
AWS
re:Invent

SVS304

Understanding AWS Lambda streaming events

Adam Wagner
Solutions Architect
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Introduction to streaming event sources for Lambda

Scaling

Monitoring

Common issues

Performance and optimization



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Session expectations

- What we will cover
  - The details of using Lambda with Amazon Kinesis Data Streams and Amazon DynamoDB Streams
    - Scaling
    - Monitoring
    - Performance and optimization
- What we won’t cover
  - What is serverless
  - What is Lambda
  - Event sources outside of Kinesis Data Streams and DynamoDB Streams

Streaming event sources



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Kinesis

Easily collect, process, and analyze video and data streams in real time



Amazon Kinesis Video Streams

Capture, process, and store video streams for analytics



Kinesis Data Streams

Build custom applications that analyze data streams



Amazon Kinesis Data Firehose

Load data streams into AWS data stores



Amazon Kinesis Data Analytics

Analyze data streams with SQL

We will focus on Kinesis Data Streams

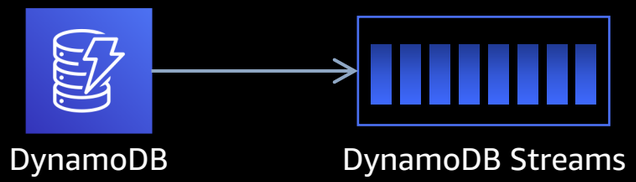
Amazon DynamoDB



DynamoDB

- Fully-managed, no SQL
- Document or key-value
- Scales to any workload
- Fast and consistent
- Access control
- Event-driven programming

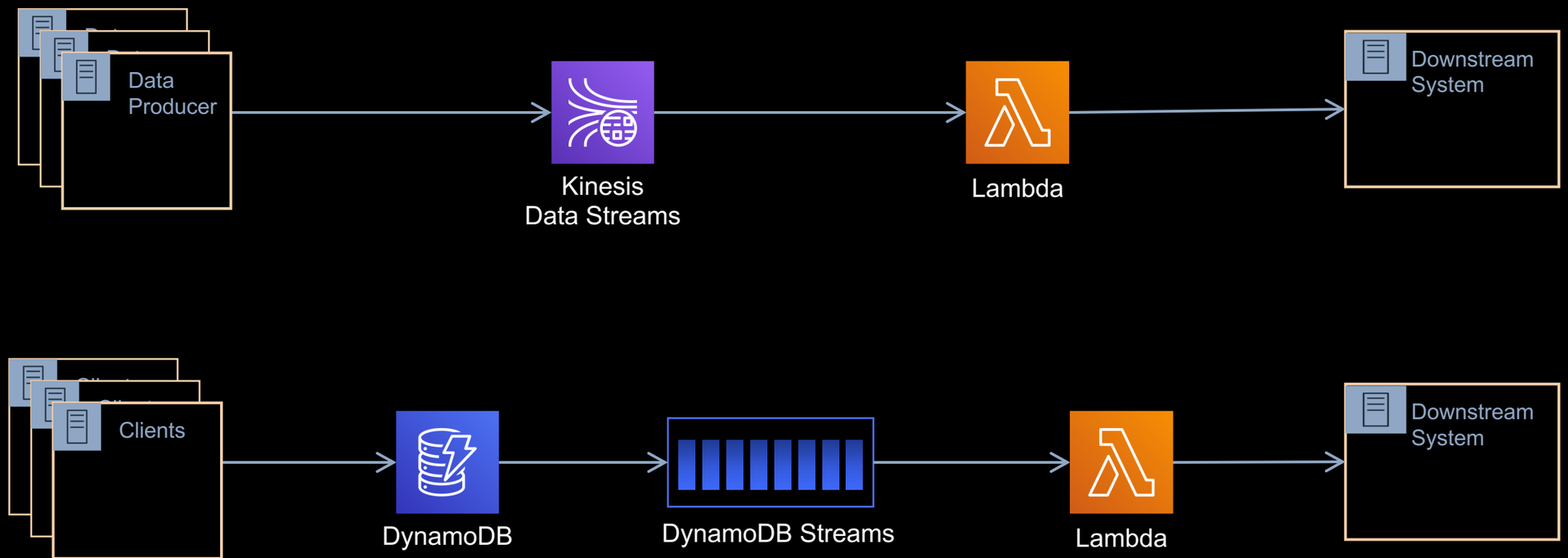
DynamoDB Streams



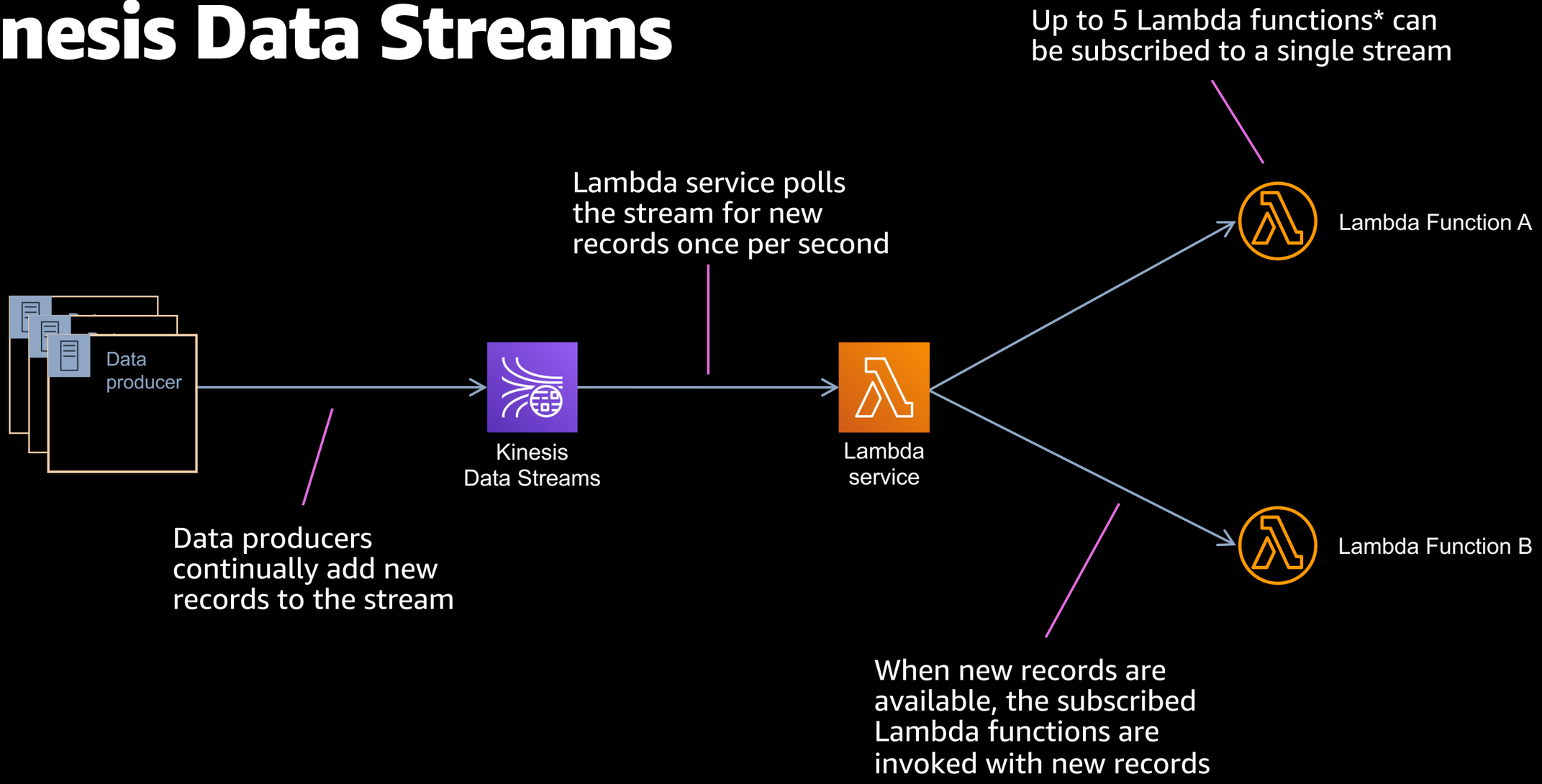
DynamoDB Streams

- ✓ Stream of item changes
- ✓ Exactly once, guaranteed delivery
- ✓ Strictly-ordered by key
- ✓ Durable, scalable
- ✓ Fully-managed
- ✓ 24-hour data retention
- ✓ Sub-second latency
- ✓ Event source for Lambda

