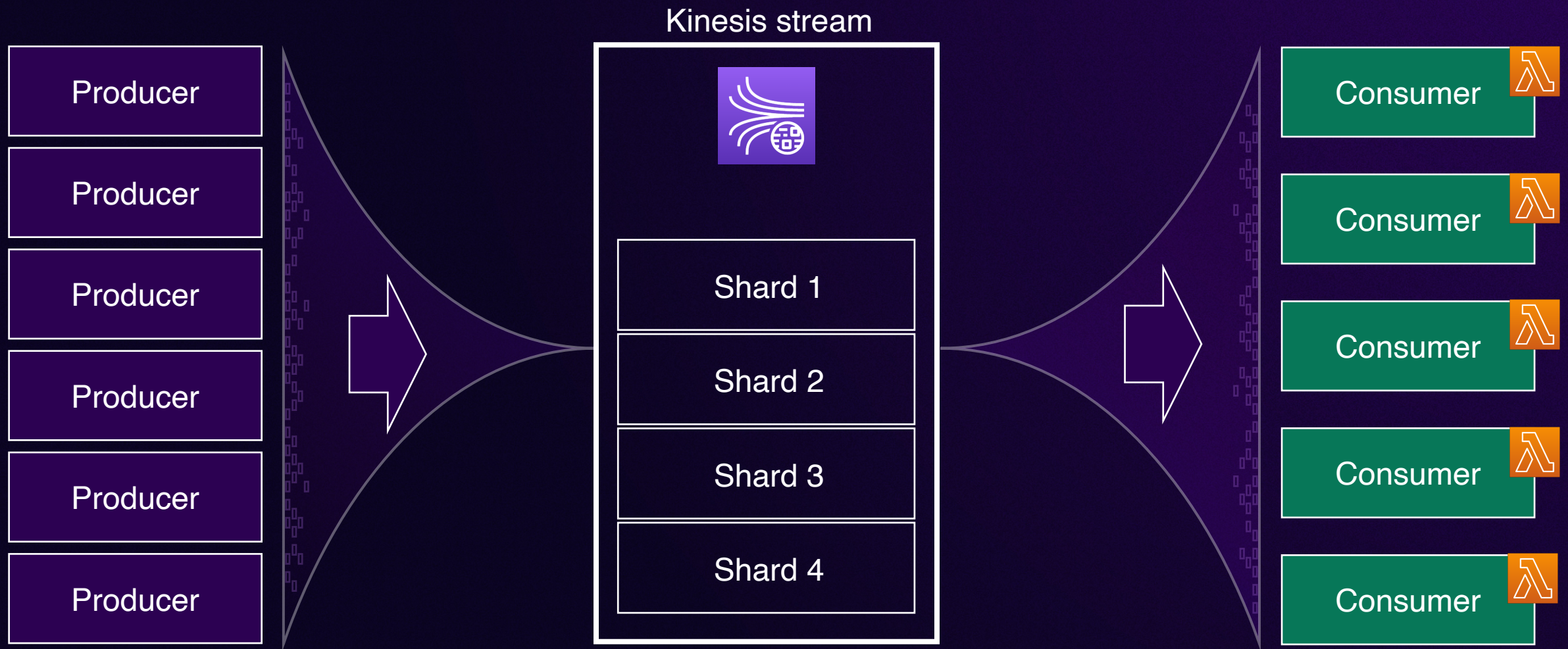


# Kafka partitions/Kinesis shards

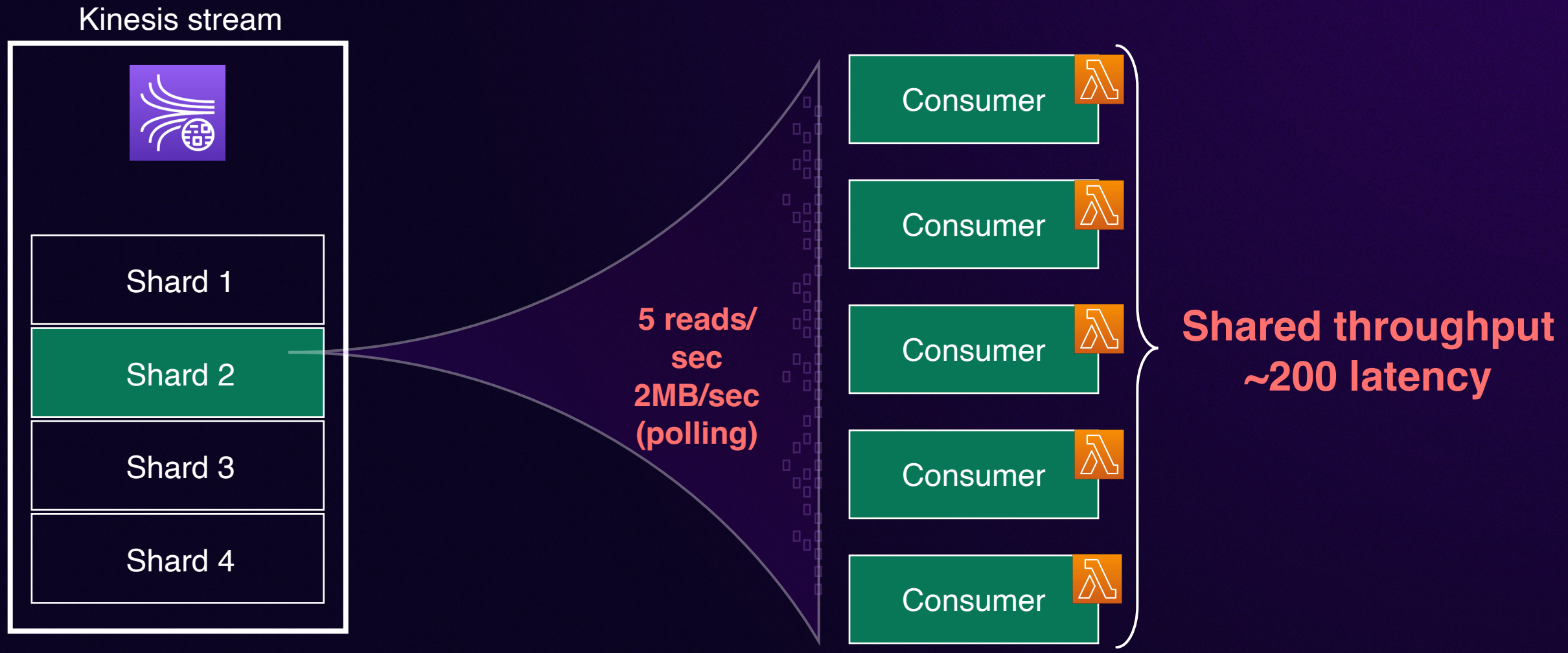


# Amazon Kinesis Data Streams

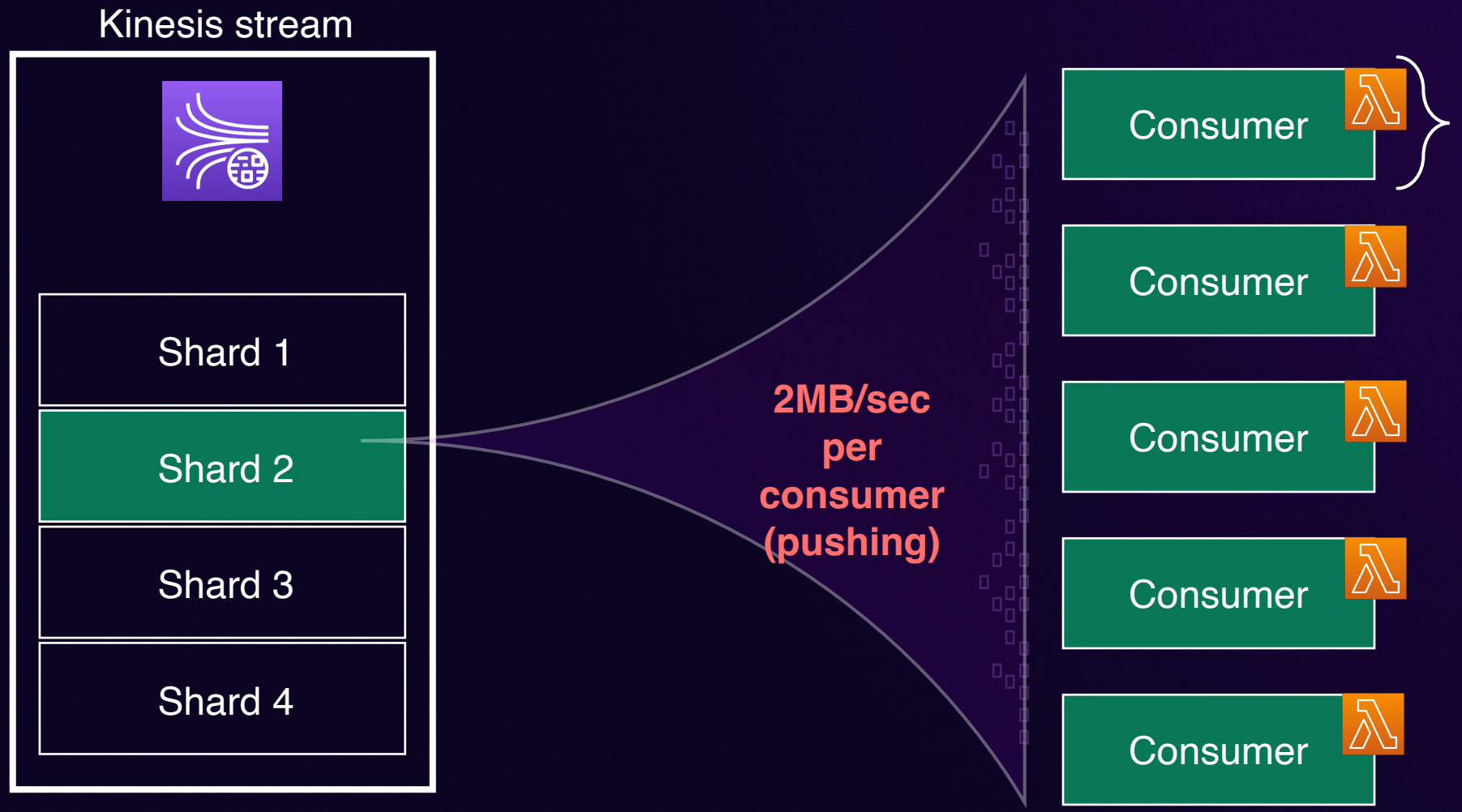
# Consuming Kinesis Data Stream



# Consuming Kinesis – shared-throughput (standard)



# Consuming Kinesis – enhanced fan-out (EFO)



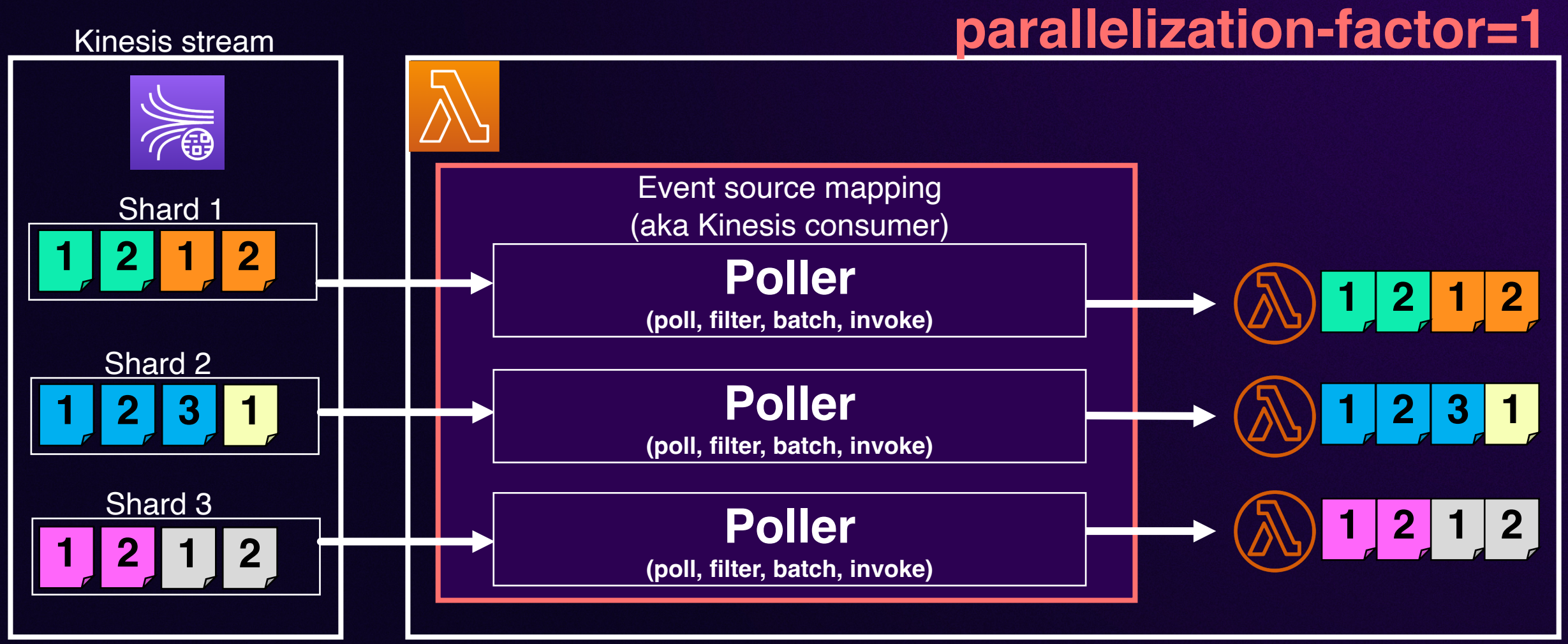

**Dedicated throughput**

**Latencies as low as 70ms**

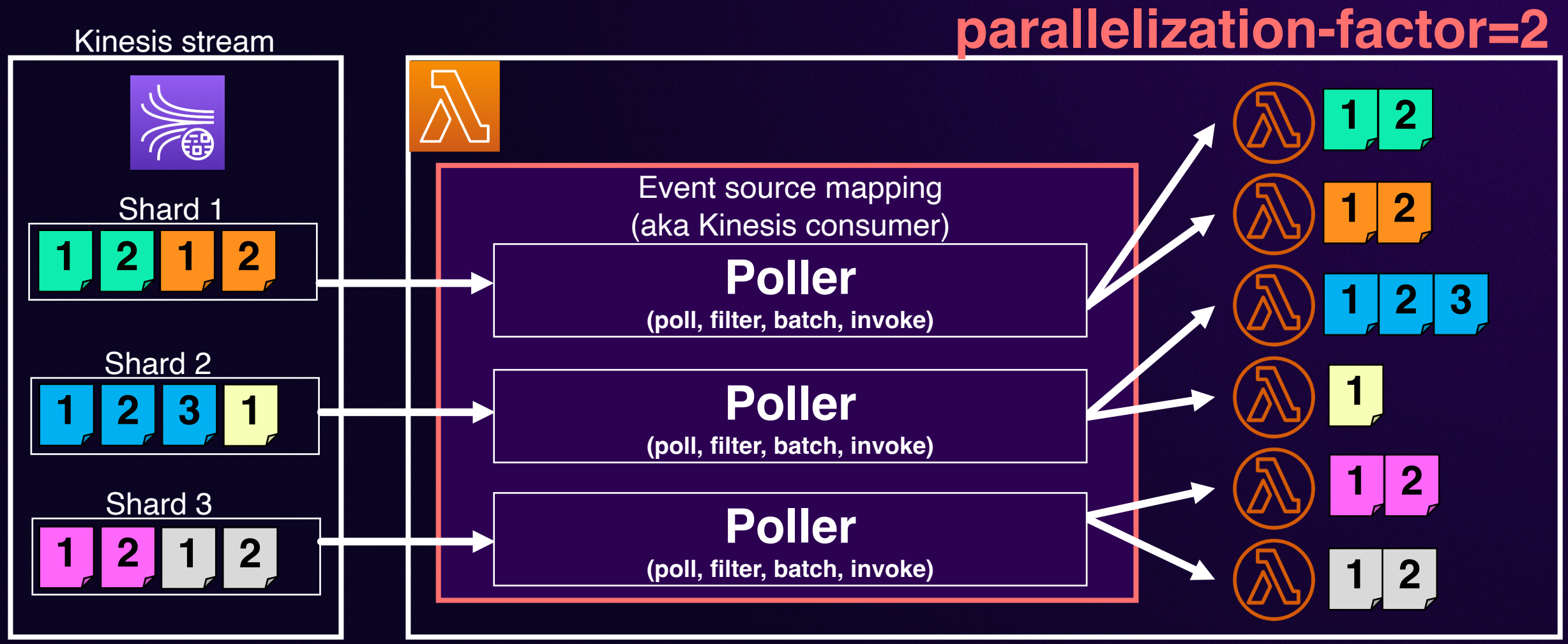
