
AWS Summit
AWS技术峰会2015 · 上海
亚马逊
aws

Amazon Kinesis在实时数据处理业务模型中的应用

发掘实时数据的价值

雷洋

主要议程

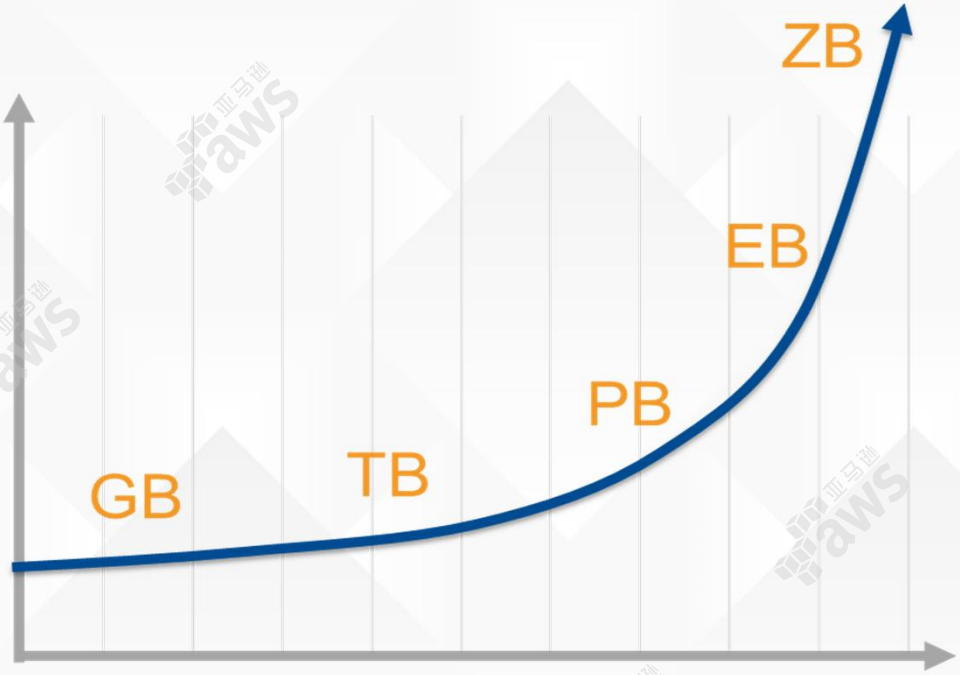
- 实时数据处理的行业背景
- Amazon Kinesis数据流处理服务介绍
- Amazon Kinesis的典型应用场景
- 用户案例分享
- 关于将网站实时访问数据可视化的演示
- 现场提问

信息大爆炸

用户数据分析 = 商业价值

行业 / 用户场景	加速数据收集和整合	长期监控 / KPI 绩效评估	可执行能力
广告，市场分析	整合广告数据	广告数据，比如覆盖率，转换率，收益率，页面评分等等	用户参与行为分析，优化广告竞标和引擎
在线购物，游戏	整合在线用户的交互数据	在线用户或是App内用户参与数据监控，例如页面访问数据，点击率	用户点击数据分析，优化推荐引擎
电子化财务服务	提升用户在银行交易网站的体验	财务市场数据	进行违规滥用行为监控, 收益风险评估, 市场交易操作审计
物联网，可穿戴设备	健身器材 , 车载感应装置, 测量数据	可穿戴设备的日常监控数据以及在线展示	智能操控

数据爆发式增长



- **IT / 应用程序日志**
  IT基础设施日志, 测量, 审计或是变更日志
- **网络站点 / 移动应用 / 网络广告**
  点击数据, 用户交互数据
- **社交媒体, 论坛, 博客**
  消息, 论坛帖子, 博客文章
- **探测器数据**
  气象数据, 智能电网, 可穿戴设备

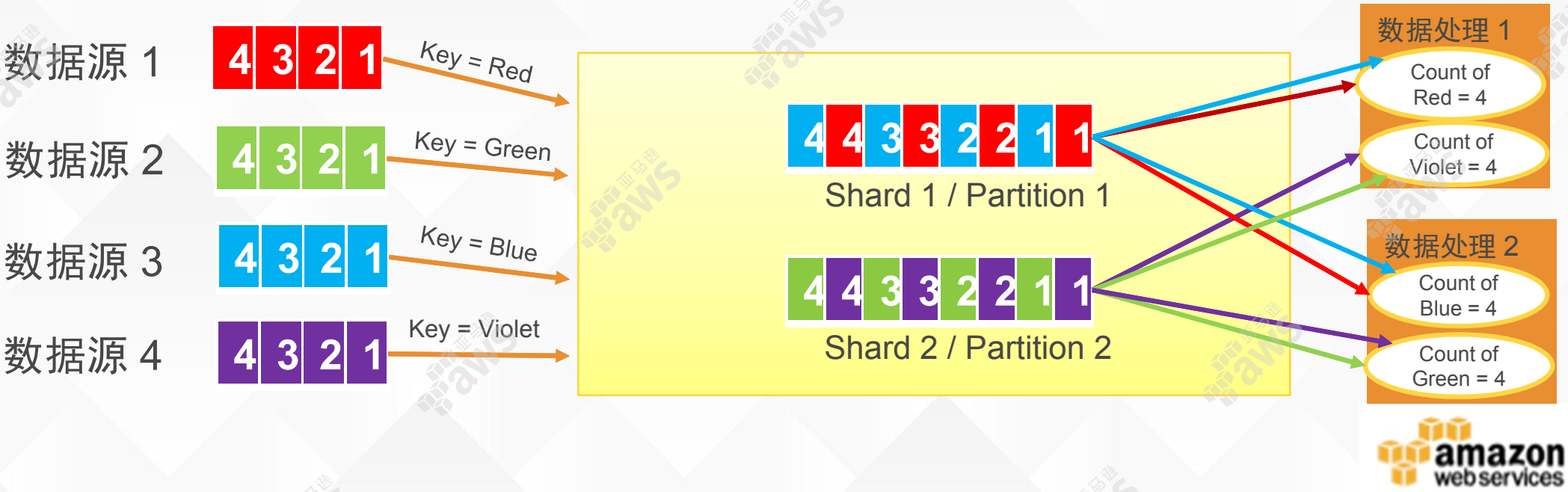
数据时效性和响应时间

- 基于算法的交易系统
- 广告实时竞标
- 常见的物联网数据处理
- 基础设施的实时监控
- 在线购物推荐
- 大部分商业智能分析
- 社交媒体公众账号在线监听

< 10 毫秒
< 100 毫秒
< 5 到10秒
< 1 分钟
< 5 到 20 秒钟
< 30 分钟

为什么需要数据流服务

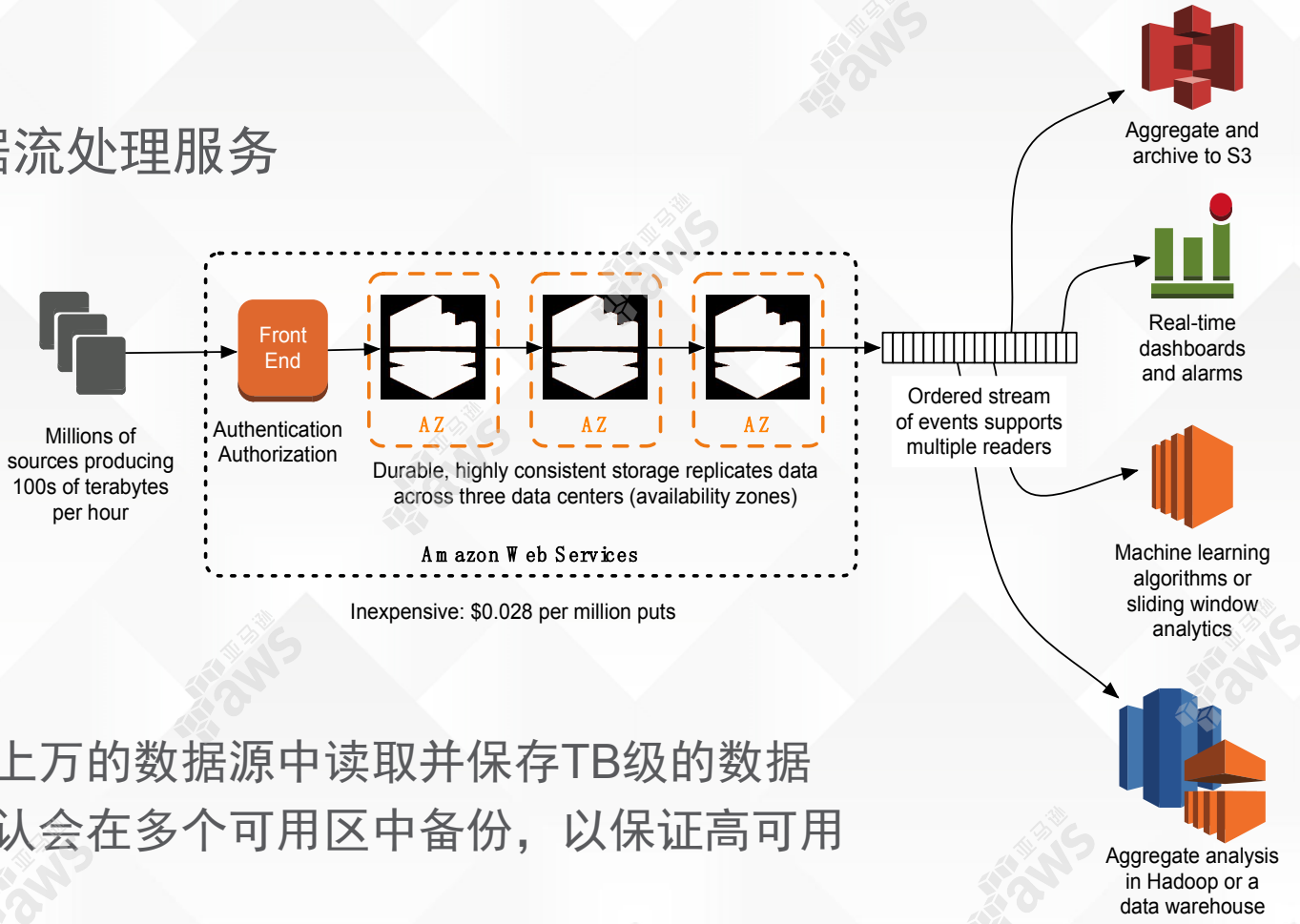
- 将生产消费端解耦合
- 持久化的数据缓存
- 收集对接多个数据流
- 保持数据的原始顺序
- 与MapReduce框架的完美融合
- 满足并行数据消费