What we're talking about today

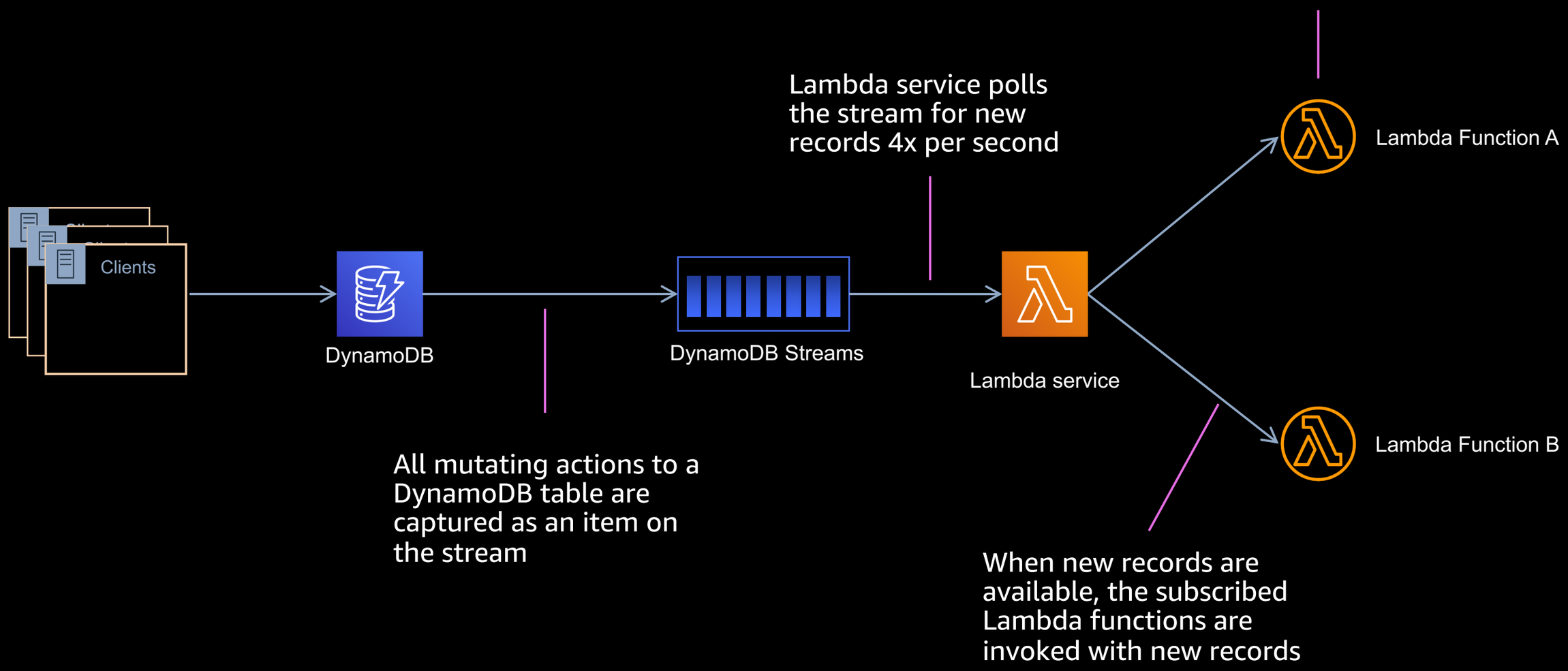


Kinesis Data Streams

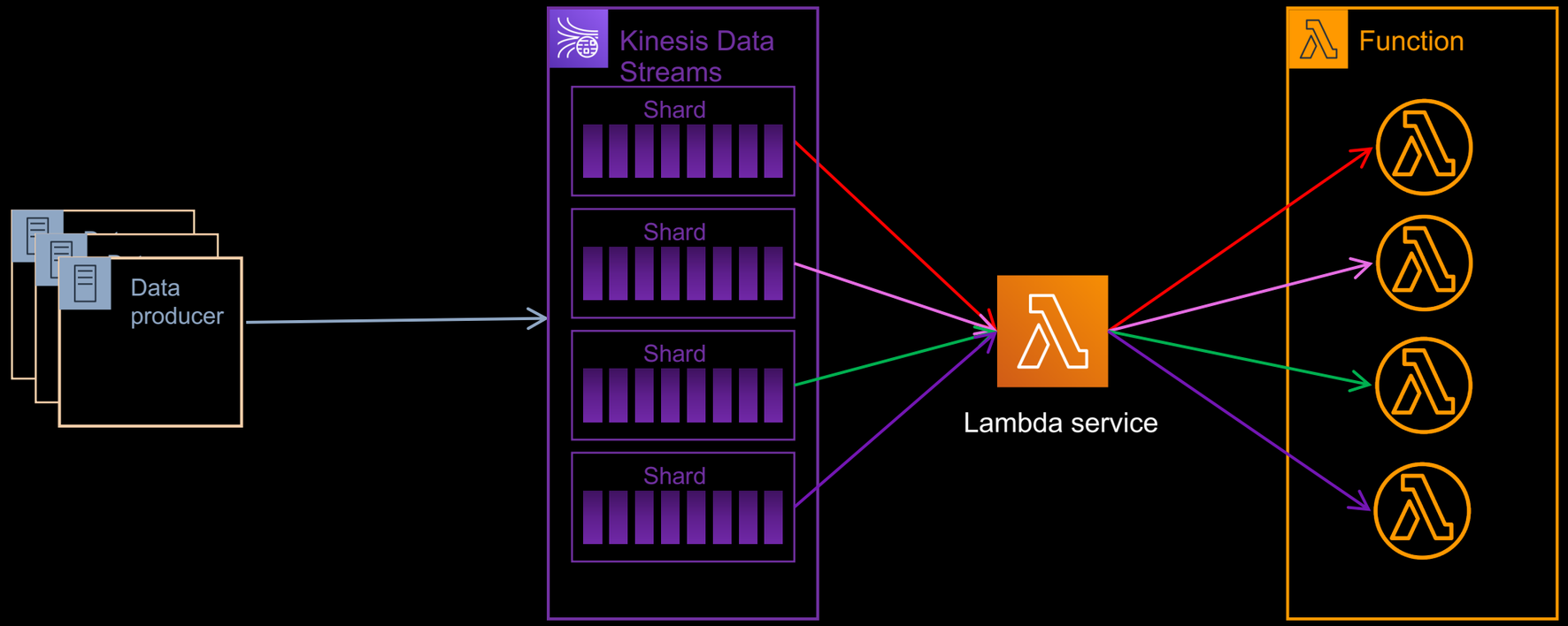


*Kinesis Data Analytics and Kinesis Data Firehose also count toward this subscriber limit