**Does not contend with 2MB/sec standard consumers**

**Up to 20 EFO consumers per stream**

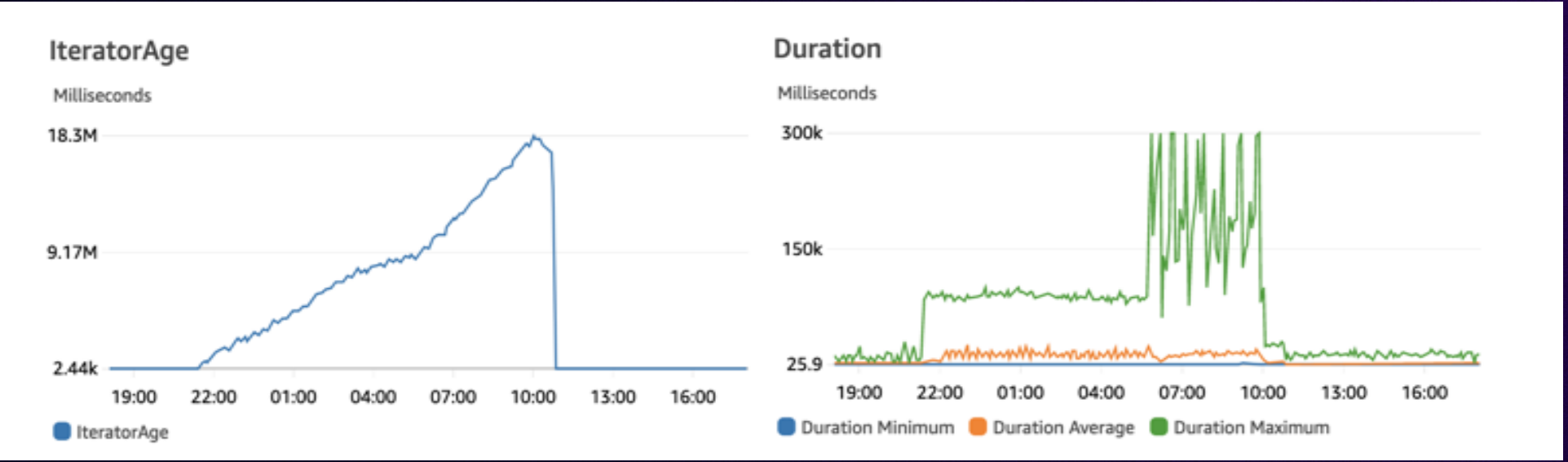
# Consuming Kinesis with Lambda ESM



# Consuming Kinesis with Lambda ESM



# Kinesis monitoring




**PutRecords.Success**
**GetRecords.Success**
**IncomingBytes / IncomingRecords**
**OutgoingBytes / OutgoingRecords**
**IteratorAgeMilliseconds**
**ReadProvisionedThroughputExceeded**
**WriteProvisionedThroughputExceeded**