amazon web services

Amazon Kinesis

全托管的实时大规模数据流处理服务



- Kinesis每小时能从成千上万的数据源中读取并保存TB级的数据
- Kinesis中处理的数据默认会在多个可用区中备份，以保证高可用

Amazon Kinesis 数据流服务

Amazon Kinesis 提供了三种数据流处理的方式

- Amazon Kinesis Streams
- Amazon Kinesis Firehose
- Amazon Kinesis Analytics

- Amazon Kinesis Streams
- Amazon Kinesis Firehose
- Amazon Kinesis Analytics

Amazon Kinesis Streams

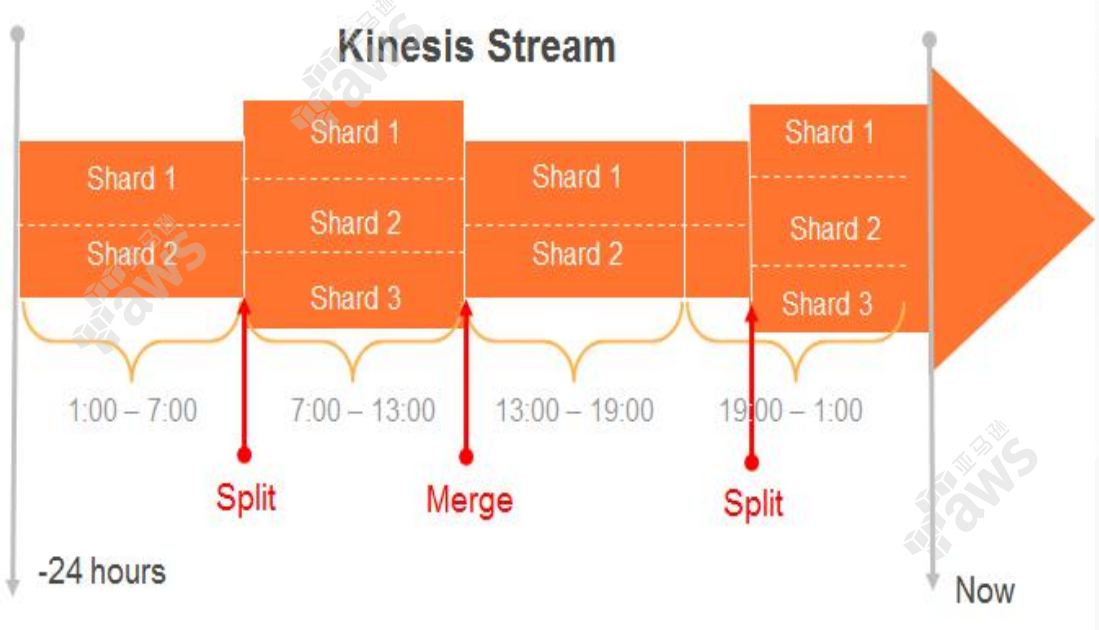
帮助用户轻松构建数据流处理应用

1. 发送用户点击数据　2. Kinesis存储数据供后端分析　3. 程序分析后给用户提供推荐　4. 用户看到个性化推荐内容

- **易维护:** 只需要设置Shard的数量就可以创建Stream, 之后还可以根据业务规模进行扩展.
- **快速搭建实时的数据处理应用:** 使用Kinesis Client Library (KCL), Apache Spark/Storm, AWS Lambda或是其他解决方案来快速搭建大数据流处理平台.
- **低成本:** 适用于不同规模的业务需求.

Amazon Kinesis Streams

- Streams是由Shards组成的
- 每个Shard可以接收1MB/秒的数据写入, 最高支持1000 交互/秒
- 每个Shard支持最高2 MB/秒数据读取
- 所有数据最长可以保留7天时间
- 可以通过分割和合并Shards来弹性扩展Kinesis stream
- 可以在7天的时间窗口内回放数据



Amazon Kinesis Firehose

将大规模的流数据装载到静态存储S3和数据仓库Redshift

收集并发送数据到 Firehose

Firehose 将数据源源不断的装载到S3 和Redshift

使用原有的商业智能工具来分析数据流

- **零运维:** 不需要编写应用程序和维护基础设施即可以实现将数据流导入到S3和Redshift数据仓库
- **直接与存储集成:** 通过简单设置就可以实现在60秒的较小延时内将数据流加密压缩，然后存储到目的地
- **弹性扩展:** 在不需要人为参与的情况下，可以根据数据规模自动弹性扩展

Firehose如何传数据到S3

- 每一个Kinesis delivery stream对应一个S3存储桶
- Buffer Size / Interval 用来控制存储在S3上的对象文件的大小和传输文件的频率
  - Buffer size – 1 MB 到 128 MB可设置
  - Buffer interval - 60 到 900 秒可设置
  - Firehose 将多条记录整合成一个对象文件
  - 这两个条件任何一个满足都会触发数据的传送
- (可选) 对数据进行压缩
  - GZIP, ZIP, SNAPPY
  - 存放到S3上的文件会比写入到Firehose的内容要少，因为它是经过压缩的
  - 目前Amazon Redshift只支持GZIP一种压缩格式

从Kinesis Firehose到Redshift数据仓库

只需要两个步骤：

1

- 使用用户提供的S3存储桶作为临时数据存储地
- 使用S3仍然是加载大规模数据到Redshift的最有效的方式
- 数据存储在S3保证了安全性及可用性

2

- Firehose 以同步的方式执行用户提供的拷贝命令，将数据从S3加载到Redshift中
- 前一个拷贝命令执行完成，在收到Redshift的确认后Firehose会执行下一个拷贝命令，以这种方式它持续的执行拷贝命令

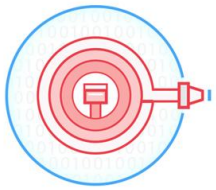
如何在Kinesis Streams和Firehose之间选择

Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data

Amazon Kinesis Streams

适用于用户自己开发程序或是使用现有的数据流处理框架的场景，Kinesis作为数据流的缓存，这种模式可以保证最小的处理时延



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift

Amazon Kinesis Firehose

适用于用户已经在使用基于S3或是Redshift的数据分析工具的场景，用户不需要开发和额外的维护就可以继续使用这些工具来进行数据分析，延续了工具和经验的持续性，使用Firehose带来大约60秒或是更长时间的延时

Amazon Kinesis Analytics

Announcement Only!