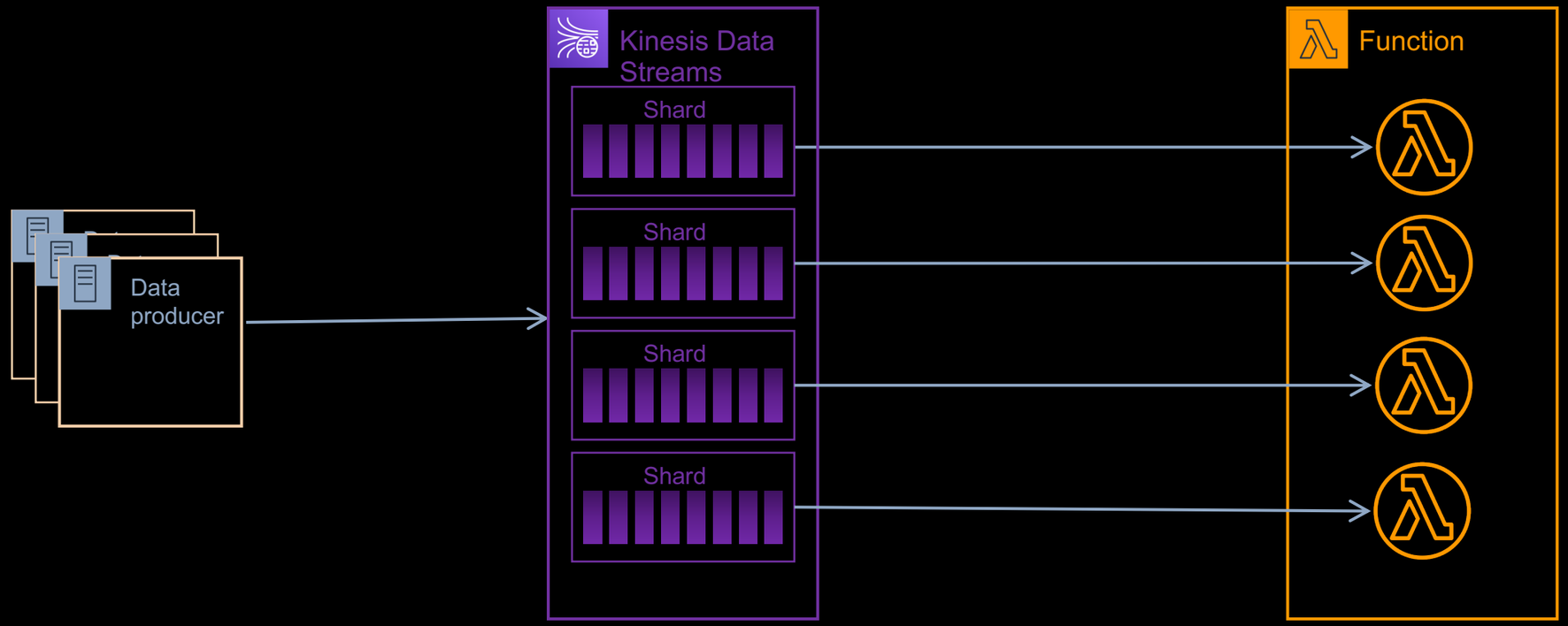
DynamoDB Streams



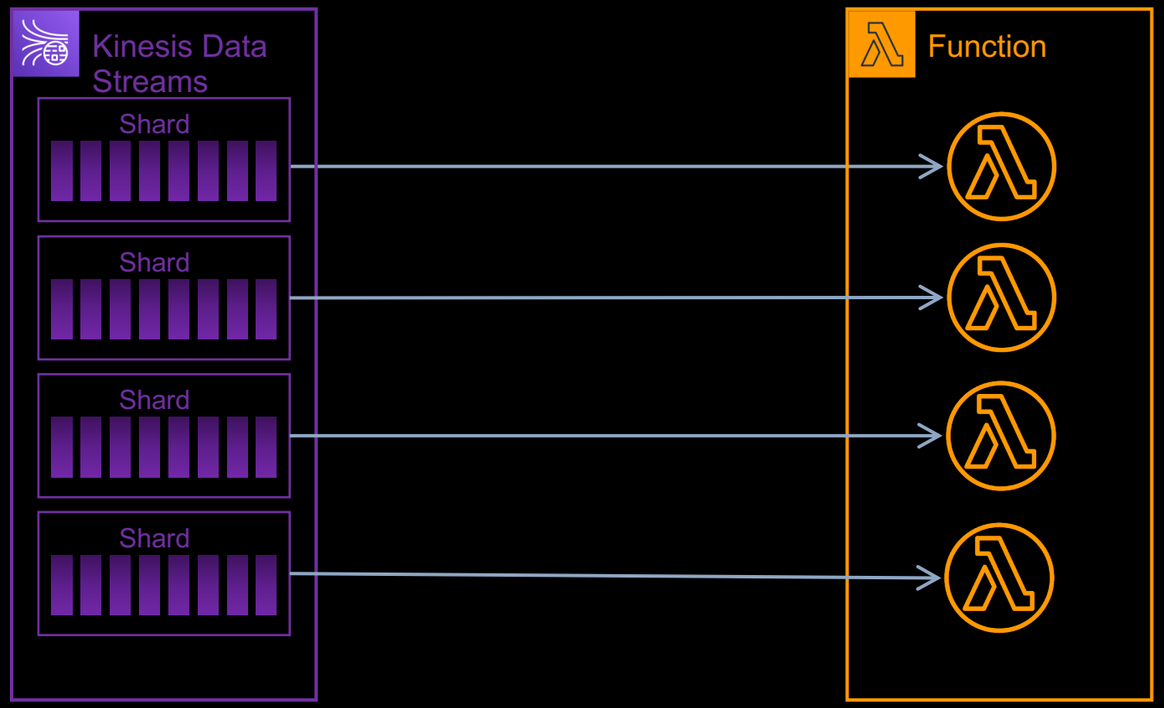
Kinesis Data Streams shard detail



Kinesis Data Streams



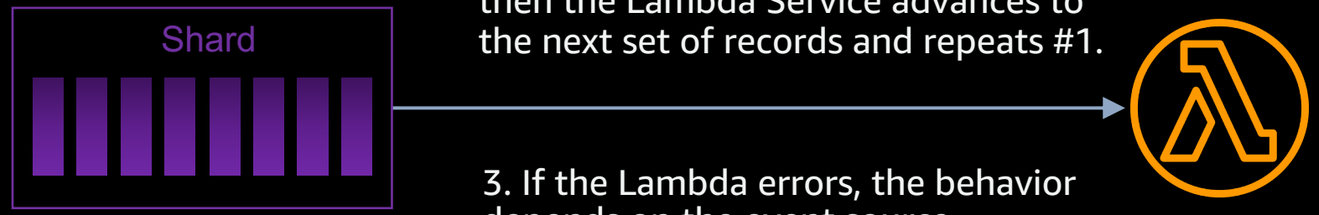
Kinesis Data Streams



Kinesis Data Stream shard level detail

1. Lambda Service polls the shard once per second for a set of records. Then synchronously invokes the Lambda Function with the batch of records.

2. If the Lambda returns successfully then the Lambda Service advances to the next set of records and repeats #1.



3. If the Lambda errors, the behavior depends on the event source configuration.

Default settings

The Lambda Service invokes the function with the same set of records and will continue to do so until it succeeds or the records age out of the stream.

Bisect batch on function error

The Lambda Service splits the batch and retries the older half first. It continues to do this until it finds the problematic record. Once it can no longer split the batch, the single problematic record is processed based on the maximum retry and record age settings. Note that earlier retries as part of the bisecting do NOT count toward maximum retries.

Maximum retries and maximum age

The Lambda Service retries a batch until either the records are older than the maximum record age or it has reached the maximum retry number. Once these are reached the records are sent to the configured failure destination. If no failure destination is configured, they are discarded.

Scaling



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

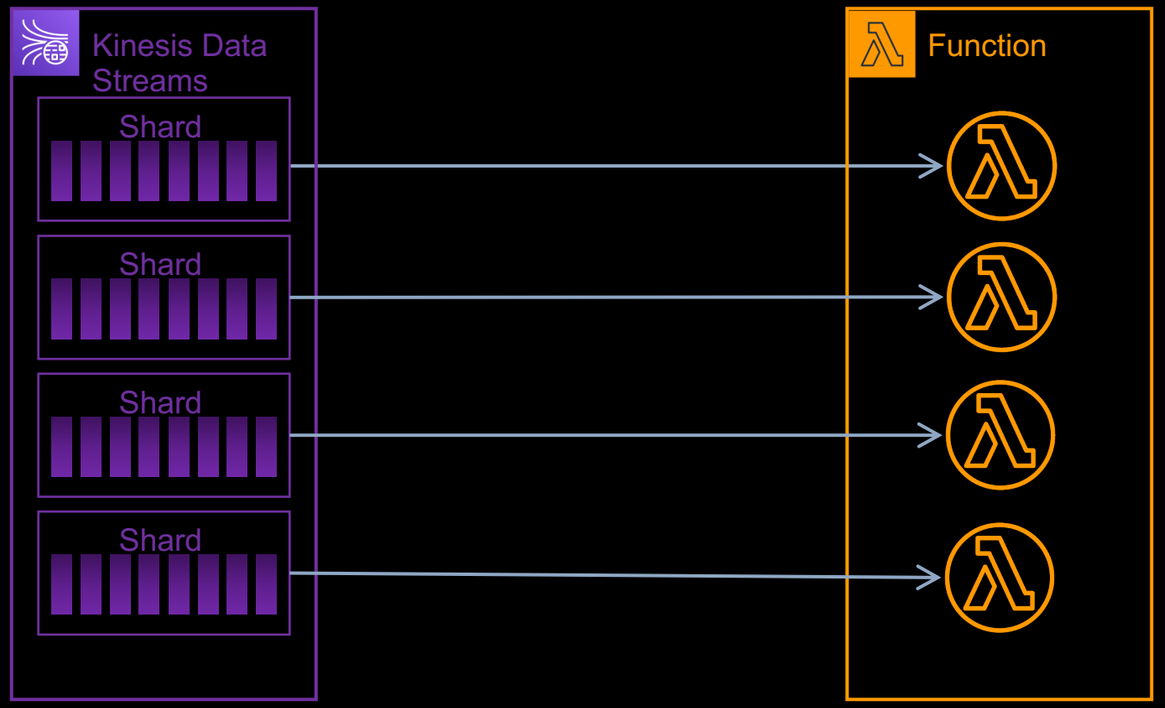
Kinesis Data Streams scaling

Kinesis Data Streams scales by adding shards

Easiest method is the UpdateShardCount API

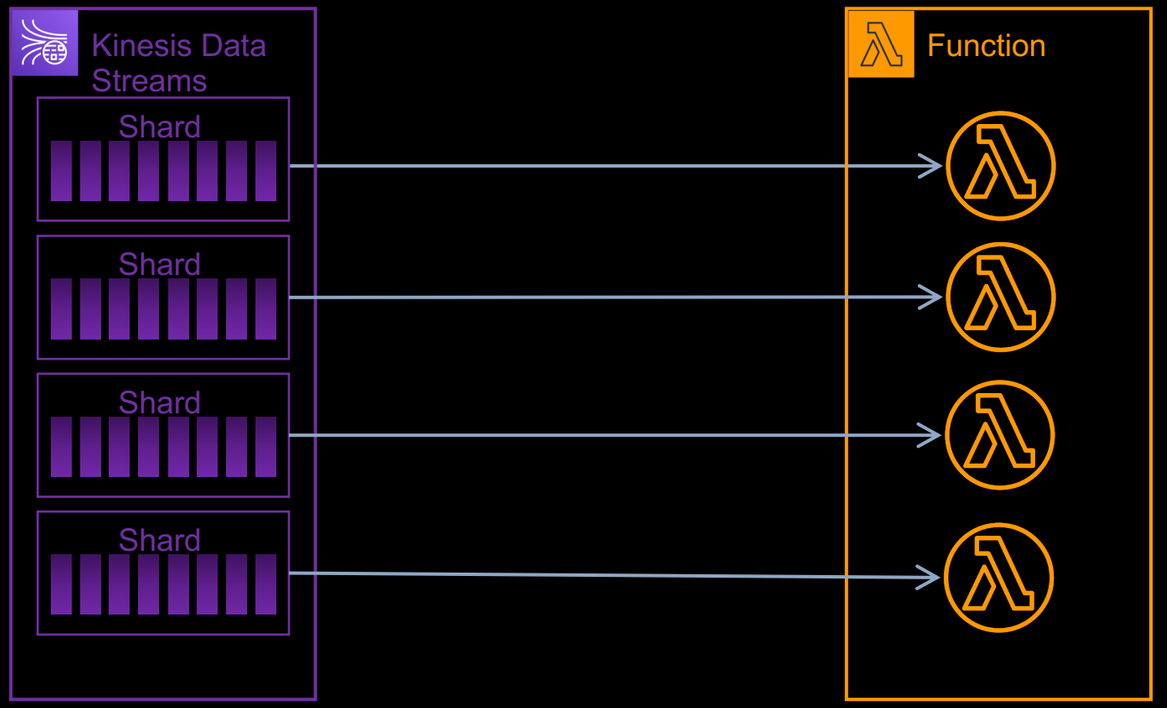
- Set a target shard count and Kinesis takes care of splitting and merging shards for you