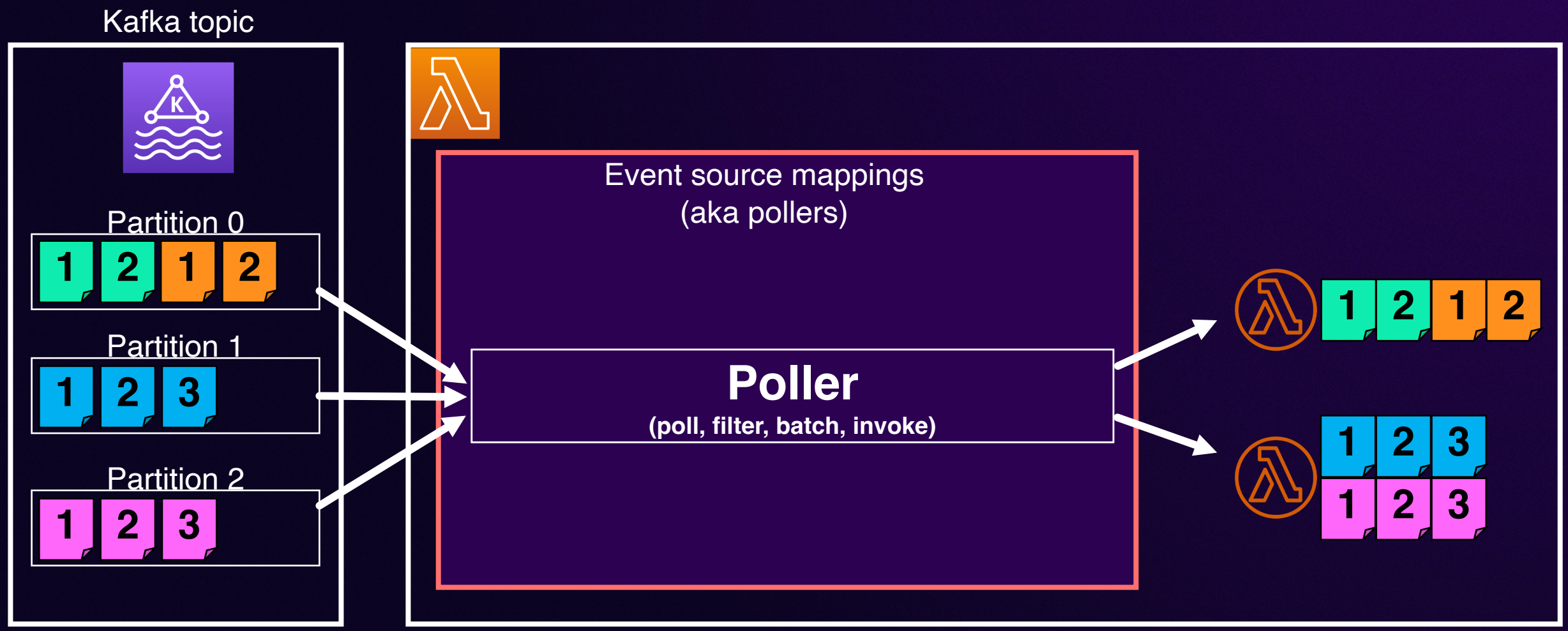
**Invocations**
**Errors**
**Throttles**
**Duration**
**ConcurrentExecutions**
**ClaimedAccountConcurrency**
**IteratorAge**

# Iterator age is growing rapidly?

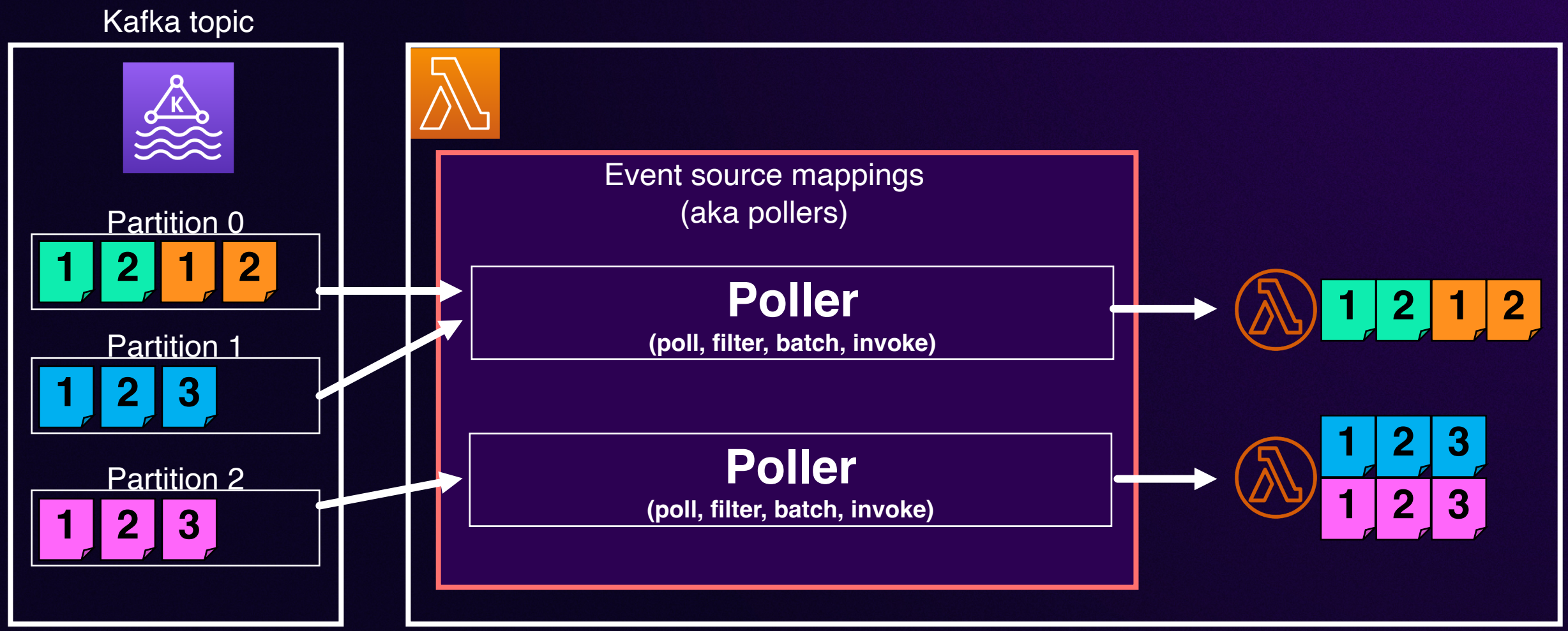
- **How many Lambda functions** are subscribed to the stream?
- Does the Lambda function show any **errors or throttles**?
- Is there a large increase in **IncomingRecords or IncomingBytes**?
- Update Lambda to log records causing errors and **return successfully**
- Scale Lambda concurrency with **parallelization factor**
- **Increase memory** allocated to the Lambda function