使用您熟悉的SQL语言来轻松实现数据分析

从Kinesis streams中接收数据

在数据流中执行SQL查询

Kinesis Analytics 可以将处理后的数据发送给您的数据分析工具，您可以来创建报警或是对数据做出实时响应

- **SQL查询：**使用您已有的SQL语言技能来分析和处理数据流
- **轻松构建实时数据应用:** 将Amazon Kinesis Analytics与您的数据分析工具集成，可以轻松实现毫秒级的低延时数据处理
- **弹性扩展:** 不需要人工参与即可实现根据流量弹性扩展

Amazon Kinesis 特点

全托管的实时数据流处理服务


易于部署和管理

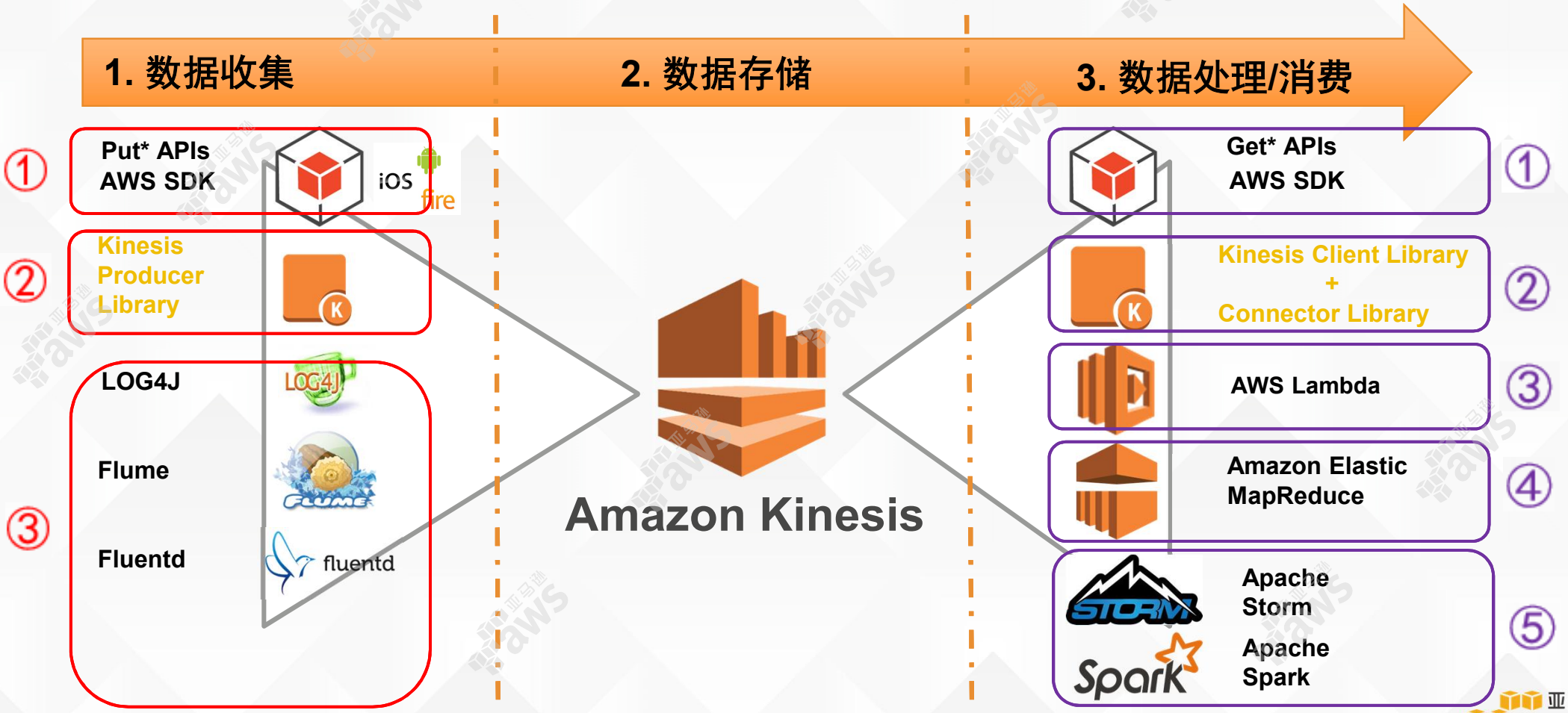

根据业务流量需要弹性扩展


低延时


按需付费，无前期投资


根据不同的业务需要选择不同的服务

Amazon Kinesis 工作模式



Amazon Kinesis 应用场景1: 通过API进行数据读写

A: 通过API来向Kinesis Stream写入数据

```
for (int j = 0; j < 10; j++) {
    // create a request object
    putRecordRequest pR = new PutRecordRequest();
    // specify stream
    pR.setStreamName( myStreamName );
    // wrap data
    pR.setData(ByteBuffer.wrap( String.format( "testData-%d", j).getBytes()));
    pR.setPartitionKey( String.format( "partitionKey-%d", j%5 ));
    pR.setSequenceNumberForOrdering( sn);
    // put data record
    putRecordResult pT = client.putRecord( putRecordRequest );
    sn = pT.getSequenceNumber();
}
```

B: 通过API从Kinesis Stream读取数据

```
List<Record> records;
while (true) {
  //Create new GetRecordsRequest with existing shardIterator.
  //Set maximum records to return to 1000.
  getRecordsRequest getRecordsRequest = new GetRecordsRequest();
  getRecordsRequest.setShardIterator(shardIterator);
  getRecordsRequest.setLimit(1000);
  getRecordsResult result = client.getRecords(getRecordsRequest);
  //Put result into record list. Result may be empty.
  records = result.getRecords();
  try {
      Thread.sleep(1000);
  } catch (InterruptedException exception) {
      throw new RuntimeException(exception);
  }
  //
  shardIterator = result.getNextShardIterator();
}
```

Amazon Kinesis 应用场景2: 用户自定义程序

使用Kinesis Client/Producer Library进行数据读写

- **KCL/KPL的价值**

① 处理重传

② 整合拆分数据记录

③ 设置检查点

④ 应对Reshard场景

⑤ 处理在多个消费者间分发流量

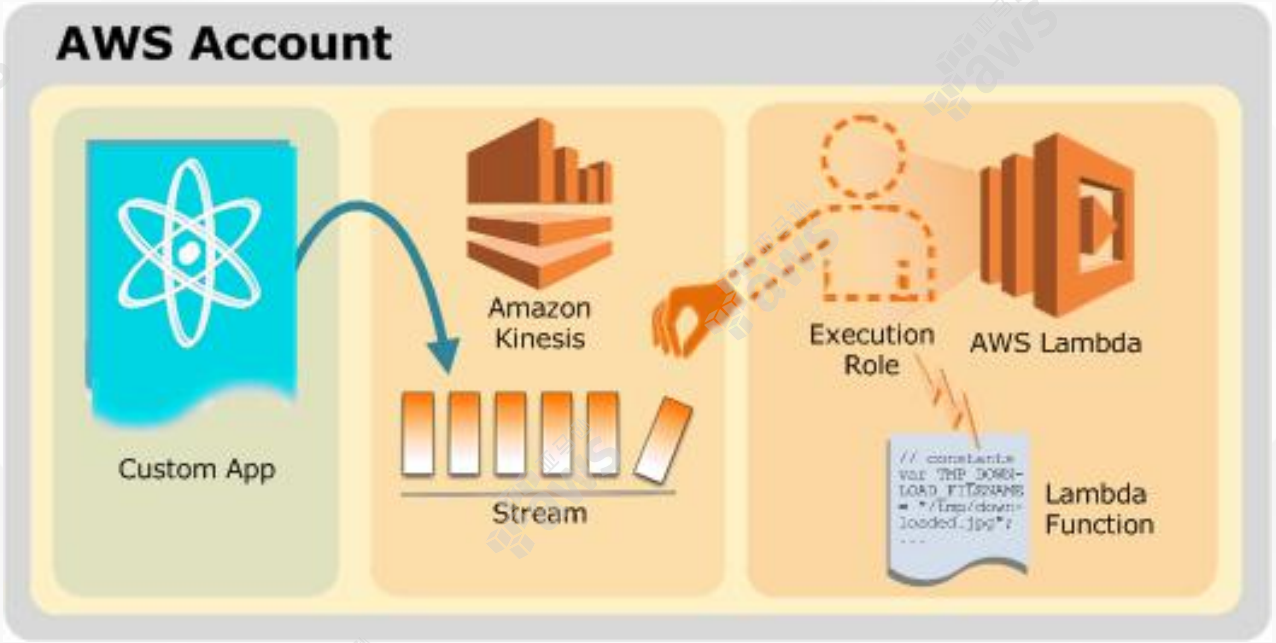
- **完整示例代码**

https://github.com/aws/aws-sdk-java/tree/master/src/samples
https://github.com/awslabs/amazon-kinesis-producer