Alternatively SplitShard and MergeShards allow for more targeted scaling



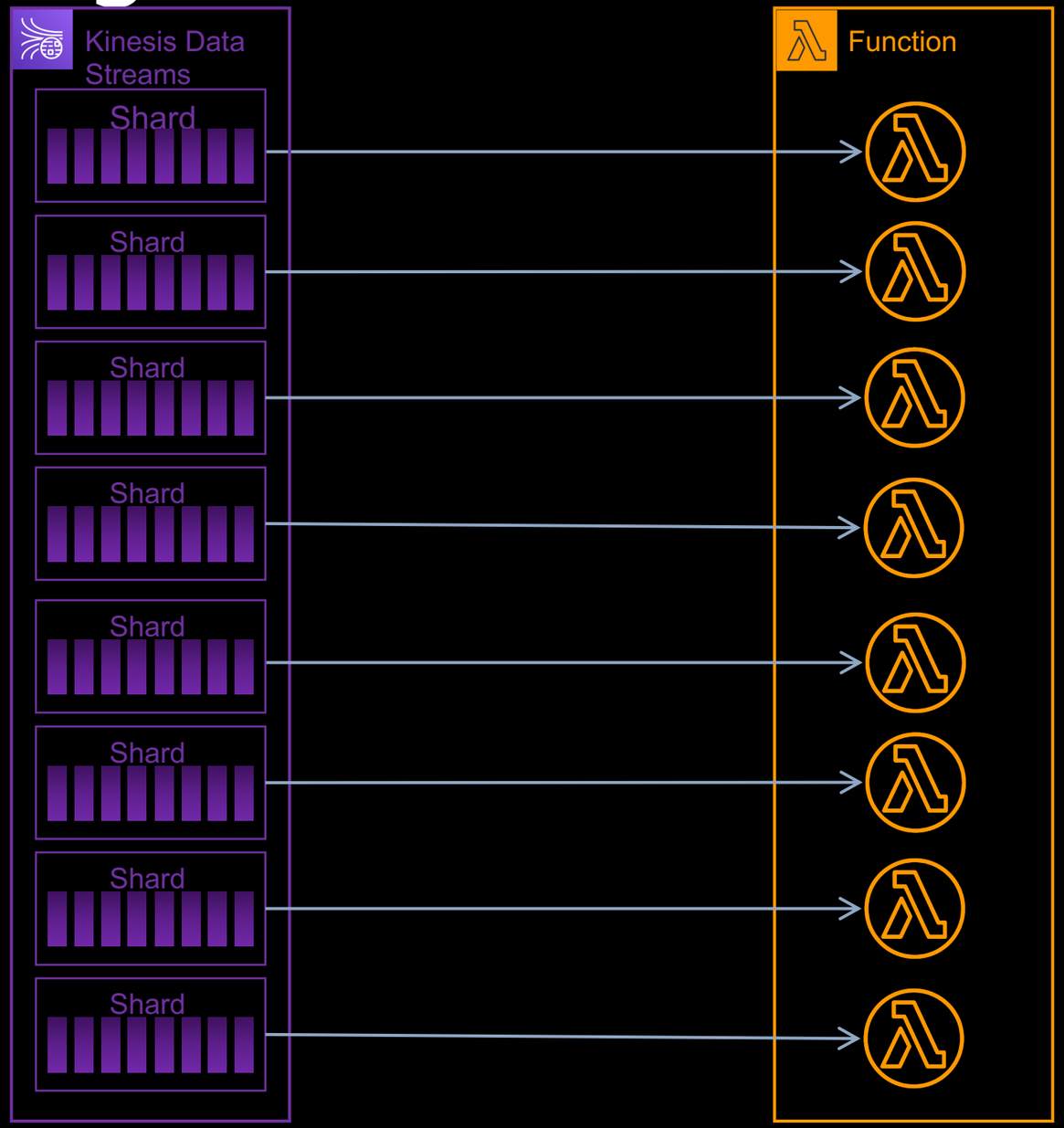
Kinesis Data Streams scaling

```
aws kinesis update-shard-count --stream-name reinvent19-01
--target-shard-count 8 --scaling-type UNIFORM_SCALING
{
    "StreamName": "reinvent19-01",
    "CurrentShardCount": 4,
    "TargetShardCount": 8
}
```



Kinesis Data Streams scaling

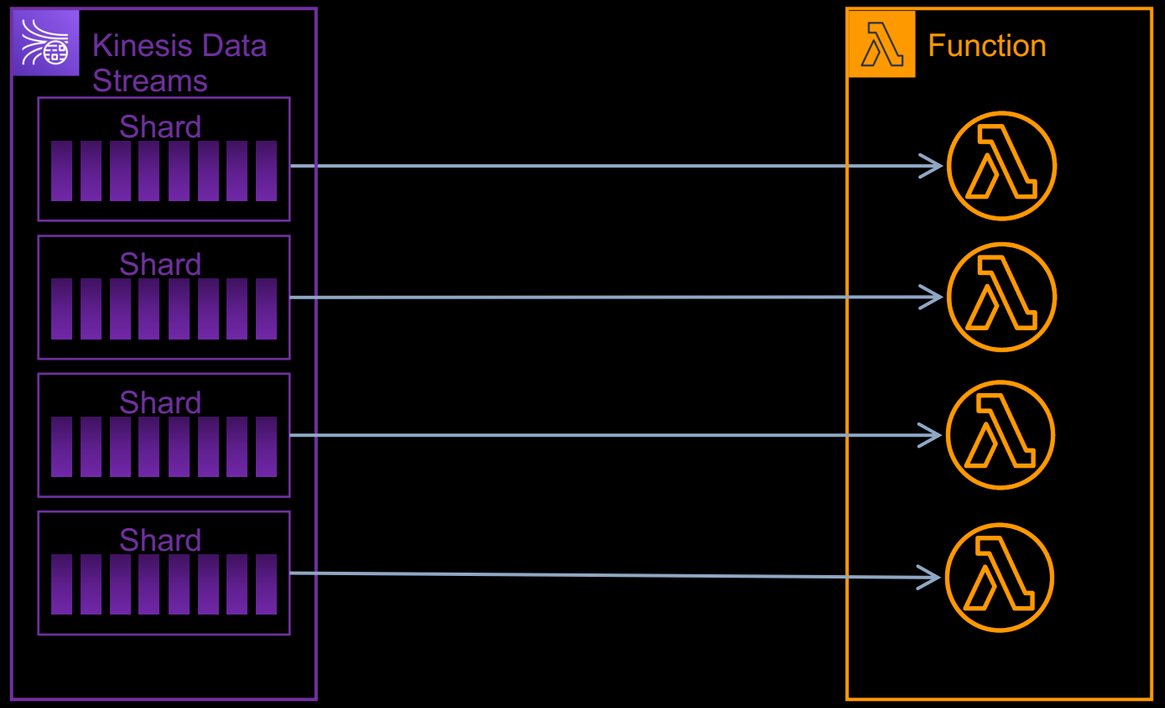
```
aws kinesis update-shard-count --stream-name reinvent19-01
--target-shard-count 8 --scaling-type UNIFORM_SCALING
{
    "StreamName": "reinvent19-01",
    "CurrentShardCount": 4,
    "TargetShardCount": 8
}
```



Kinesis Data Streams scaling

UpdateShardCount API Limits

- Scale more than twice per rolling 24-hour period per stream
- Scale up to more than double your current shard count for a stream
- Scale down below half your current shard count for a stream
- Scale up to more than 500 shards in a stream
- Scale a stream with more than 500 shards down, unless the result is less than 500 shards
- Scale up to more than the shard limit for your account



Most of these limits can be adjusted through AWS Support

Throughput considerations

Kinesis max message ingest per shard

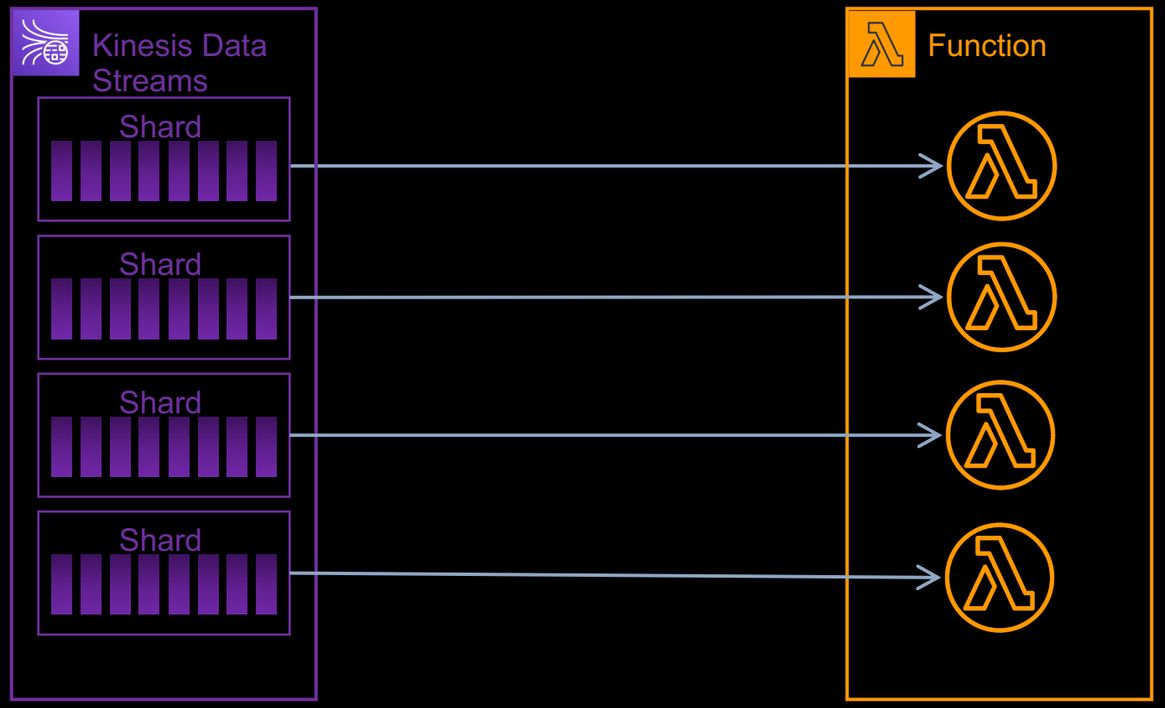
MIN(1MiB/average record size, 1,000)