# Amazon MSK
# Apache Kafka

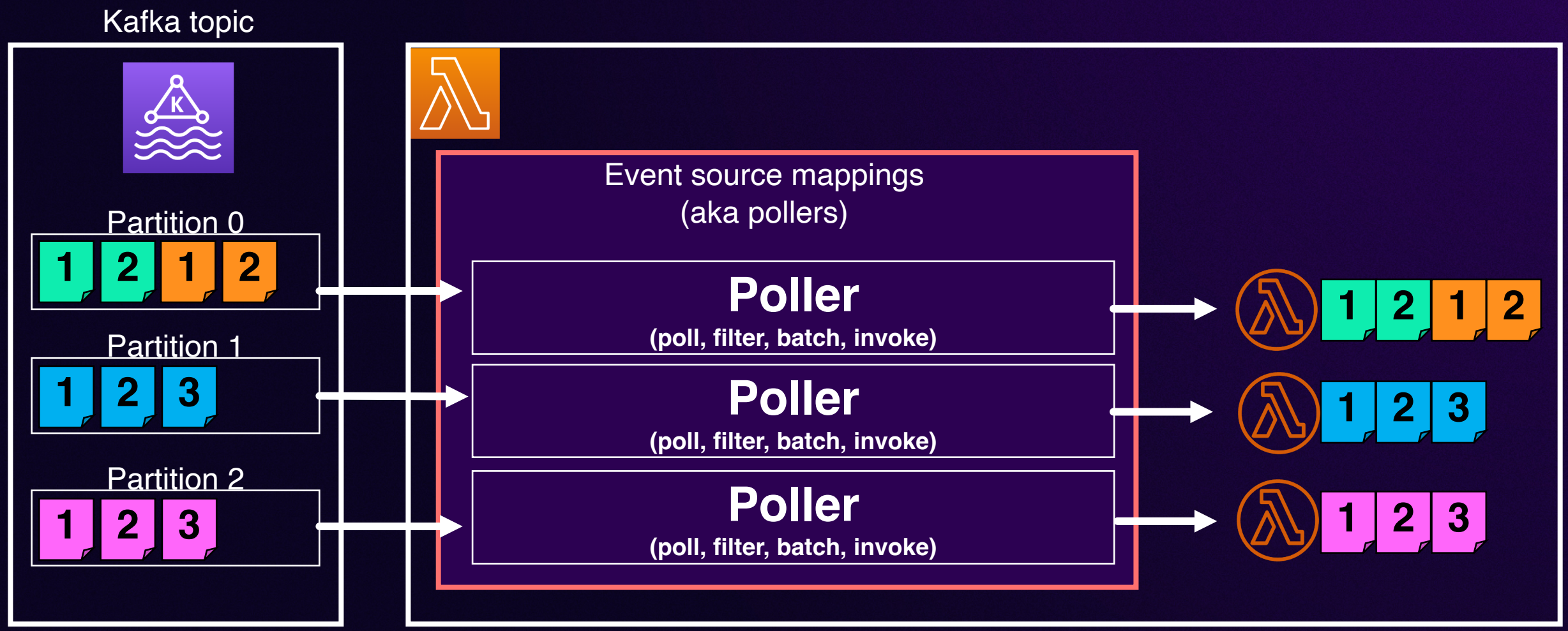
# Consuming Kafka with Lambda - scaling



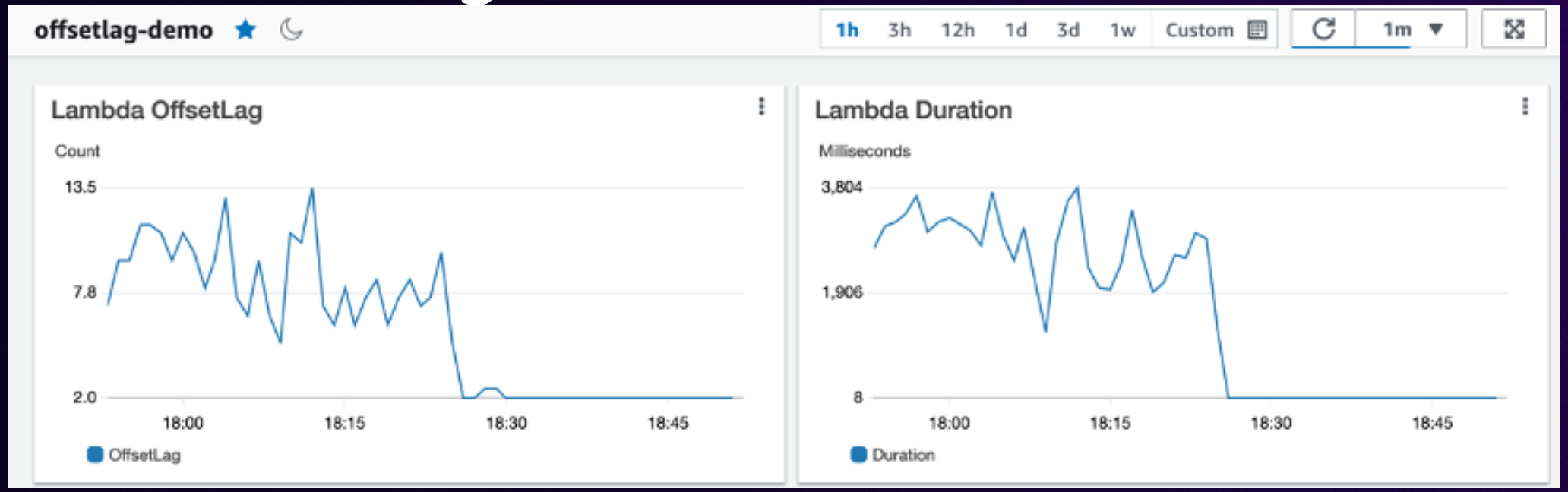
# Consuming Kafka with Lambda - scaling



# Consuming Kafka with Lambda - scaling



# Kafka monitoring




**PartitionCount**
**BytesInPerSec**
**BytesOutPerSec**
**MaxOffsetLag**
**OffsetLag**

**Throttles**
**Duration**
**ConcurrentExecutions**
**ClaimedAccountConcurrency**
**OffsetLag**

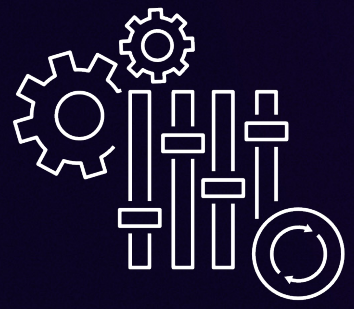


https://docs.aws.amazon.com/lambda/latest/dg/monitoring-metrics.html

But what if…

# “My Kafka workload is very spiky, latency sensitive, and requires faster, predictable performance”

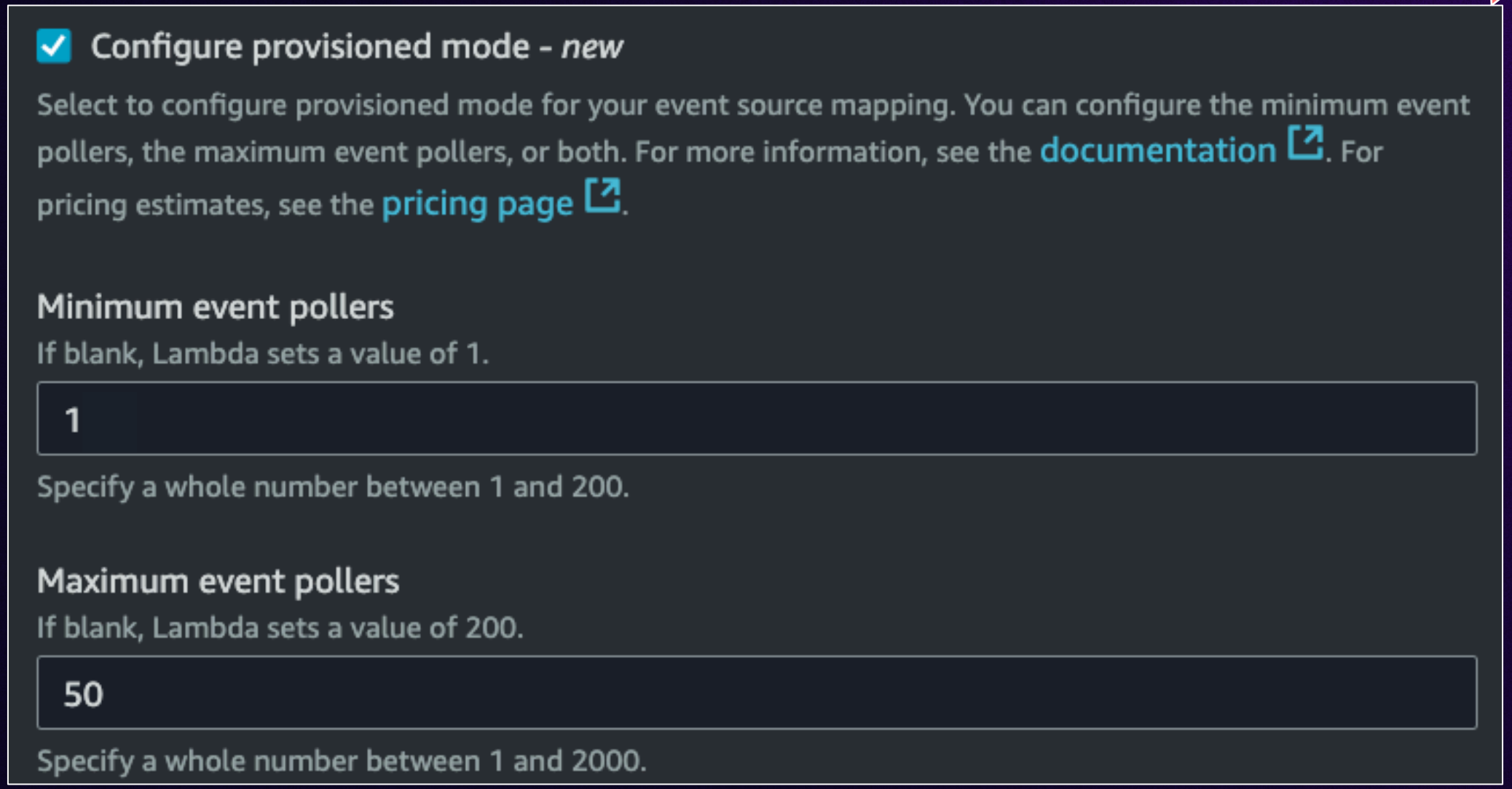
# Announcing Provisioned Mode for Kafka ESM

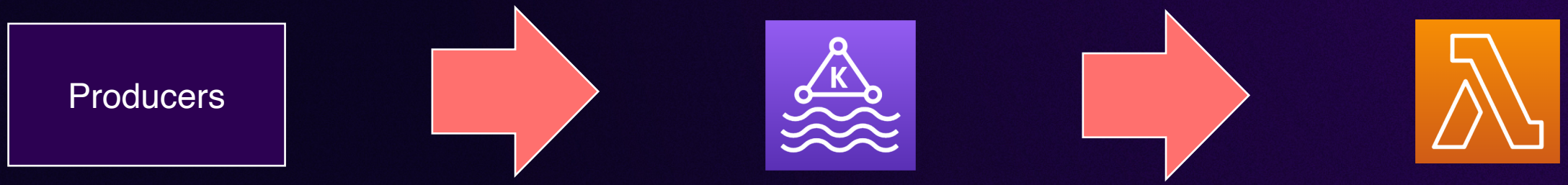
Configurable **minimum** and **maximum** number of **always-on** event pollers

**Faster scaling**, great **for latency-sensitive** workloads

# Announcing Provisioned Mode for Kafka ESM

# Let’s see the performance difference

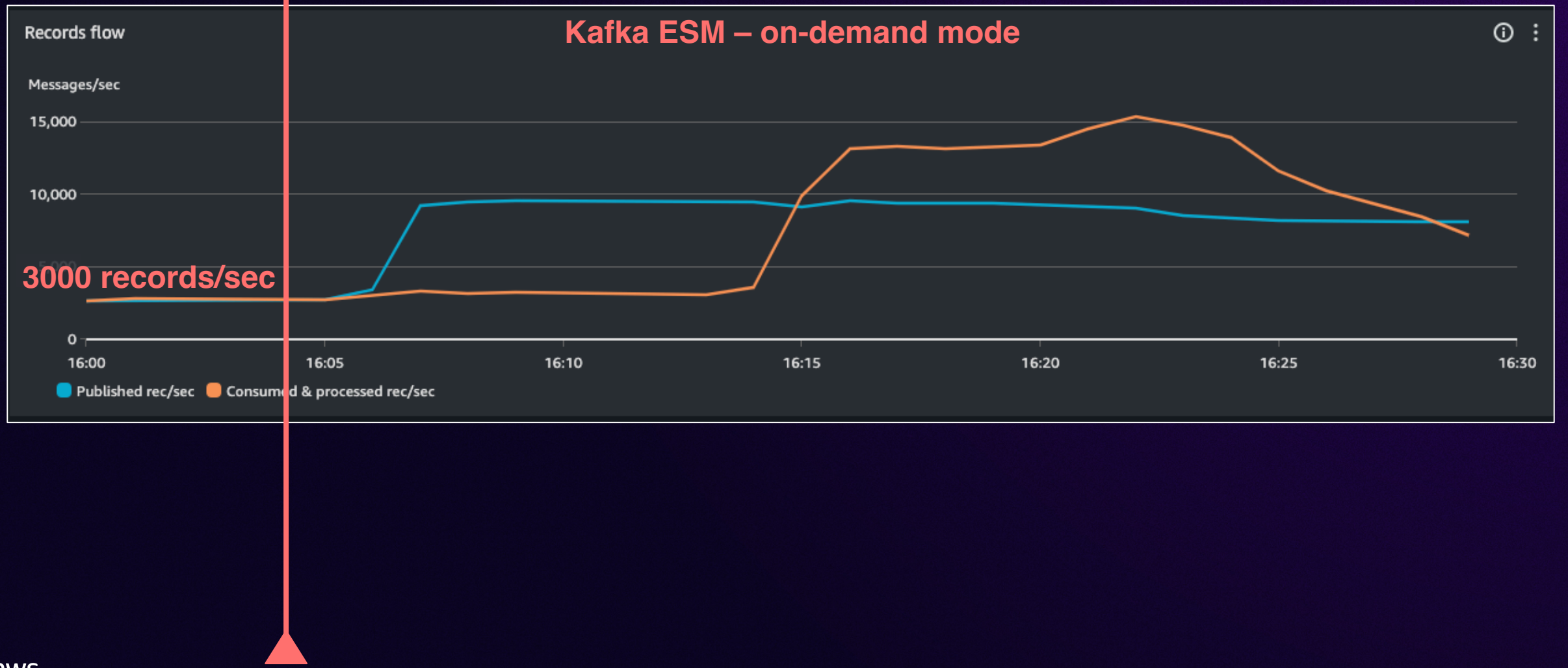



- Record size 1.5KB
- Random partition key
- **Initial traffic – 3,000 records / second**
- **Traffic spike – 9,000 records / second**