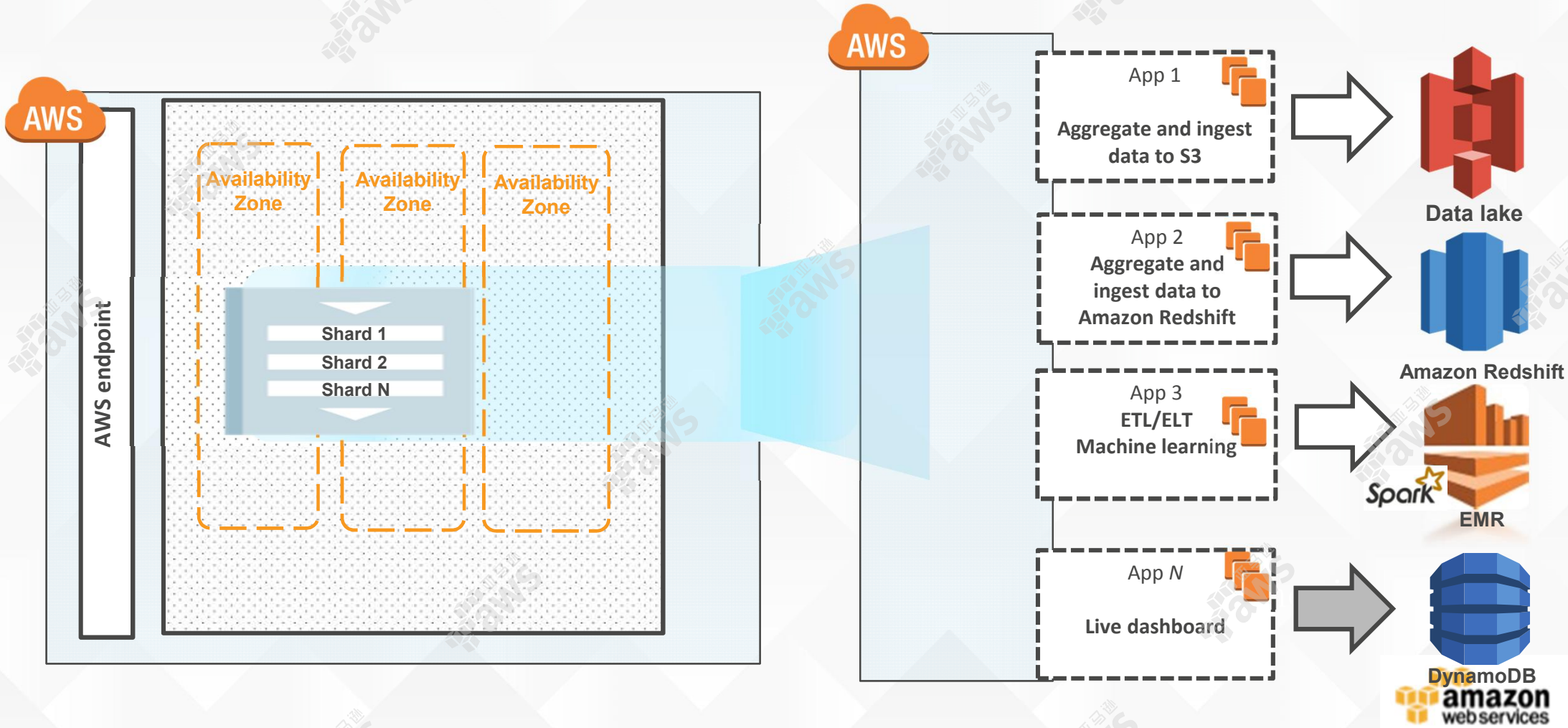
KCL示例代码片段：

```
public class SampleRecordProcessor implements IRecordProcessor {

    private static final Log LOG = LogFactory.getLog(SampleRecordProcessor.class);
    private String kinesisShardId;

    // Backoff and retry settings
    private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
    private static final int NUM_RETRIES = 10;

    // Checkpoint about once a minute
    private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
    private long nextCheckpointTimeInMillis;

    private final CharsetDecoder decoder = Charset.forName("UTF-8").newDecoder();

    public SampleRecordProcessor() {
        super();
    }

    @Override
    public void initialize(String shardId) {
        LOG.info("Initializing record processor for shard: " + shardId);
        this.kinesisShardId = shardId;
    }

    @Override
    public void processRecords(List<Record> records, IRecordProcessorCheckpointer checkpointer) {
        LOG.info("Processing " + records.size() + " records from " + kinesisShardId);

        // Process records and perform all exception handling.
        processRecordsWithRetries(records);
    }
```

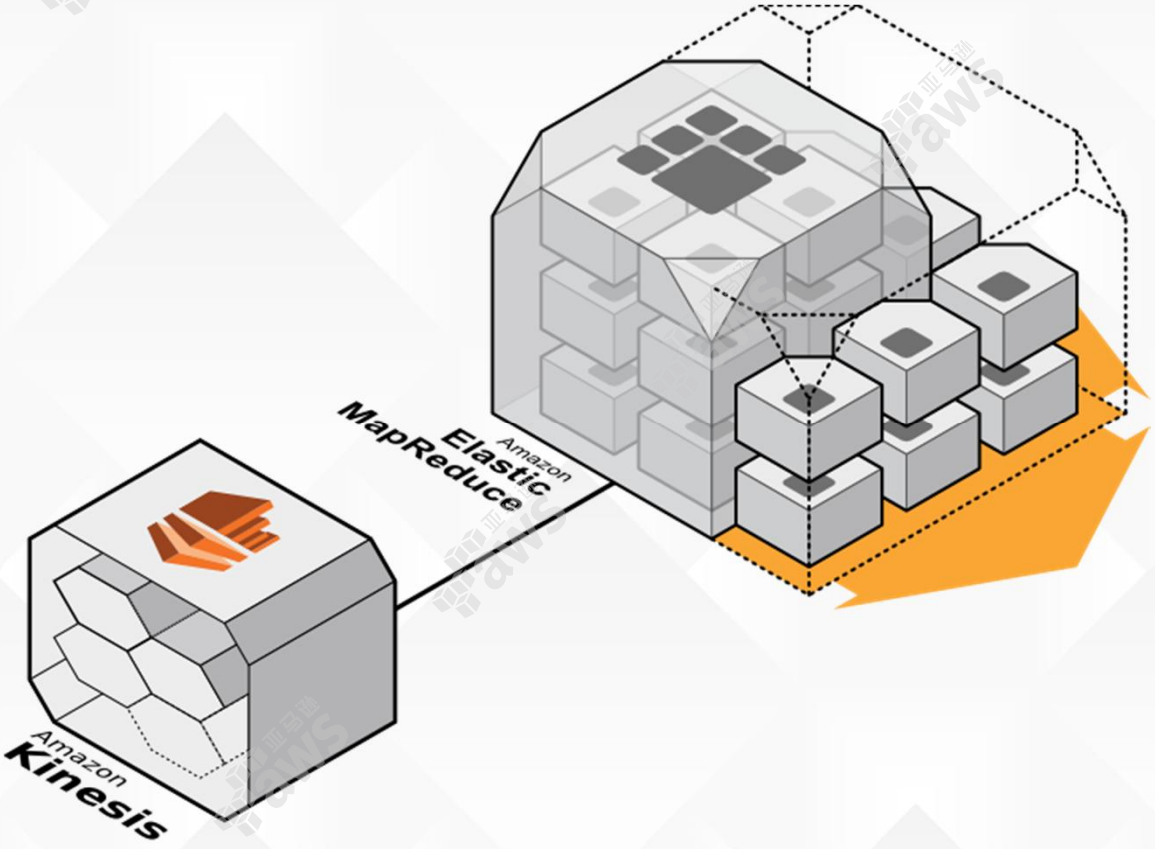
Amazon Kinesis 应用场景3: Lambda事件驱动



Amazon Kinesis 应用场景4



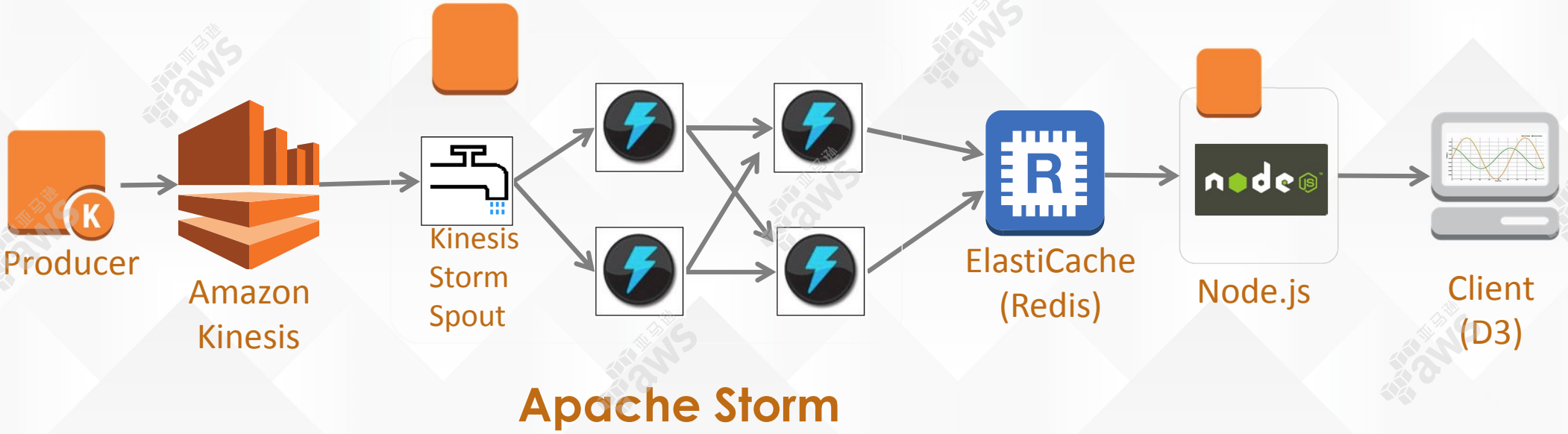
Amazon Kinesis 应用场景4.1: EMR



Amazon Kinesis 应用场景4.1: Hive

```
CREATE TABLE call_data_records (
  start_time bigint,
  end_time bigint,
  phone_number STRING,
  carrier STRING,
  recorded_duration bigint,
  calculated_duration bigint,
  lat double,
  long double
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ","
STORED BY
'com.amazon.emr.kinesis.hive.KinesisStorageHandler'
TBLPROPERTIES("kinesis.stream.name"="MyTestStream");
```