Lambda maximum messages: steady state

MIN(1MiB/average record size, 1,000)

Lambda maximum messages: edge cases

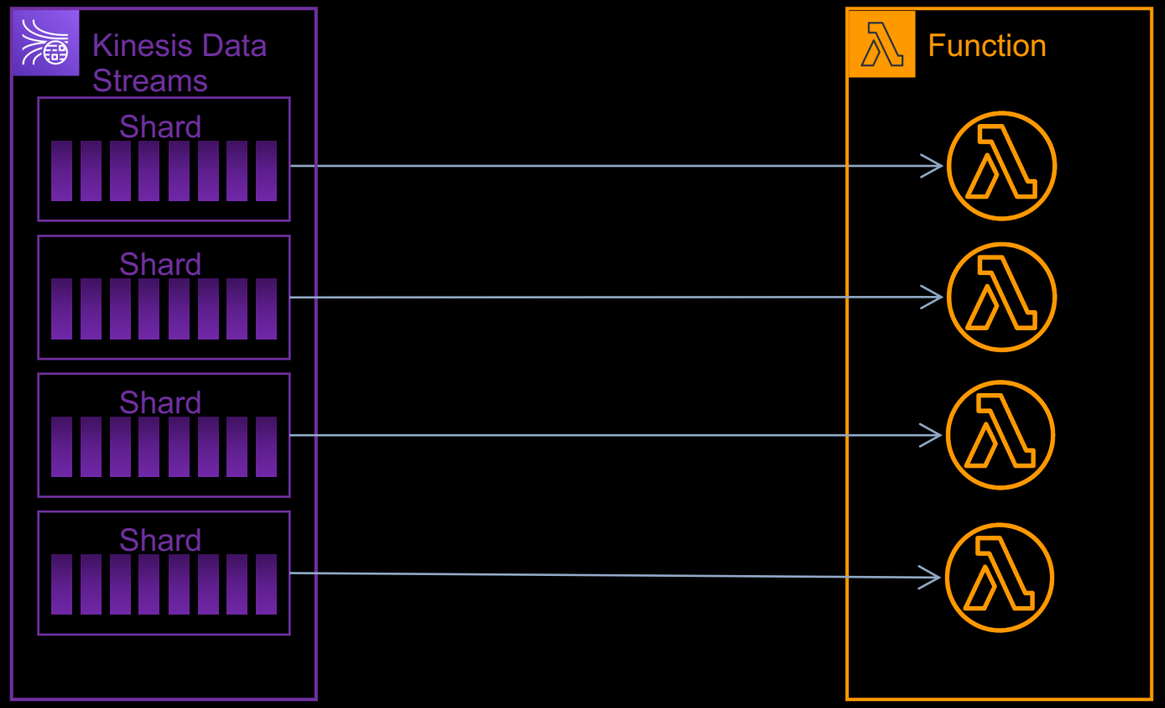
MIN(6MiB/average record size, 10,000)



Kinesis Data Streams scaling

Best practices

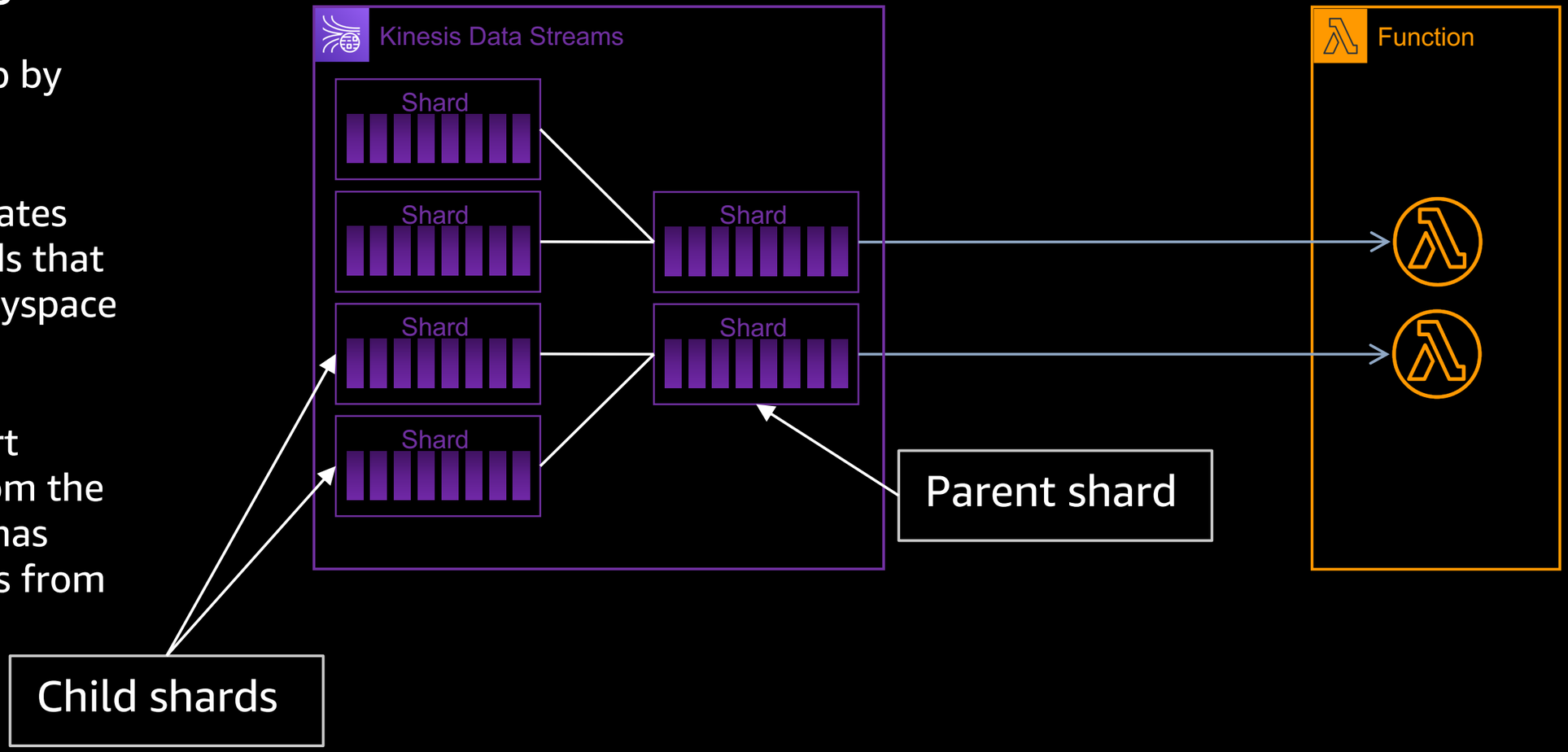
- Auto-scale your shard count using AWS Application Auto Scaling: https://aws.amazon.com/blogs/big-data/scaling-amazon-kinesis-data-streams-with-aws-application-auto-scaling/
- Scale conservatively to leave overhead for bursts of traffic
- Scale your shard count to match your Lambda throughput