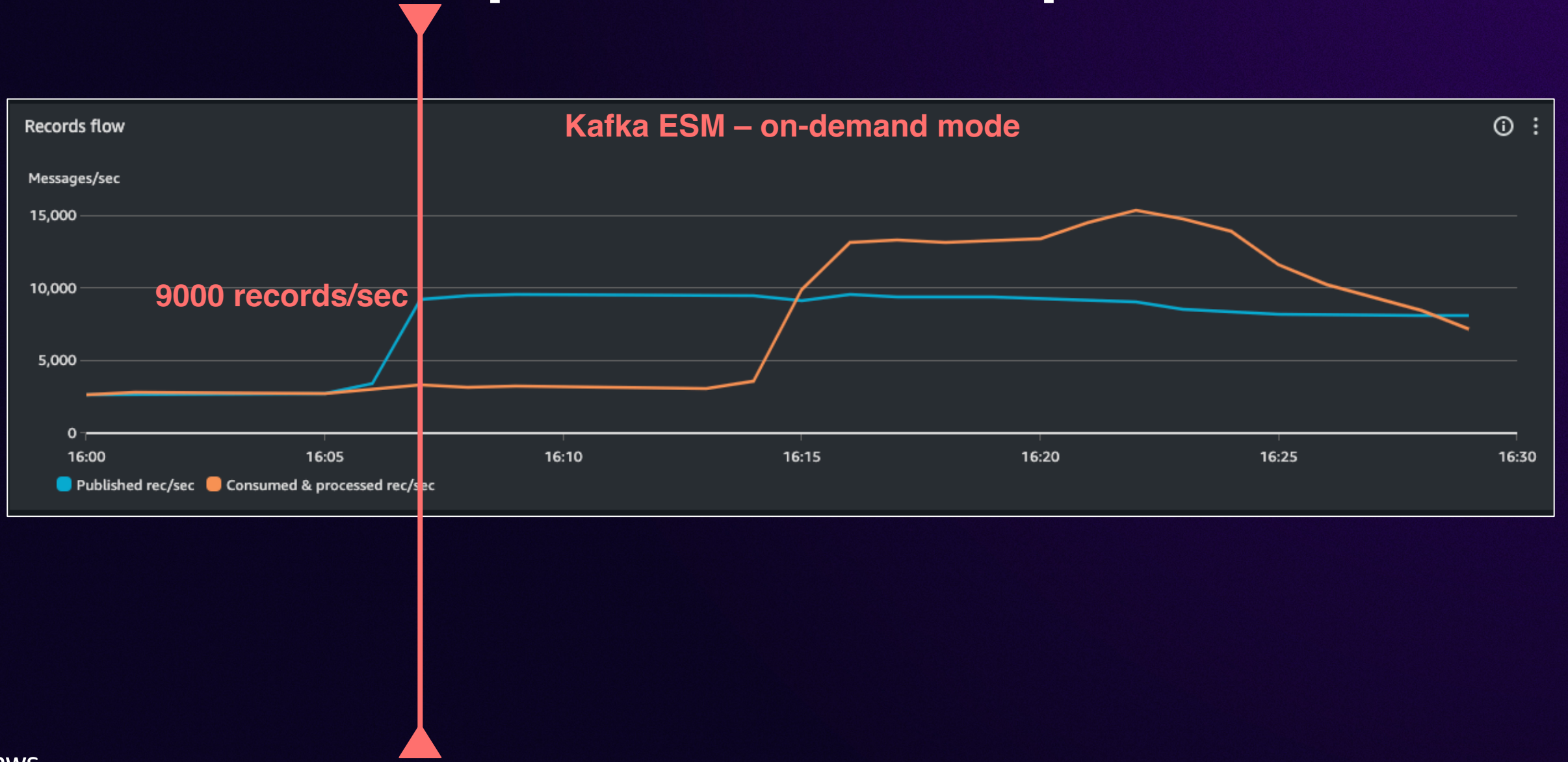
- MSK cluster
- 2 brokers
- 1 topic
- 100 partitions

- BatchSize = 50
- Batching window = 1 sec
- Mean duration = 200ms
- **Min pollers = 5**

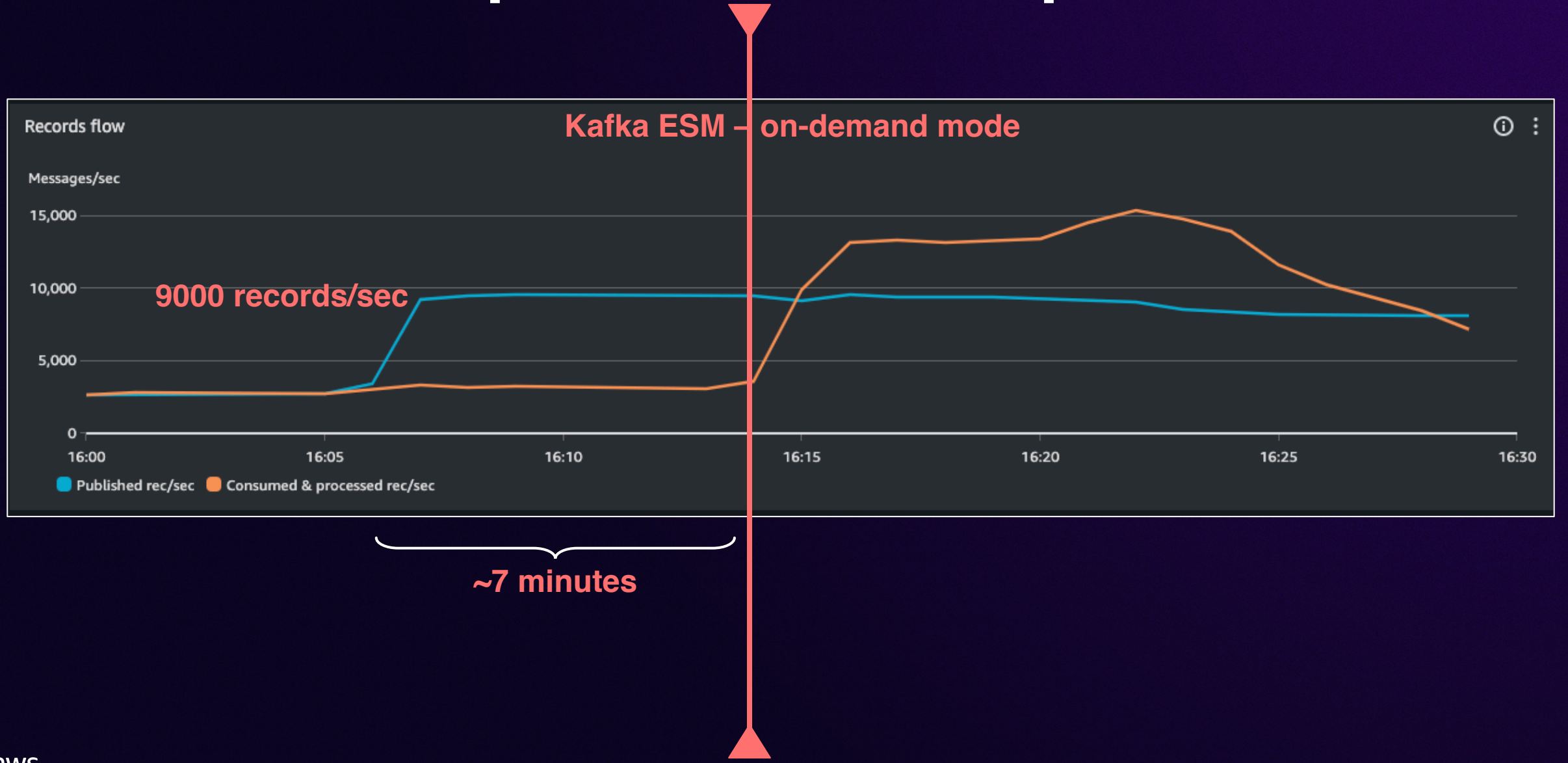
# On-demand vs. provisioned ESM performance



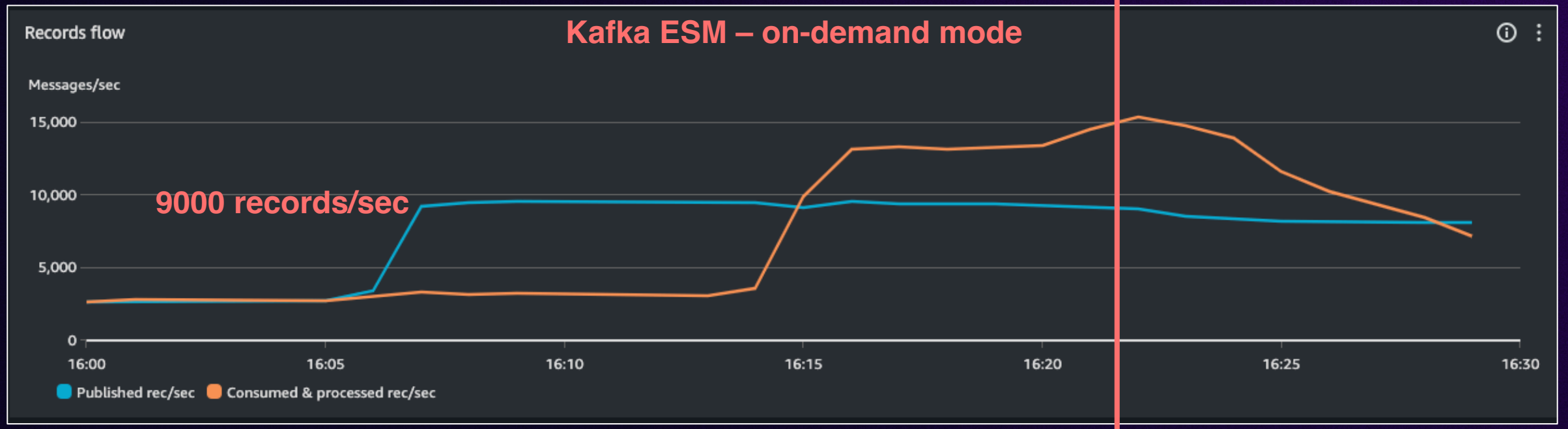
# On-demand vs. provisioned ESM performance



