AWS Kinesis Data Stream

Haoliang Yu @ Esri

Arlington VA AWS Meetup

03/15/2018

About me

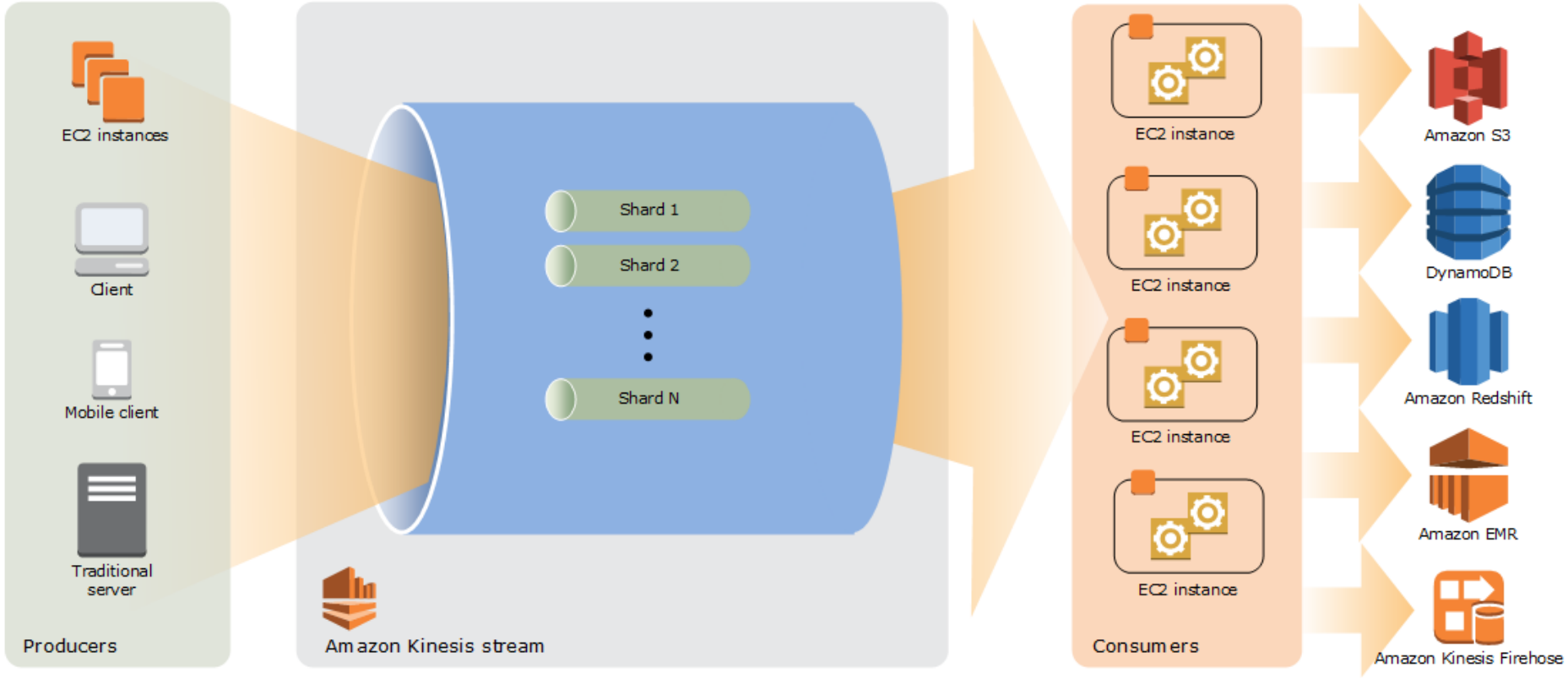
- Haoliang Yu, or Joe
- Software engineer at Esri DC R&D
- JavaScript + X
- At work: build the open data search service with AWS and Elastic Cloud
- At home: build open-source libraries and websites
- github.com/haoliangyu

AWS Kinesis Data Stream

A data stream service that provides:

- real-time stream data ingesting and processing
- concurrent data processing without conflict
- durable data storage
- easy scaling (up or down)