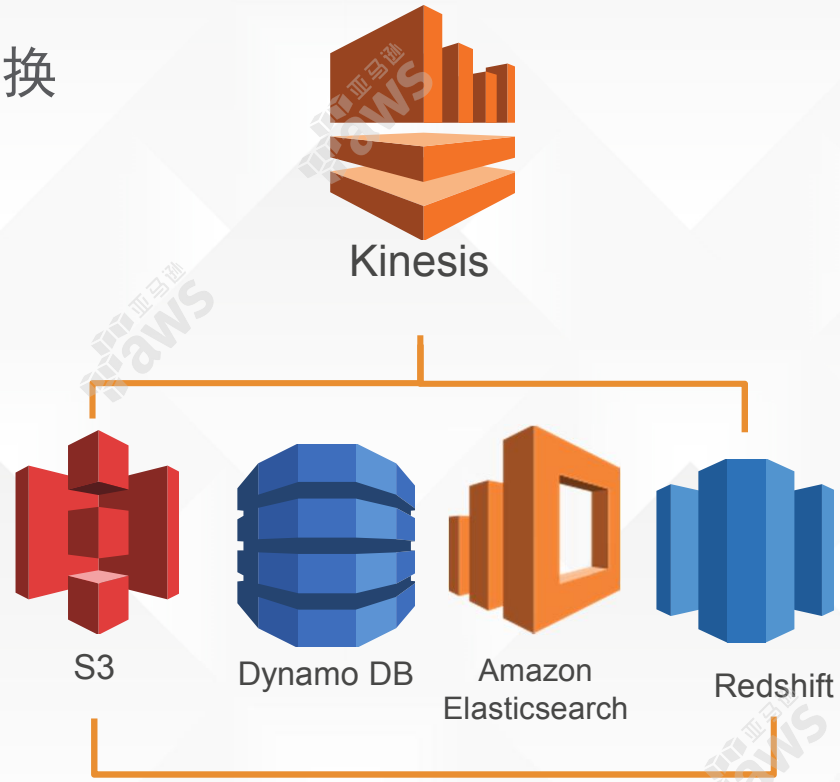
Amazon Kinesis 应用场景5: Apache Storm



http://blogs.aws.amazon.com/bigdata/post/Tx36LYSCY2R0A9B/Implement-a-Real-time-Sliding-Window-Application-Using-Amazon-Kinesis-and-Apache

Amazon Kinesis Connector Library

- 将数据按照用户需要的格式进行转换
- 将数据缓存下来，然后做批量处理
- 发送数据到对应的AWS服务
  - Amazon S3
  - Amazon Redshift
  - Amazon DynamoDB
  - Amazon Elasticsearch



Amazon Kinesis Agent

客户端代理程序使数据上传到Firehose变得更容易

- 监控文件的更新，发送更新的数据记录
- 内置文件循环，断点检查和故障重试等功能
- 以简单可靠并且及时的方式帮助用户传送数据
- 向AWS CloudWatch服务发送运行状况监控数据，以保证用户可以有效的监控数据流的工作状态和为故障定位提供了可靠的依据
  - 支持Amazon Linux AMI 2015.09或更新版本, or Red Hat Enterprise Linux 7或更新版本. 适用于部署在Linux环境上的Web服务器, front ends, 日志服务器等
- 也支持上传数据到Amazon Kinesis Streams

AdRoll Amazon Kinesis用户案例分享

AdRoll 是一家网络广告提供商，他在多个平台，多种终端类型下帮助用户通过智能，灵活有效的市场活动来提升品牌形像

AdRoll 需要处理的数据

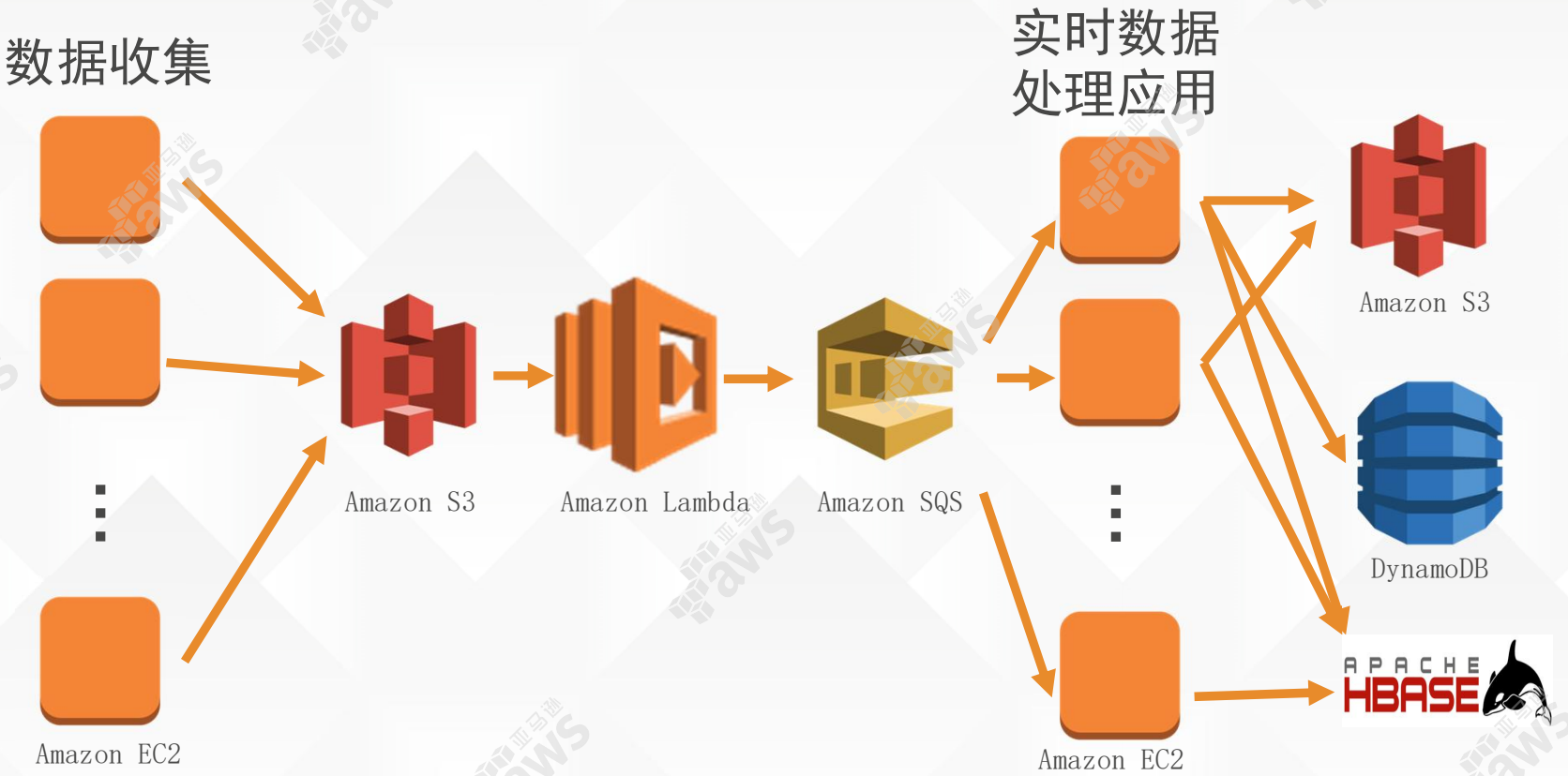
需要处理的数据类型：

- 实时数据
  - 广告主网站的用户点击数据
  - 投放的广告Serving ads
    - 广告展示 / 点击次数 / 转化率
    - 实时广告竞标数据(RTB)
- 与广告业务系统性能有关的回头率数据