# On-demand vs. provisioned ESM performance

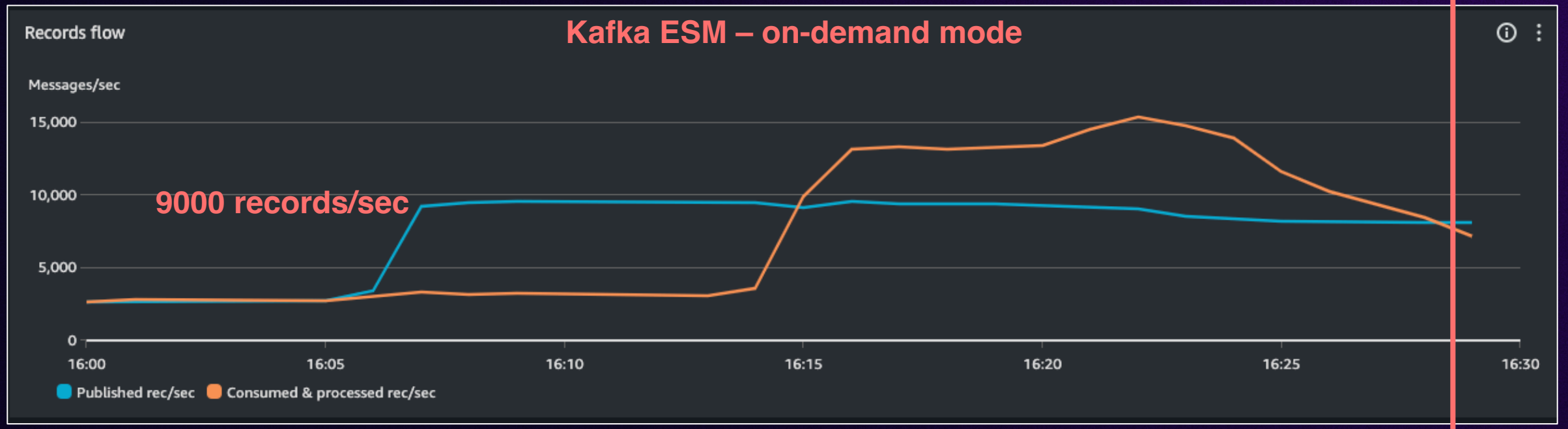


# On-demand vs. provisioned ESM performance




~7 minutes

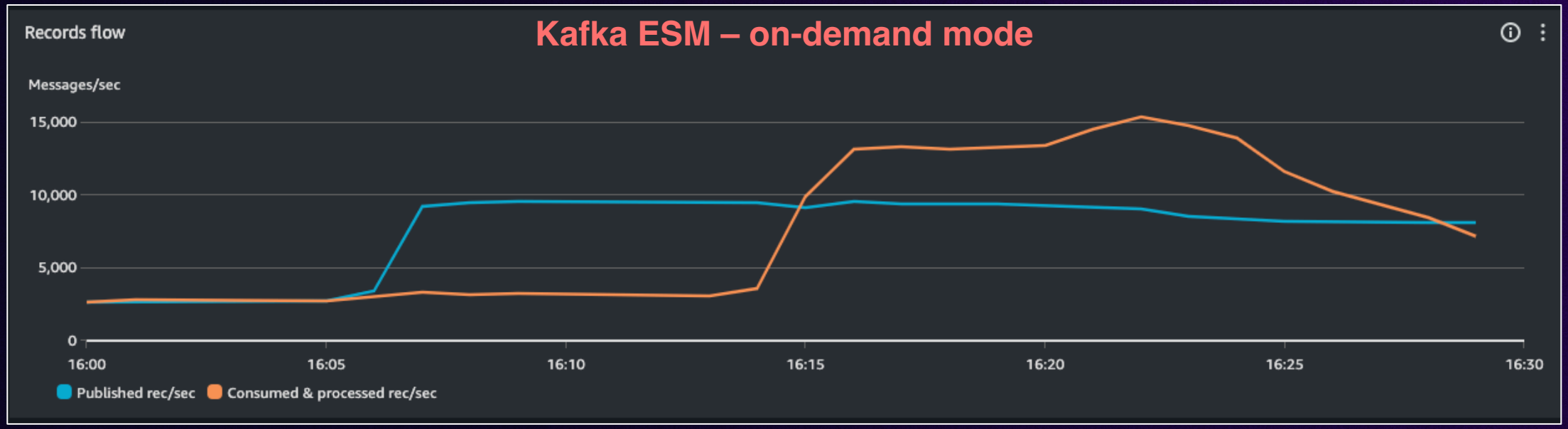
# On-demand vs. provisioned ESM performance

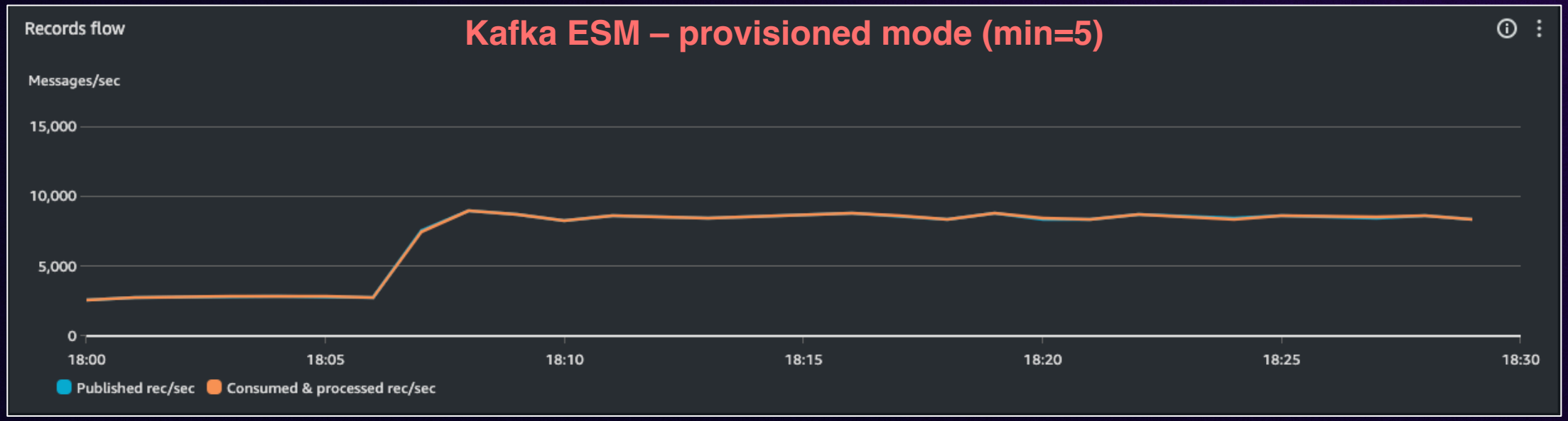



~7 minutes

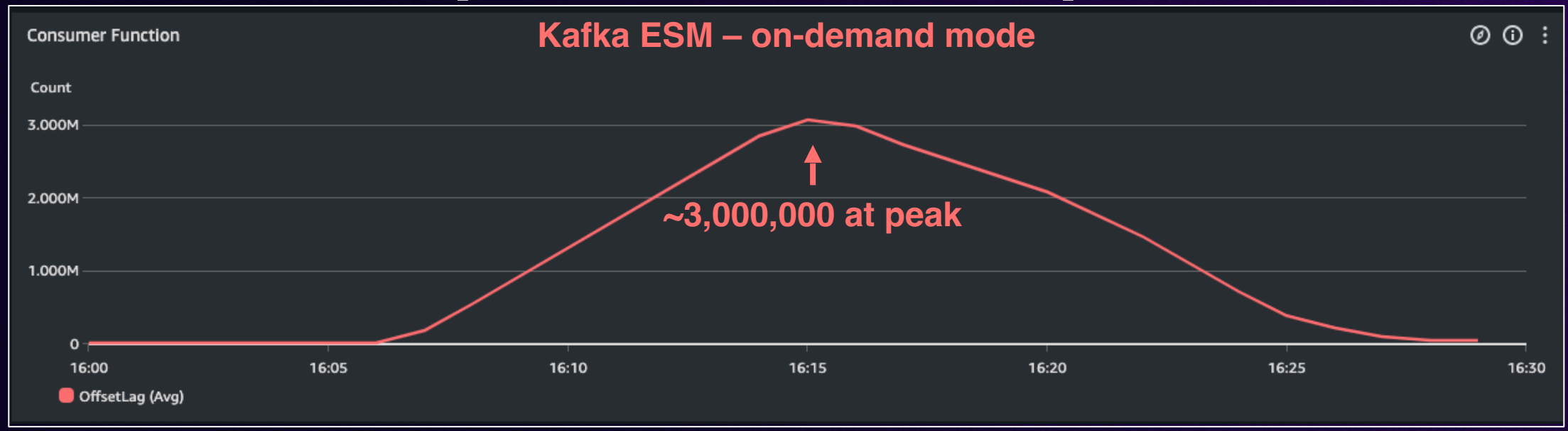
~15 minutes

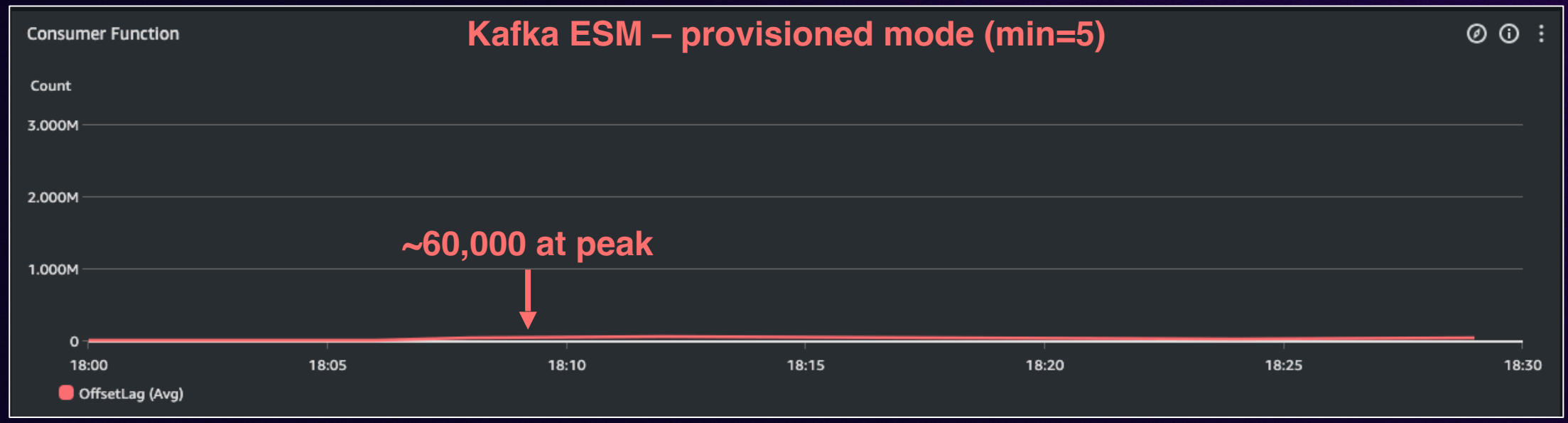
# On-demand vs. provisioned ESM performance