Kinesis Data Streams scaling . . . more detail

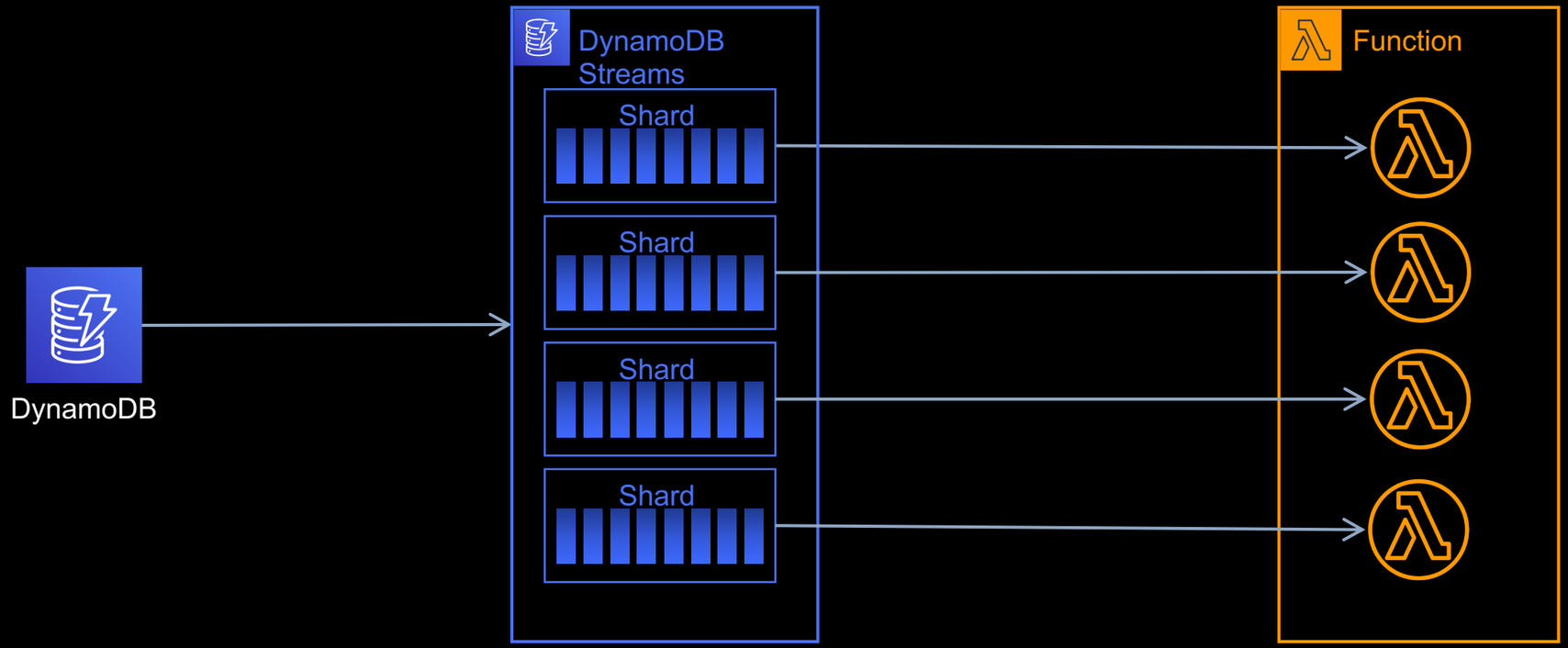
Splitting shards

- The stream scales up by splitting shards
- Splitting a shard creates two, new child shards that split the partition keyspace of the parent shard
- Lambda will not start receiving records from the child shards until it has processed all records from the parent shard



DynamoDB Streams scaling

- Unlike Kinesis, there is no direct control of the number of shards
- The number of shards depends on the volume of data in the table, as well as the amount of read and write throughput provisioned for the table



DynamoDB on-demand vs. provisioned capacity

On-demand

- Pay per request
- Scaling is completely managed
- Good for variable, bursty, workloads
- Matches very well with many Serverless workloads

Provisioned capacity

- Set read capacity (RCU) and write capacity (WCU)
- Optionally auto-scale
- Good for steady-state workloads

DynamoDB on-demand scaling

DynamoDB on-demand supports large scaling spikes

From the DynamoDB docs –

> DynamoDB tables using on-demand capacity mode automatically adapt to your application's traffic volume. On-demand capacity mode instantly accommodates up to double the previous peak traffic on a table. For example, if your application's traffic pattern varies between 25,000 and 50,000 strongly consistent reads per second where 50,000 reads per second is the previous traffic peak, on-demand capacity mode instantly accommodates sustained traffic of up to 100,000 reads per second. If your application sustains traffic of 100,000 reads per second, that peak becomes your new previous peak, enabling subsequent traffic to reach up to 200,000 reads per second.

In general, when a table's ability to serve a set amount of traffic doubles, so does the number of shards in the stream

DynamoDB on-demand scaling

Key takeaways

- On-demand mode has the ability to scale quickly
- Make sure Lambda can match that scaling
  - Monitor Lambda concurrency
    - Alarm on concurrent executions at an account level
    - As well as on any functions that have a limit/reservation

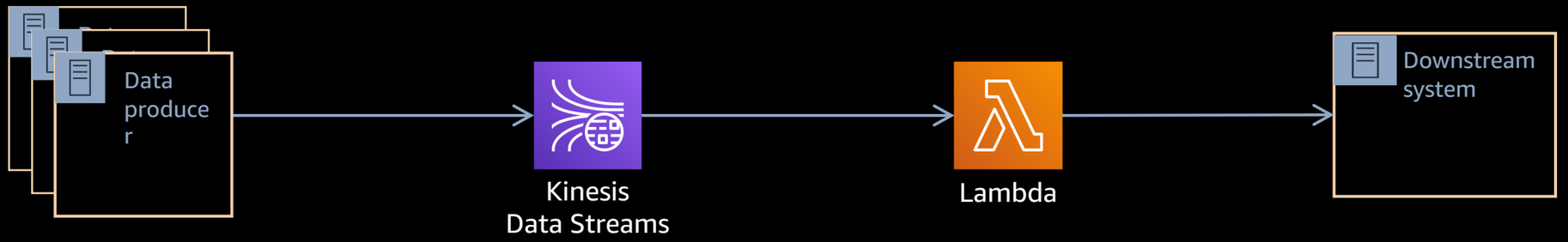
Monitoring



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Kinesis Data Streams monitoring



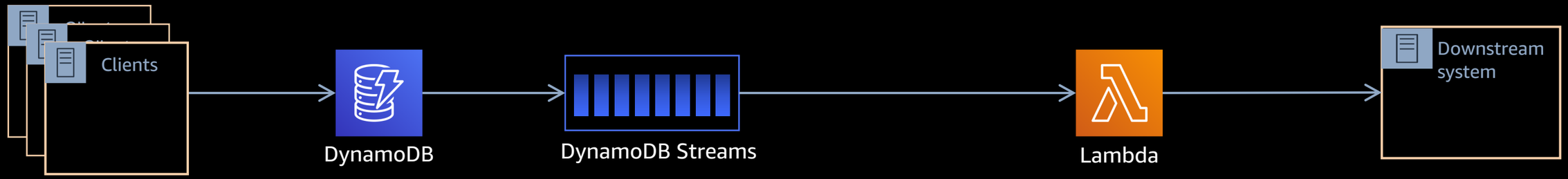
Kinesis metrics/alarms

- GetRecords.IteratorAgeMilliseconds
- IncomingRecords/IncomingBytes
- ReadProvisionedThroughputExceeded
- WriteProvisionedThroughputExceeded

Lambda metrics/alarms

- Errors
- IteratorAge
- Throttles
- Duration

DynamoDB Streams monitoring



DynamoDB Streams metrics/alarms

- ReturnedRecordsCount/ReturnedBytes
- UserErrors

Lambda metrics/alarms

- Errors
- IteratorAge
- Throttles
- Duration

Common issues

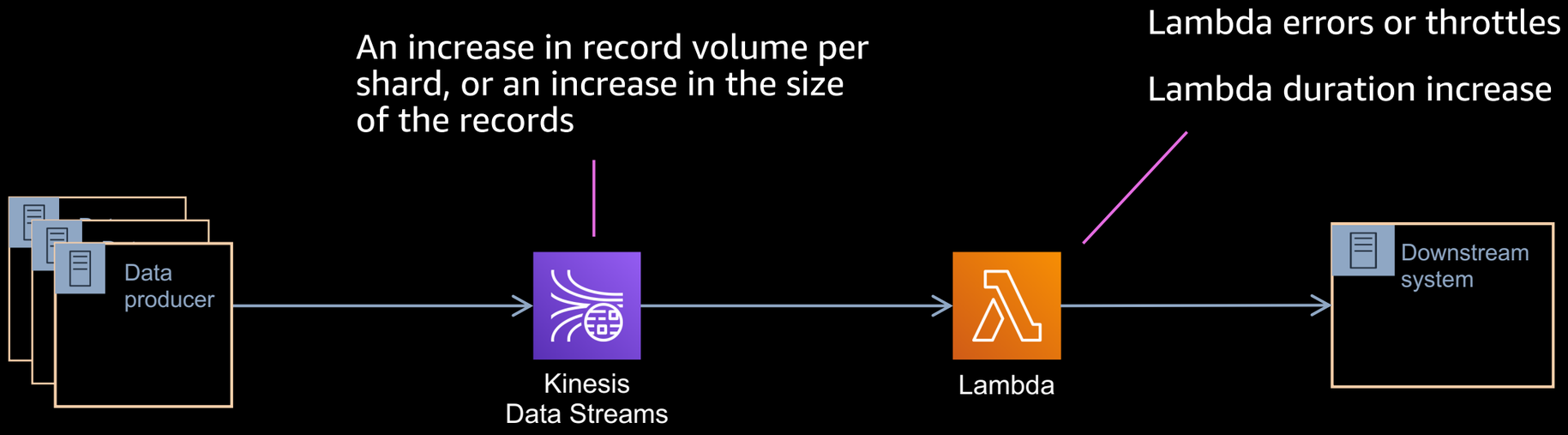
- Kinesis Data Streams
  - IteratorAge is growing rapidly
  - ReadProvisionedThroughputExceeded throttles
- DynamoDB Streams
  - IteratorAge is growing rapidly
  - Rapid growth in Lambda concurrency

Kinesis Data Streams IteratorAge is growing rapidly

- Initial questions
  - How many Lambda functions subscribed to the stream?
  - Does the Lambda function show any errors?
  - Does the Lambda function show any throttles?
  - Is there a large increase in Kinesis Data Streams metrics IncomingRecords or IncomingBytes?