数据的规模：

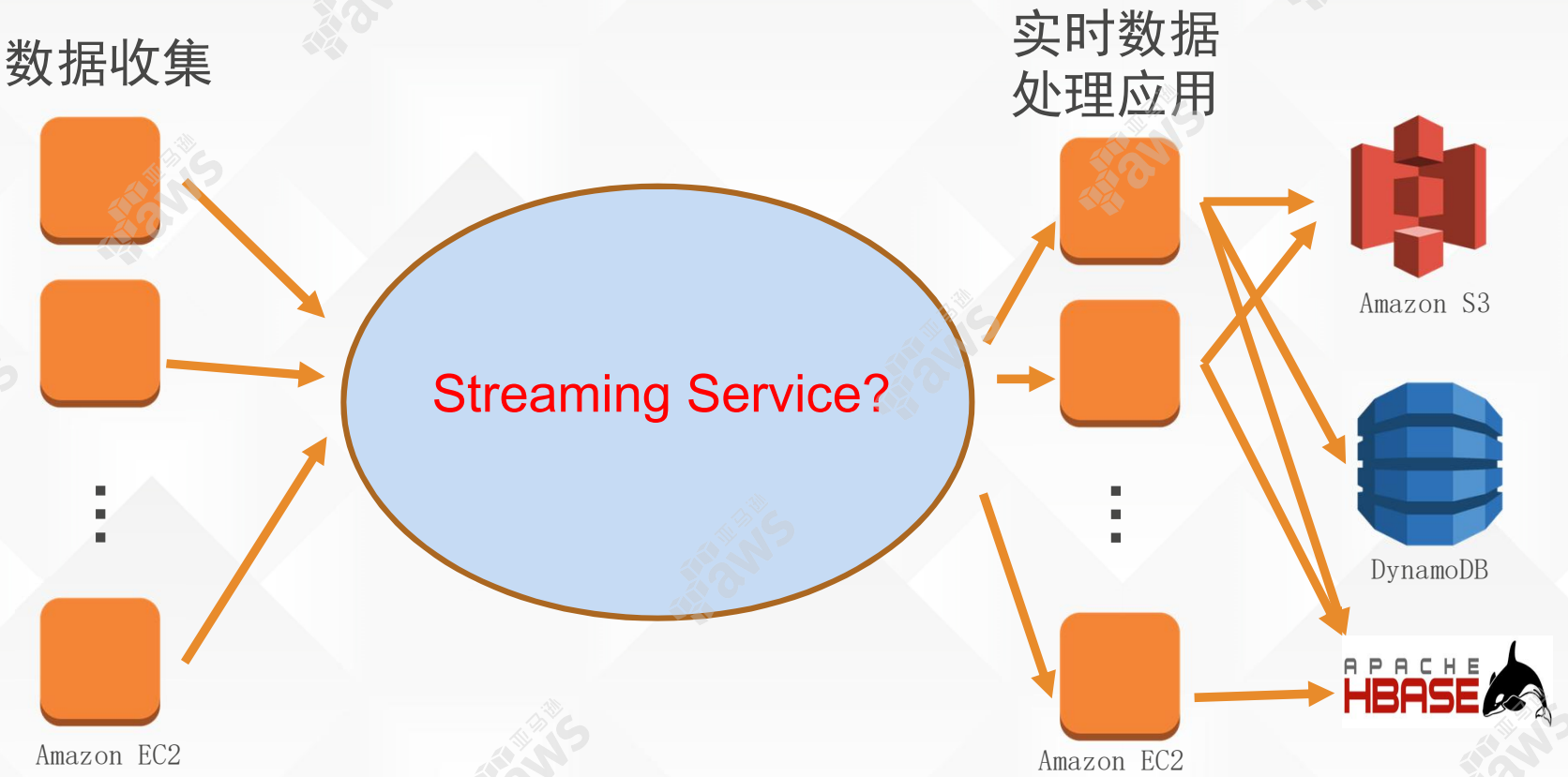
- 每天的广告展示数据: 2亿次事件 / 240GB
- 每天的用户点击数据: 16亿次事件 / 700GB
- 每天的实时广告竞标数据: 600亿次事件 / 80TB