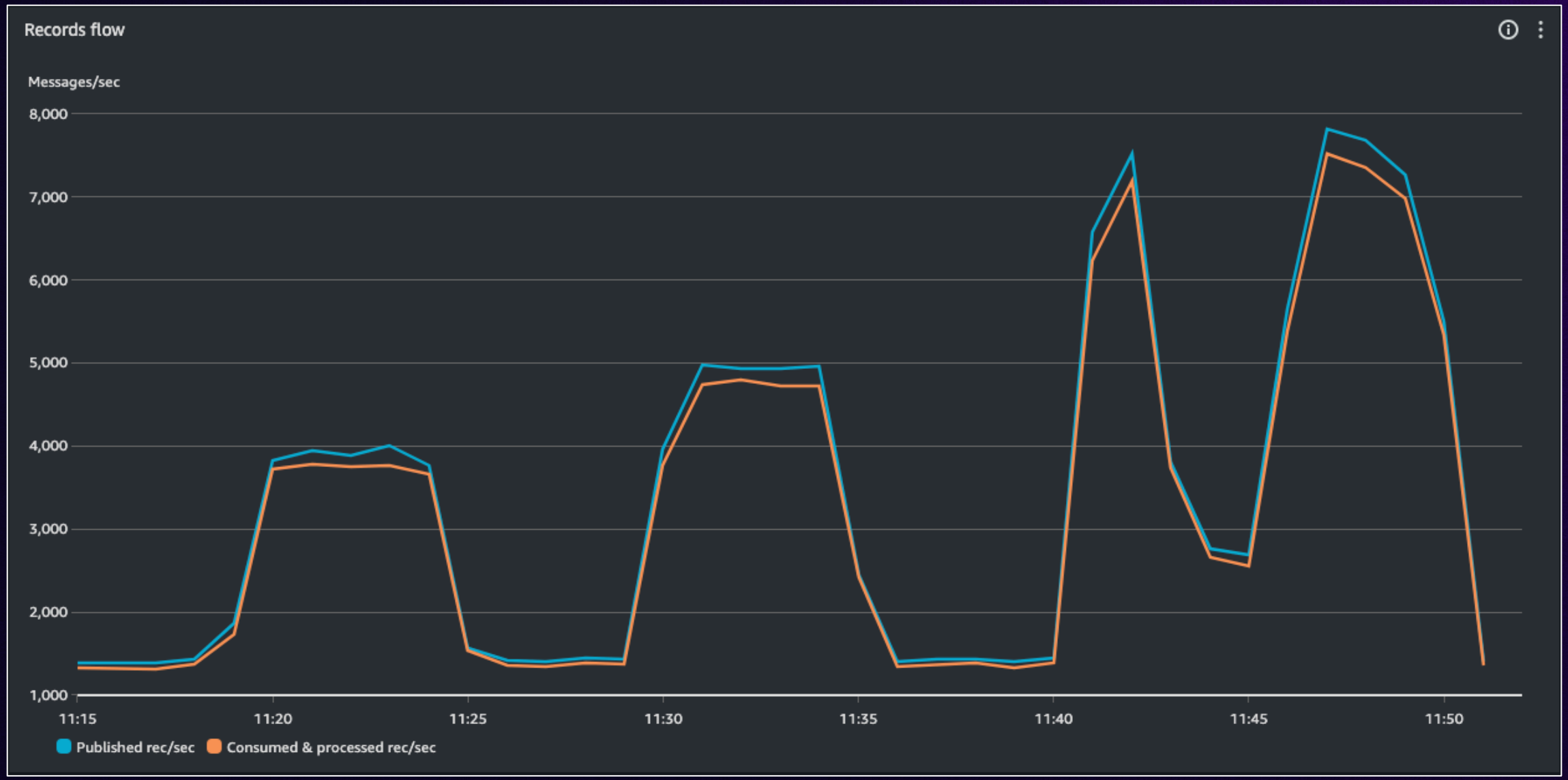



# On-demand vs. provisioned ESM performance



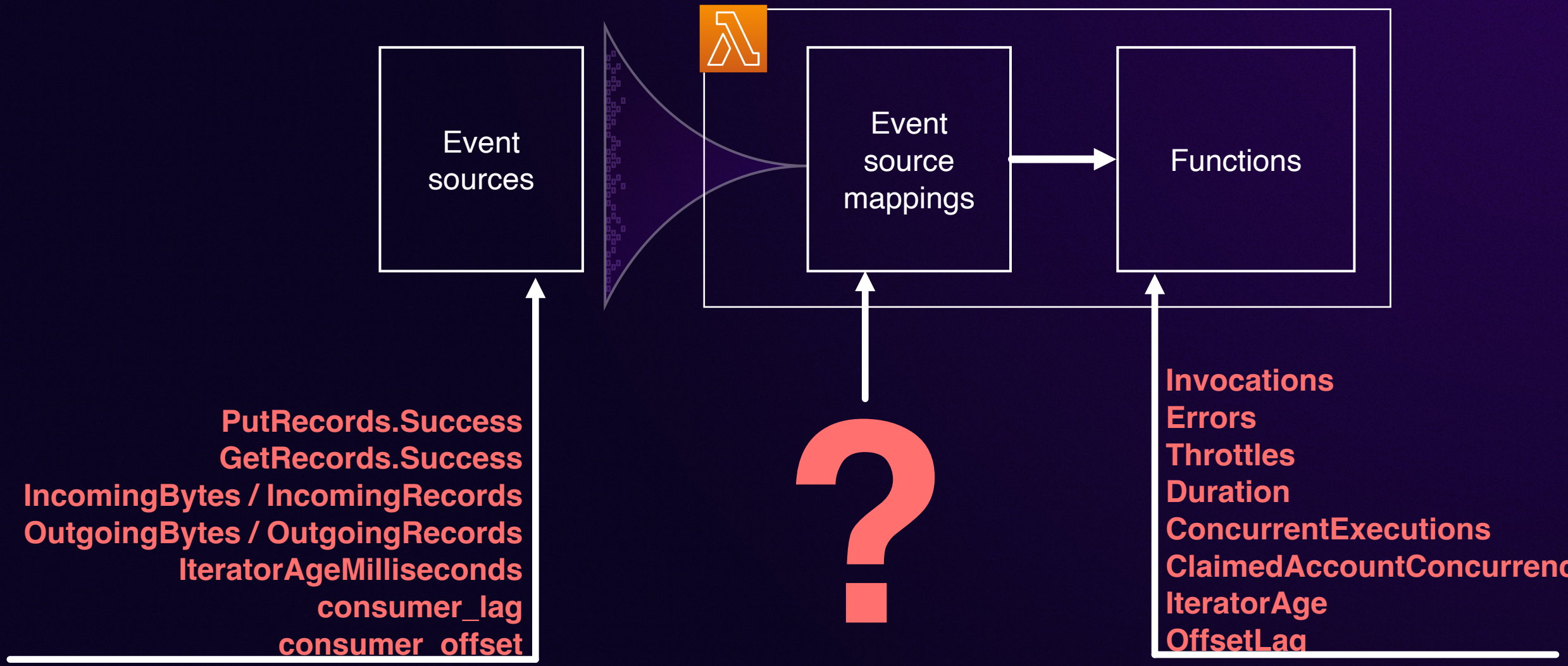



# Remember the spiky workload?



# Observability

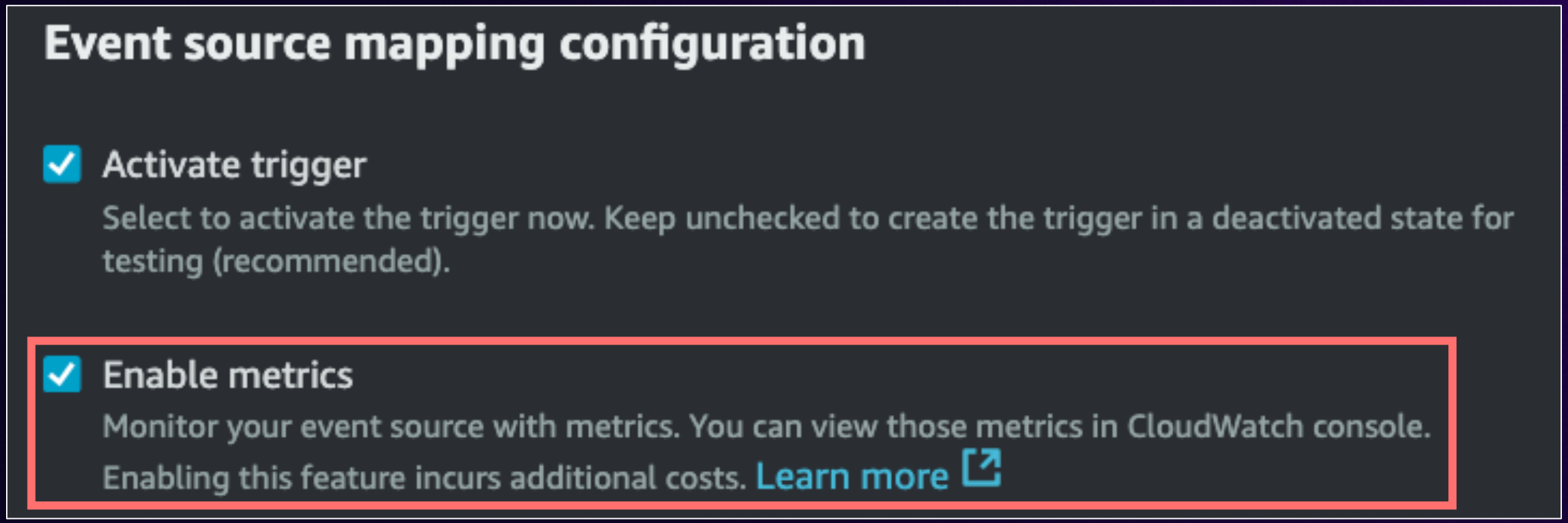
# Event source mappings observability



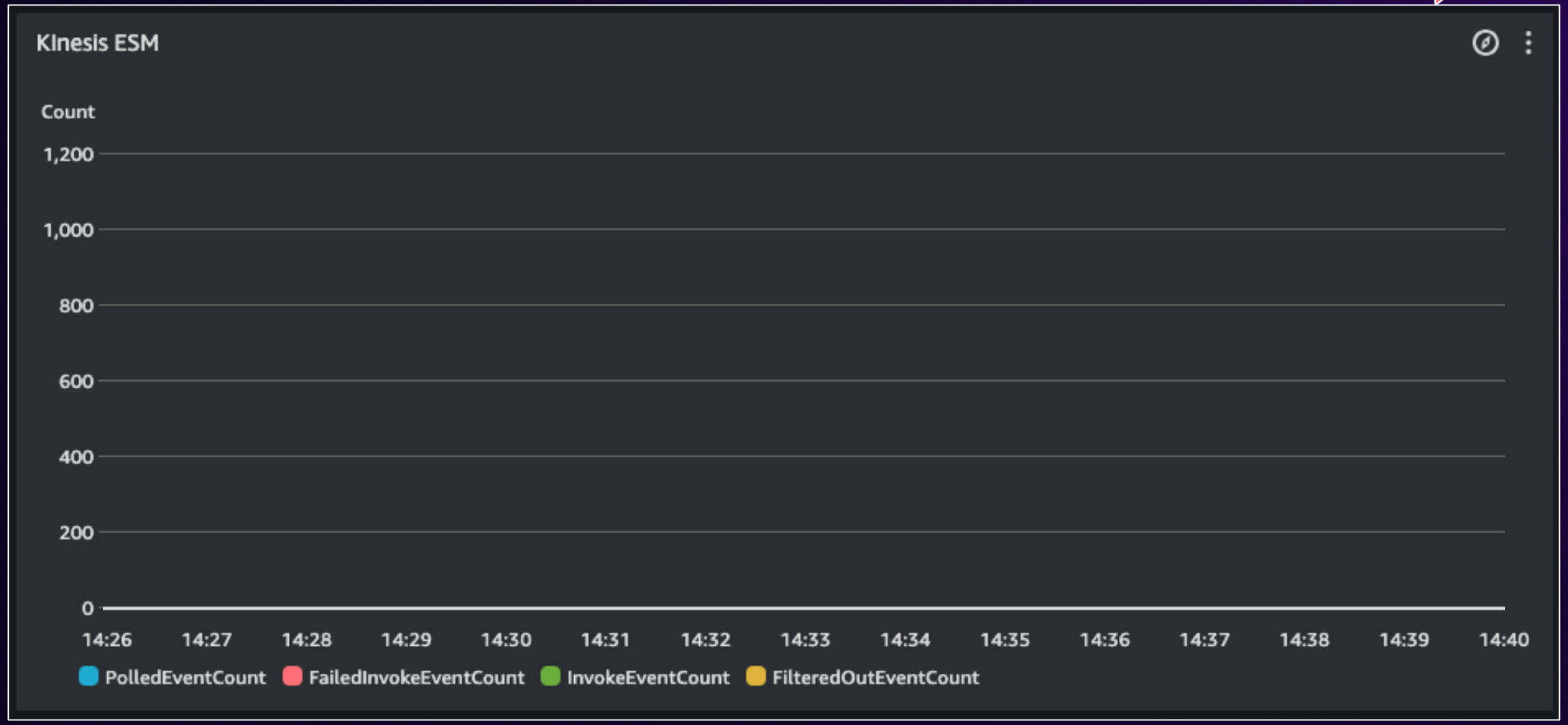
# Announcing Enhanced ESM Observability


Detailed **out-of-the-box ESM metrics** providing insights into the state of ingested messages

# Announcing Enhanced ESM Observability

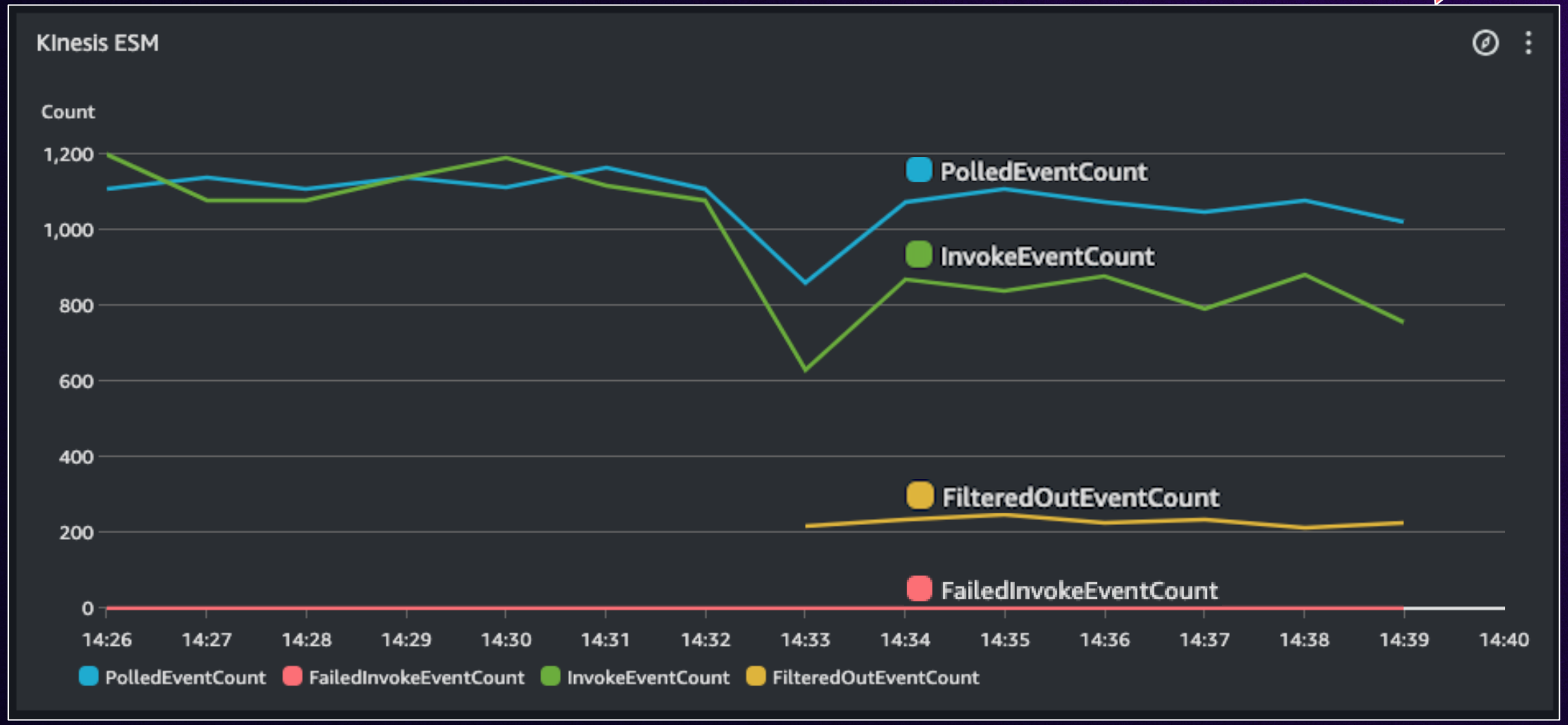



# Announcing Enhanced ESM Observability



# Announcing Enhanced ESM Observability

NEW

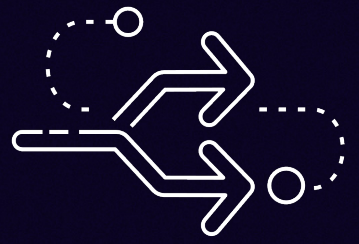


# Announcing Enhanced ESM Observability

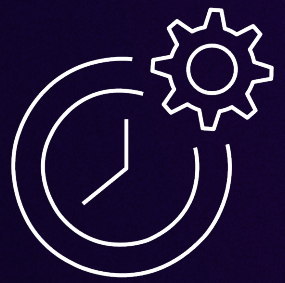

| | Amazon SQS | DynamoDB streams | Kinesis data streams |
|---|---|---|---|
| PolledEventCount | ✅ | ✅ | ✅ |
| FilteredOutEventCount | ✅ | ✅ | ✅ |
| InvokedEventCount | ✅ | ✅ | ✅ |
| FailedInvokeEventCount | ✅ | ✅ | ✅ |
| DeletedEventCount | ✅ | | |
| DroppedEventCount | | ✅ | ✅ |
| OnFailureDestinationDeliveredEventCount | | ✅ | ✅ |

# Wrapping up

# Improving throughput



**Process data in parallel**



**Reduce processing duration**

**Filter irrelevant messages out**

**Batch messages**

**Gracefully handle failures**

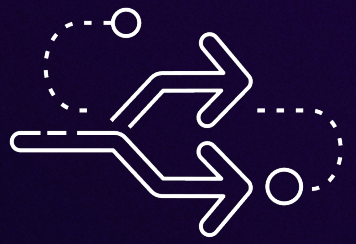