Kinesis Data Streams IteratorAge growing rapidly

Potential causes



Kinesis Data Streams IteratorAge is growing rapidly

- Solutions
  - Update Lambda to log records causing errors, and return successfully
  - Does the Lambda function show any errors?
  - Does the Lambda function show any throttles?
  - Is there a large increase in Kinesis Data Streams metrics IncomingRecords or IncomingBytes?
  - Scale Lambda concurrency with parallelization factor

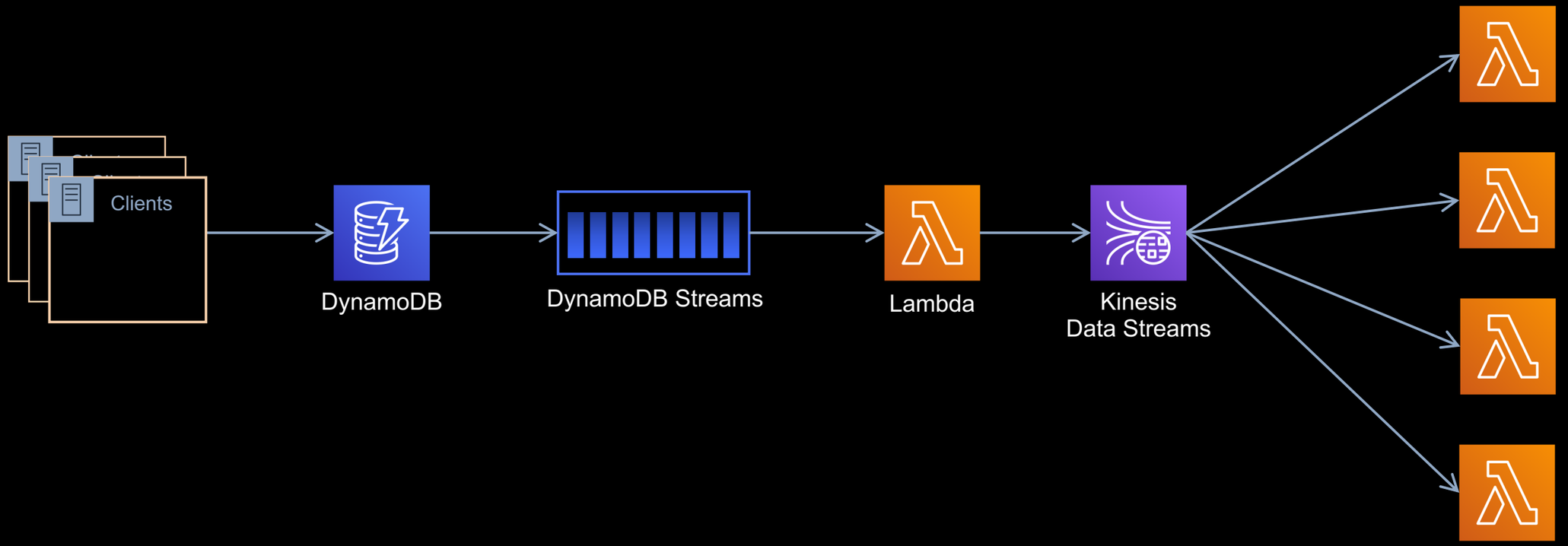
Kinesis Data Streams ReadProvisionedThroughputExceeded

- The 5 read/sec or 2MiB/sec limit is being hit
  - Use Enhanced-Fan-Out or remove one or more subscribers
    - Remember that Kinesis Data Firehose and Kinesis Data Analytics are subscribers as well

DynamoDB Streams IteratorAge is growing rapidly

- Initial questions
  - How many Lambda functions subscribed to the stream?
  - Does the Lambda function show any errors or throttles?
  - Does the Lambda function show an increase in duration?
  - Is there a large increase in the DynamoDB Table Write (WCU) Metrics?
  - Is there a large increase in the DynamoDB Streams metrics ReturnedRecordsCount or ReturnedBytes?

DynamoDB Streams fan-out



Performance and optimization



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Performance

- What matters to your application?
  - End-to-end latency
  - Overall cost
- Kinesis Enhanced Fan–Out (EFO)
- Small messages in Kinesis
  - Agg/DeAgg libraries
- Low throughput streams – batch window to the rescue

Lambda supports Kinesis Data Streams Enhanced Fan-Out (EFO) and HTTP/2 for faster streaming

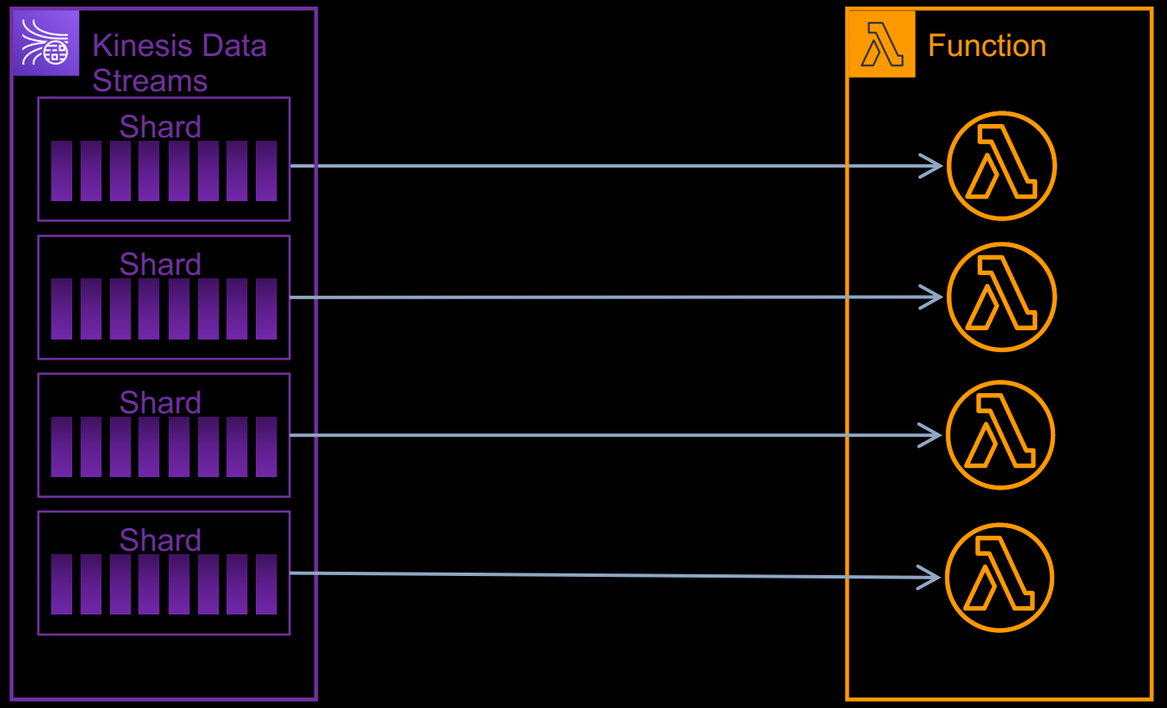


EFO allows customers to scale the number of functions reading from a stream in parallel while maintaining performance

HTTP/2 data retrieval API improves data delivery speed between data producers and Lambda functions by more than 65%

Parallelization factor

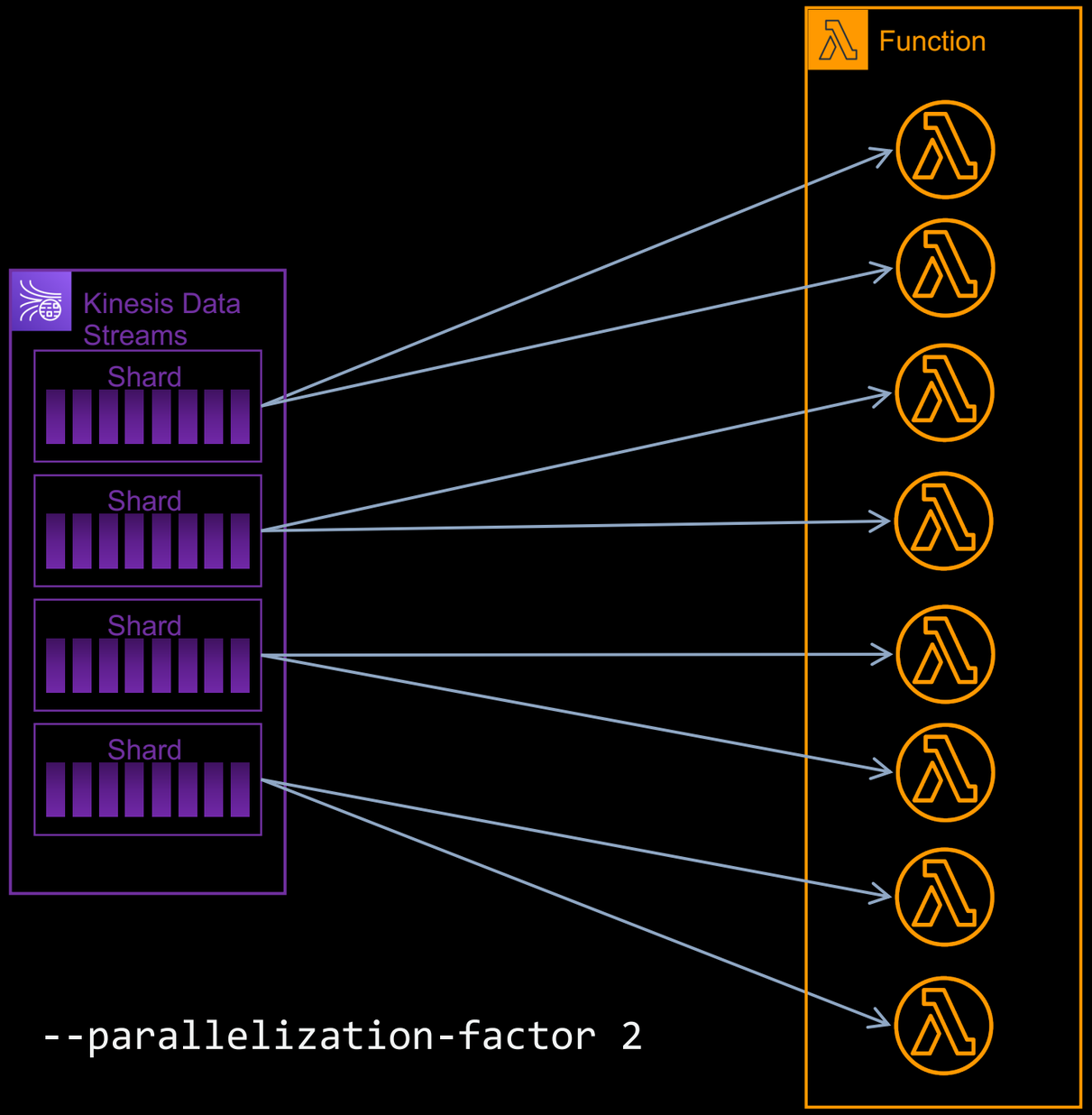
- Adds Lambda parallelization per shard
- Setting of 1 is the same as the current behavior; maximum setting is 10
- Batching via partition keys to maintain in-order processing per partition key
- Works with both Kinesis Data Streams and DynamoDB Streams




```
--parallelization-factor 1
```