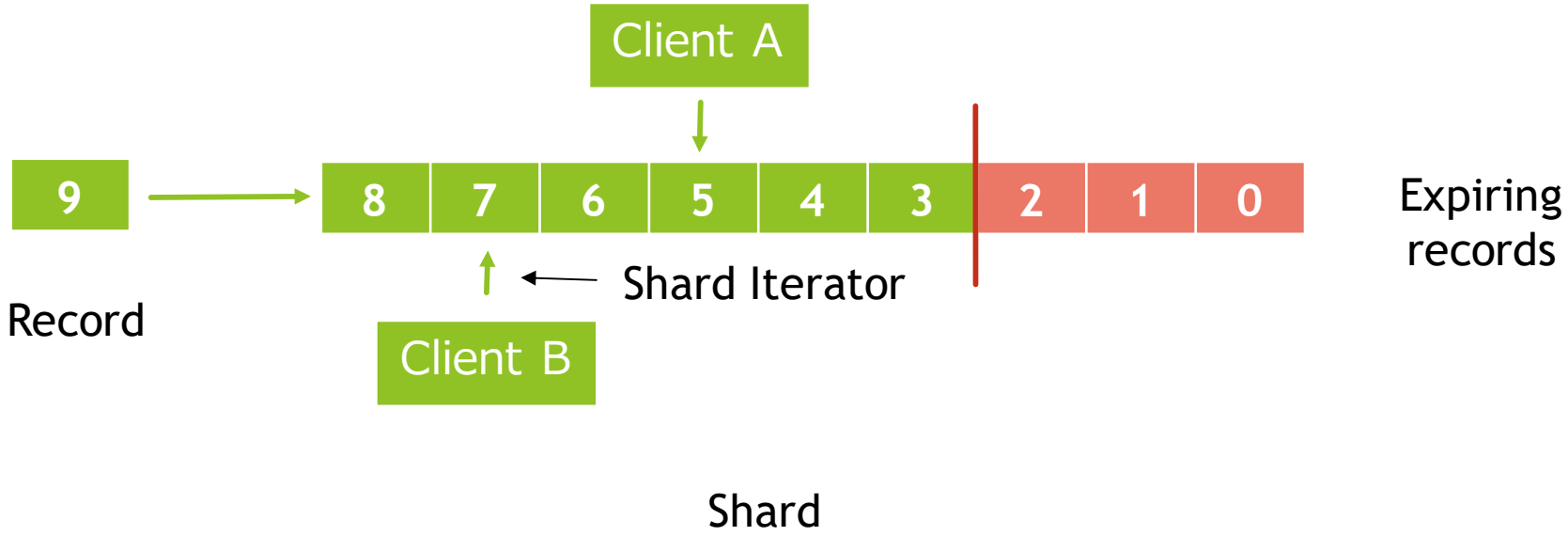
AWS Kinesis Data Stream



- Component
  - Producer
  - Stream
  - Client
- Stream
  - Shard
  - Shard iterator
- Record
  - Partition key
  - Sequence number
  - Arrival time

AWS Kinesis Data Stream

A closer look on a shard



Price Model

- Pay as you use, but no free use
- Price elements
  - Number of shards (1 MB/s input, 2 MB/s output per shard)
  - Number of PUT payloads (pay per unit, 25KB)
  - Extended data retention (payer per shard hour, up to 7 days)
- Getting data from a stream is free!

Compared to SQS

	Kinesis Data Stream	SQS
Similarity	• A record channel • Records in order • Charged by record volume	• A message channel • Messages mostly in order • Charged by message volume
Difference	• Records can be routed to different shards/apps • Durable • Records can be read many times • Scale with shard number • Charged by shard number	• Apps consume the same queue • Not durable • Messages are read once • Scale magically • Charged by API requests

Use Cases

- Data recovery
  - If a data storage server goes down, we can use the kinesis stream to restore recent changes with a backup
- Stream partition
  - Route records to different shards based on the partition key
- Multiple stream consumers
  - Save a log stream into S3 and use another application to analyze the same stream
  - Use a separate customer to writer data with a newer format to a separate server, while keeping the data processed in the old server

Develop with AWS SDK

Low-level development – put record

```
const AWS = require('aws-sdk');
const kinesis = new AWS.Kinesis();

kinesis
  .putRecord({
    // stream record in buffer
    Data: new Buffer('your record in string'),
    // AWS hash the partition key and use the value
    // to determine the shard to add
    PartitionKey: 'go to which shard',
    StreamName: 'kinesis stream name'
  })
  .promise()
  .then(() => {
    // do something
  });
```

Developers need to decide how to store records in a stream with multiple shards.

Develop with AWS SDK

Low-level development – get records

```
const AWS = require('aws-sdk');
const kinesis = new AWS.Kinesis();

kinesis
  .getShardIterator({
    ShardId: '123',
    ShardIteratorType: 'AFTER_SEQUENCE_NUMBER',
    StreamName: 'nest',
    StartingSequenceNumber: '1'
  })
  .promise()
  .then((result) => {
    const iterator = result.ShardIterator;
    return kinesis.getRecords({
      ShardIterator: 'iterator',
      Limit: 10
    });
  })
  .promise()
  .then((result) => {
    const records = result.Records;
    const nextIterator = result.NextShardIterator;
  });
```

Developers need to create shard iterators and maintain their state for distributed applications.

Develop with KPL/KCL

- Kinesis Producer Library (KPL)
  - Automatically retry failed records
  - Group records to reduce network I/O (thought delay is added)
  - Available only for Java
- Kinesis Client Library (KCL)
  - Automatically store and manage the iterator state for each shard
  - Automatically create workers with the record processing function provided by the developer and destroy them when not necessary
  - Available for Java, Node.js, Python, .Net, Ruby

Final

- Q & A
- Send any question to haoliang.yu@outlook.com
- Slides available at github.com/haoliangyu/aws-kinesis-talk
- Thank you!