# Improving throughput

Evenly distribute records with partition key

Buffer at the producer side

Increase the number of partitions/shards

Increase parallelization factor (Kinesis)



Use enhanced fan-out (Kinesis)

# Next steps

https://aal80.github.io/reinvent2024-svs217

# Check out these other sessions

SVS321 **AWS Lambda and Apache Kafka for real-time data processing (Breakout)**
Watch on YouTube in a few weeks

SVS406 **Scale streaming workloads with AWS Lambda (Chalk talk)**
Thu Dec 05 16:00 - MGM Grand Premier 309

SVS216 **Serverless data processing with AWS Lambda and Apache Kafka (Builder)**
Wed Dec 04 08:30 - Mandalay Bay Surf B

SVS407 **Understanding AWS Lambda event source mapping (Chalk talk)**
Wed Dec 04 12:00 - MGM Grand Premier 320

SVS309 **Building EDAs with Apache Kafka and Amazon EventBridge (Chalk talk)**
Wed Dec 04 08:30 - Caesars Forum Academy 416

# Continue your AWS Serverless learning

## Learn at your own pace

Expand your serverless skills with our learning plans on **AWS Skill Builder**

## Increase your knowledge

Use our **AWS Ramp-Up Guides** to build your serverless knowledge

## Earn AWS Serverless badge


Demonstrate your knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

# Thank you!

Please complete the session survey in the mobile app

**Anton Aleksandrov**
antonal80