AdRoll 自建实时数据流处理系统



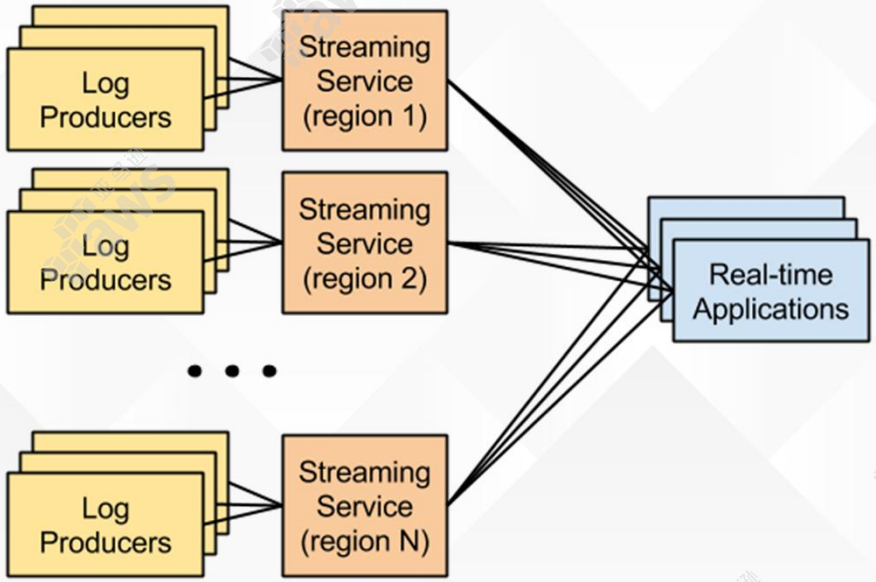
AdRoll 自建系统性能表现

AdRoll 寻找更优的解决方案

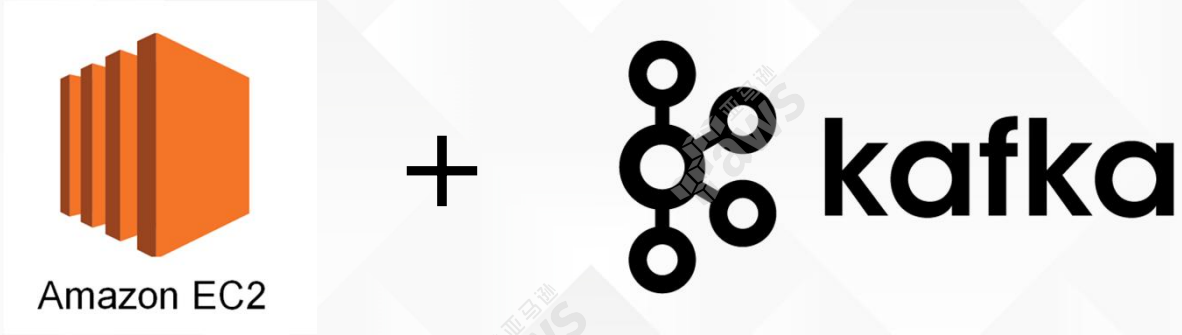


关于实时数据流处理系统需求

- 低延时
- 可以水平扩展
- 数据持久性
- 高可用
- 运营维护成本低
- 快速全球部署能力



AdRoll 首先尝试使用Kafka



AdRoll Kafka的使用体验

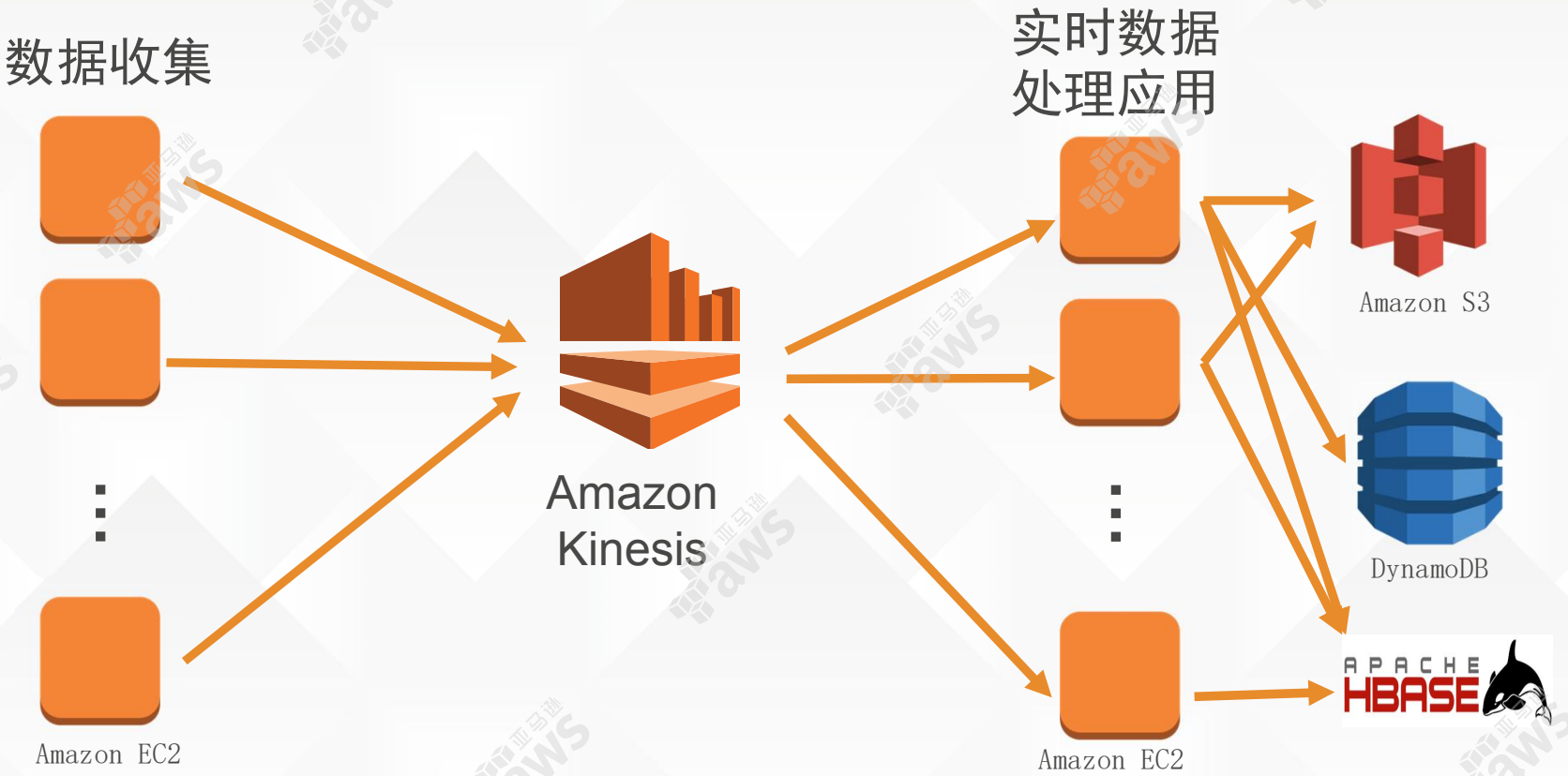
- 在使用过程中，在下面的场景遇到困难

  a) 无法准确评估应该为每个主题分配多少的磁盘空间
  b) 跨数据中心的数据镜像复制/同步

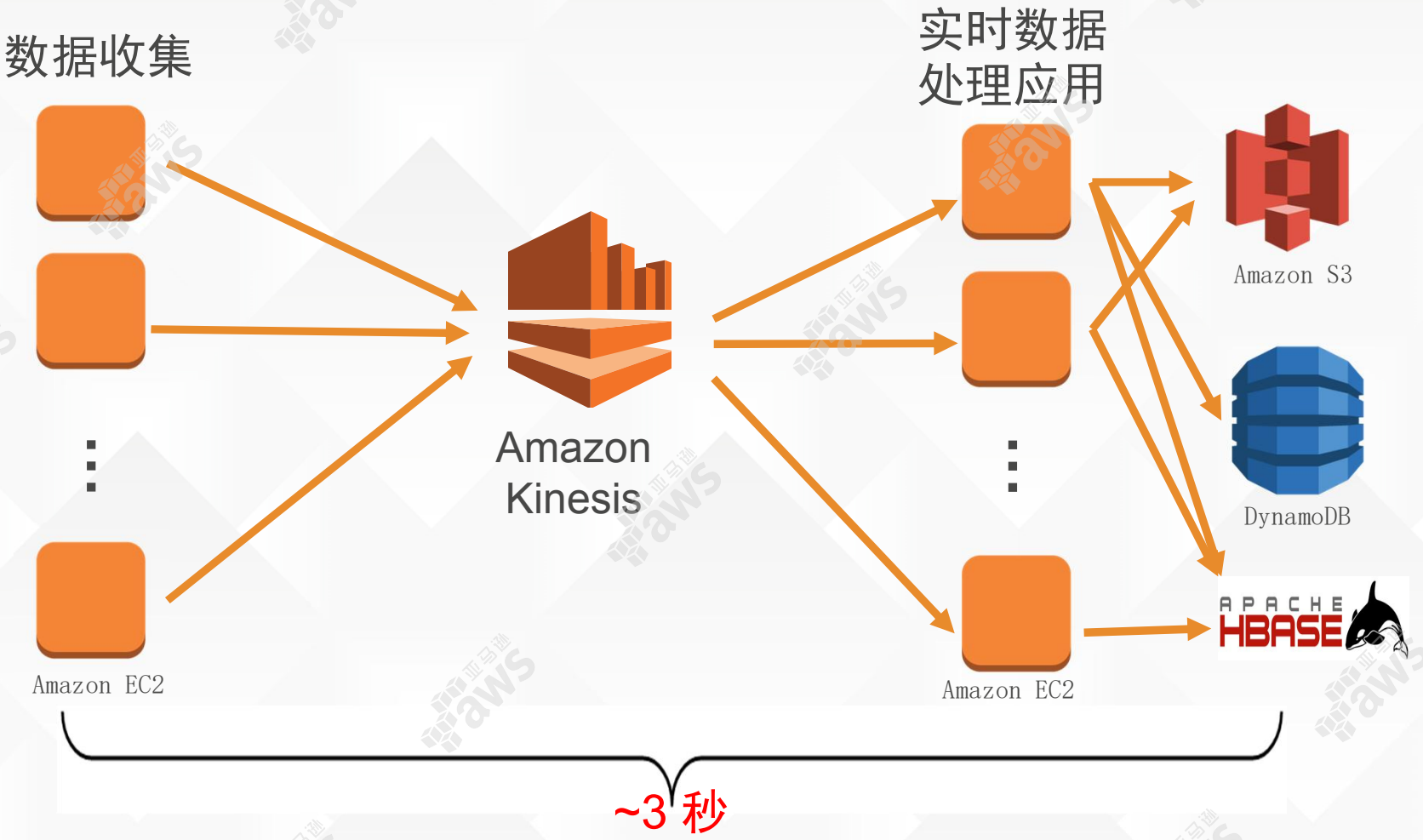
- 需要单独的人力资源来管理和维护Kafka平台，增加了成本

AdRoll 最终解决方案

AdRoll Kinesis拓扑结构



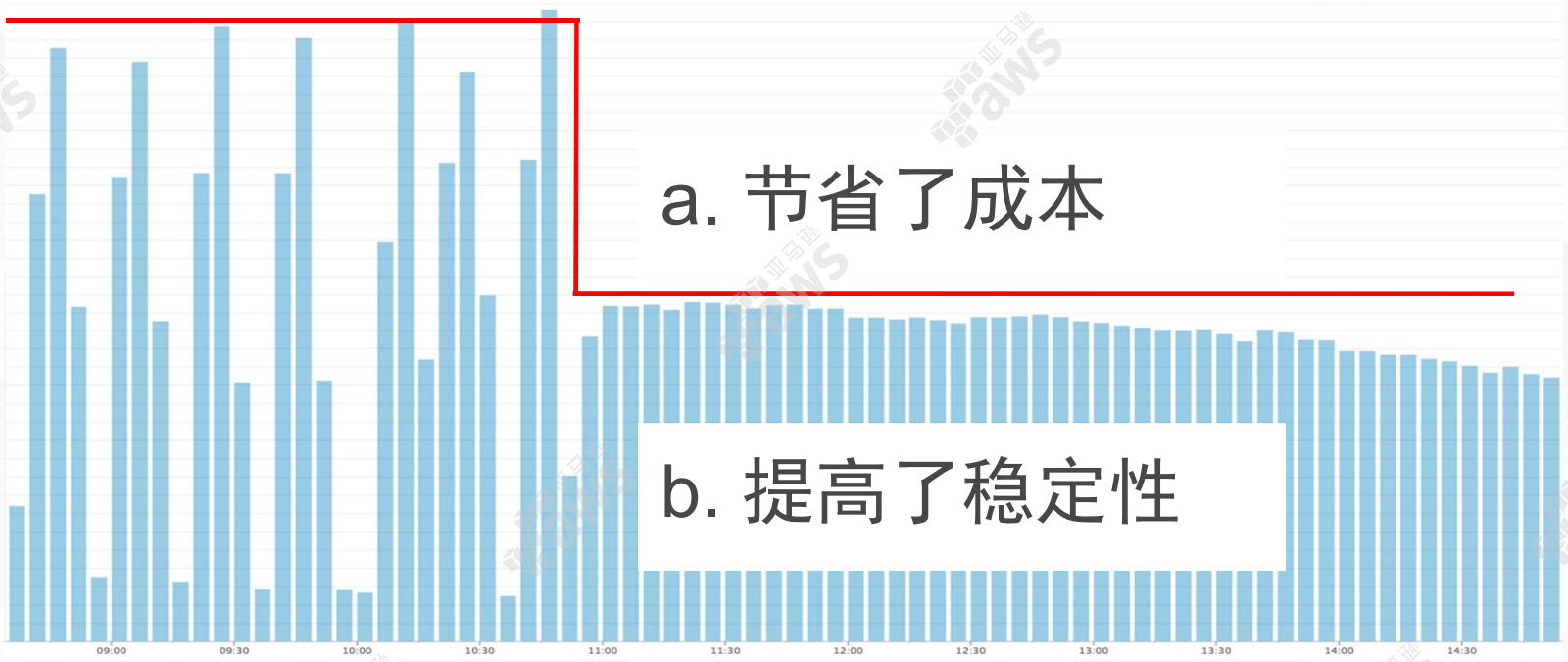
AdRoll Kinesis性能表现



AdRoll Kinesis带来的价值

① 延时从15分钟降到了3秒

② 与此同时节省了成本，提高了稳定性



AdRoll Kinesis服务使用情况

a) 5个区域

b) 155条Kinesis stream

c) 264个Shards

d) 25亿事件 (1个亿的Kinesis记录) / 每天

演示

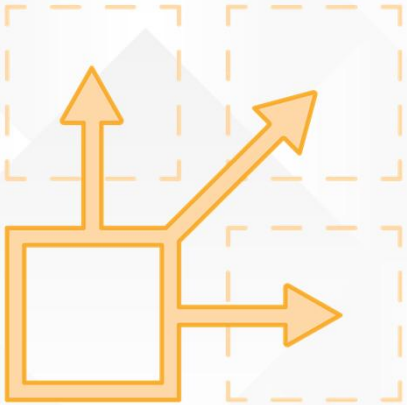
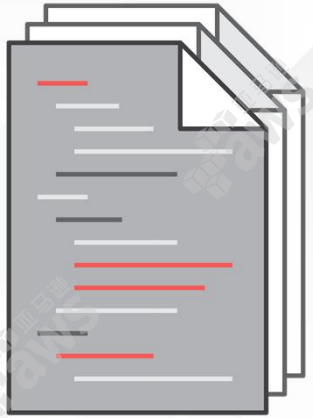
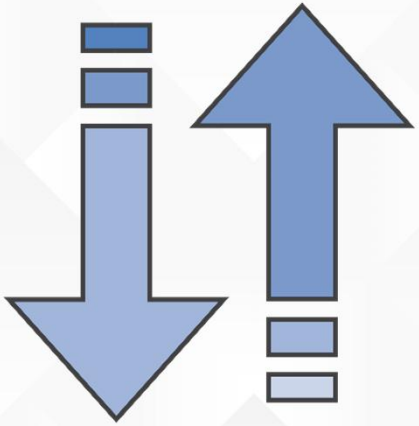
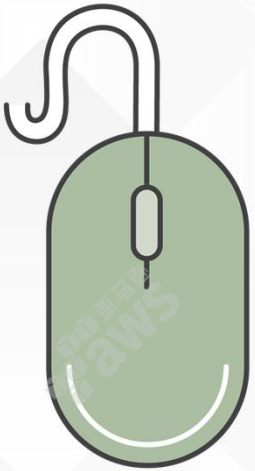
使用Amazon Kinesis将Web访问数据可视化

演示拓扑结构


Producer
Amazon Kinesis
Kinesis-enabled Application
Amazon DynamoDB
Dashboard
The same Amazon EC2

CloudFormation

- 模板服务 (EC2, RDS, S3, Security Group, Route Tables, etc.)
- JSON脚本文件,可以像软件一样管理基础架构
- 快速部署，演示，测试，以及复制同样的环境到其他区域
- 可以设置资源之间的依赖关系，可以执行更新

演示应用程序

```
        "files" : {
          "/var/kinesis-data-vis-sample-app/watchdog.sh" : {
            "content" : {"Fn::Join" : ["", [
              "#!/bin/bash\n",
              "if ! ps aux | grep HttpReferrerCounterApplication | grep -v grep ; then\n",
              "    # Launch the Kinesis application for counting HTTP referrer pairs\n",
              "    java -cp /var/kinesis-data-vis-sample-app/lib/\\* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerCounterApplication ", { "Ref" : "KCLDynamoDBTable" }, " ", { "Ref" : "KinesisStream" }, " ", { "Ref" : "CountsDynamoDBTable" }, " ",
"Ref" : "AWS::Region" }, " &>> /home/ec2-user/kinesis-data-vis-sample-app-kcl.log &\n",
              "fi\n",
              "if ! ps aux | grep HttpReferrerStreamWriter | grep -v grep ; then\n",
              "    # Launch our Kinesis stream writer to fill our stream with generated HTTP (resource, referrer) pairs.\n",
              "    # This will create a writer with 5 threads to send records indefinitely.\n",