Parallelization factor

- Adds Lambda parallelization per shard
- Setting of 1 is the same as the current behavior; maximum setting is 10
- Batching via partition keys to maintain in order processing per partition key
- Works with both Kinesis Data Streams and DynamoDB Streams



Optimizing small messages in Kinesis

- Kinesis per Shard Write Limits
  - 1MiB/sec or 1,000 messages/sec
  - With high volumes of small messages, you can reach the 1,000 messages/sec limit easily
  - This leads to lower throughput per shard and higher costs

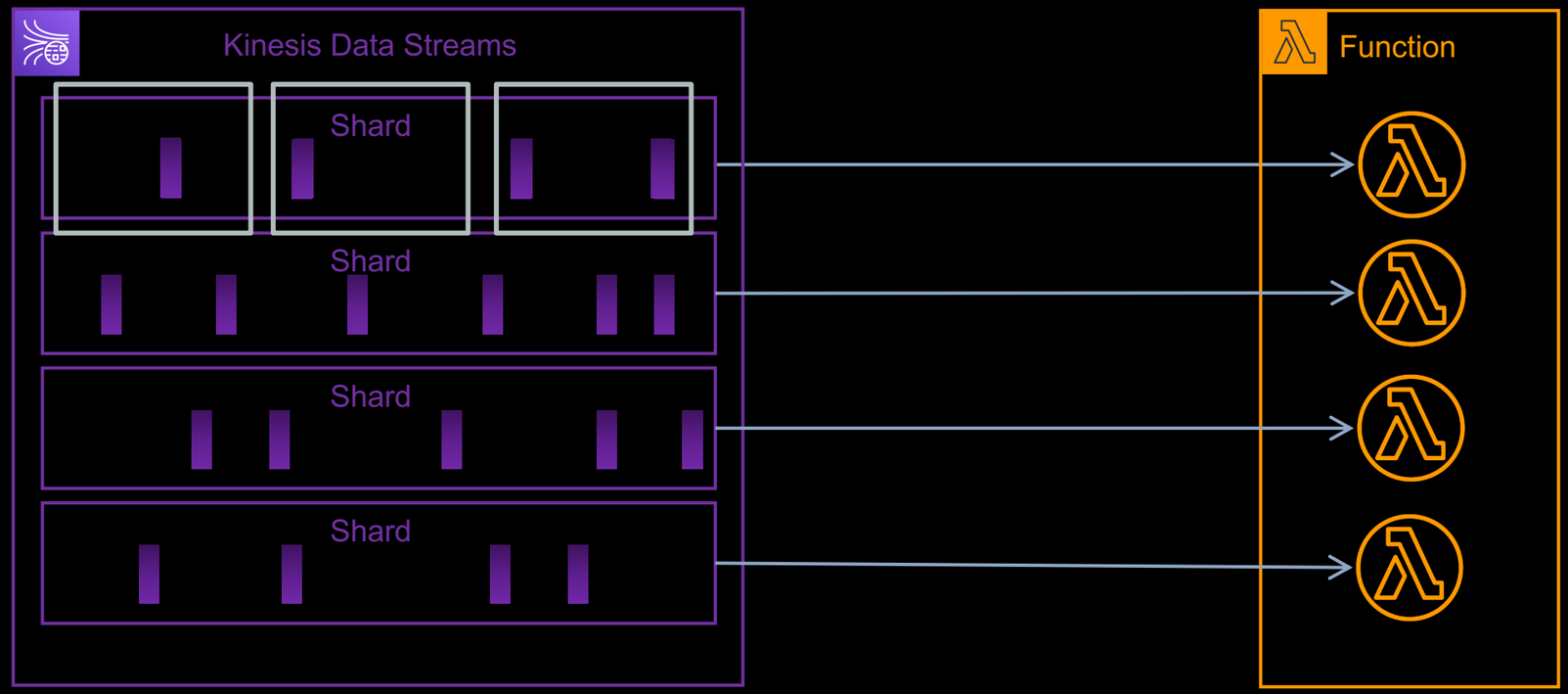
Aggregation is the answer!

Aggregation / De-aggregation options

- Producer side
  - Kinesis Producer Library (KPL) (https://github.com/awslabs/amazon-kinesis-producer)
  - Kinesis Aggregation Library (https://github.com/awslabs/kinesis-aggregation)
- Consumer side within Lambda
  - Kinesis Aggregation Library (https://github.com/awslabs/kinesis-aggregation)
    - Java, node.js, and python versions available

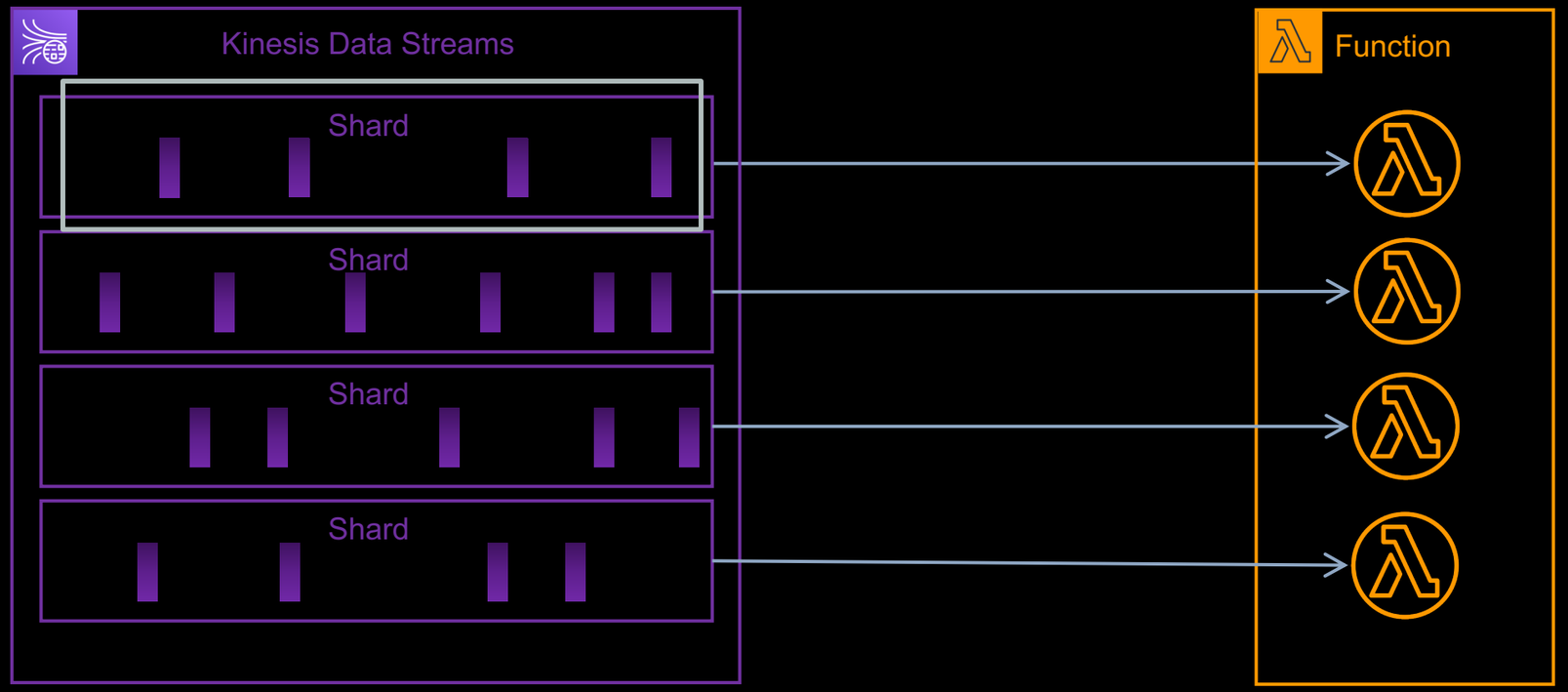
Low throughput streams

Lambda triggered with very small batches
Leads to higher cost per message
For archiving workloads, the resulting payload is too small



Batch window

- Additional knob to tune the stream trigger
  - Set a time to wait before triggering; max 5 minutes, set in seconds
  - Batch size is still respected and will trigger on full batches before the batch window is up
  - Works for both Kinesis Data Streams and DynamoDB Streams triggers



A summary of new controls

	Asynchronous	Kinesis/DynamoDB	Kinesis EFO
Record/event age	Yes	Yes	No
Retry attempt	Yes	Yes	No
On-success destination	Yes	No	No
On-failure destination	Yes	Yes	No
Parallelization factor & batch window	N/A	Yes	Yes
Bisect on function error	N/A	Yes	No

Check out these other sessions!

SVS407: Deep dive on streaming with Amazon Kinesis and AWS Lambda

SVS303: Scalable serverless event-driven architectures with SNS, SQS & Lambda

Thank you!

Adam Wagner
Solutions Architect
AWS

@oathead

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.