              "    java -cp /var/kinesis-data-vis-sample-app/lib/\\* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerStreamWriter 5 ", { "Ref" : "KinesisStream" }, " ", { "Ref" : "AWS::Region" }, " &>> /home/ec2-user/kinesis-data-vis-sample-
app-publisher.log &\n",
              "fi\n",
              "if ! ps aux | grep WebServer | grep -v grep ; then\n",
              "    # Launch the webserver\n",
              "    java -cp /var/kinesis-data-vis-sample-app/lib/\\* com.amazonaws.services.kinesis.samples.datavis.WebServer 80 /var/kinesis-data-vis-sample-app/wwwroot ", { "Ref" : "CountsDynamoDBTable" }, " ", { "Ref" : "AWS::Region" }, " &>> /home/ec2-
user/kinesis-data-vis-sample-app-www.log &\n",
              "fi\n"
            ]]},
            "mode" : "000755",
            "owner" : "ec2-user",
            "group" : "ec2-user"
          },
```

1. 计数程序，　参数：a. KCLDynamoDBTable，　b. Kinesis stream，　c. DynamoDB计数表

2. 原始数据输入程序，　主要参数：　a. Kinesis stream

3. Web服务器，用于展示数据，主要参数：　DynamoDB计数表

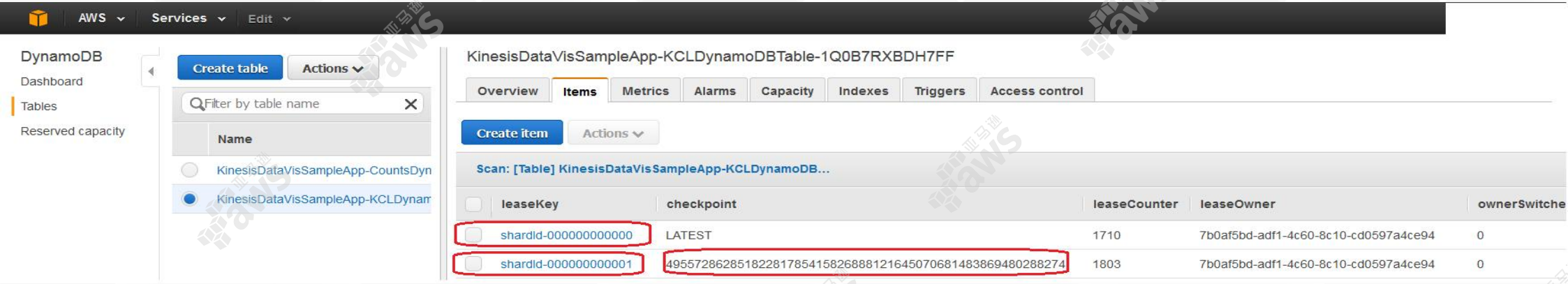
```
root      1723  0.0  0.2  88868  2724 ?        Ss   02:06   0:00 sendmail: accepting connections
smmsp     1731  0.0  0.1  80328  2040 ?        Ss   02:06   0:00 sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
root      1739  0.0  0.1 119504  1252 ?        Ss   02:06   0:00 crond
root      1749  0.0  0.0  17056   372 ?        Ss   02:06   0:00 /usr/sbin/atd
root      1777  0.9  8.5 1443036 87560 ?       Sl   02:07   0:25 java -cp /var/kinesis-data-vis-sample-app/lib/* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerCounterApplication KinesisDataVisSampleApp-KCLDynamoDBTable-1Q0B7
root      1781  3.3  9.2 1439952 93968 ?       Sl   02:07   1:33 java -cp /var/kinesis-data-vis-sample-app/lib/* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerStreamWriter 5 KinesisDataVisSampleApp-KinesisStream-1CNQZR9AY0I3
root      1787  0.6  7.9 1443036 80852 ?       Sl   02:07   0:17 java -cp /var/kinesis-data-vis-sample-app/lib/* com.amazonaws.services.kinesis.samples.datavis.WebServer 80 /var/kinesis-data-vis-sample-app/wwwroot KinesisDataVisSampleApp
```

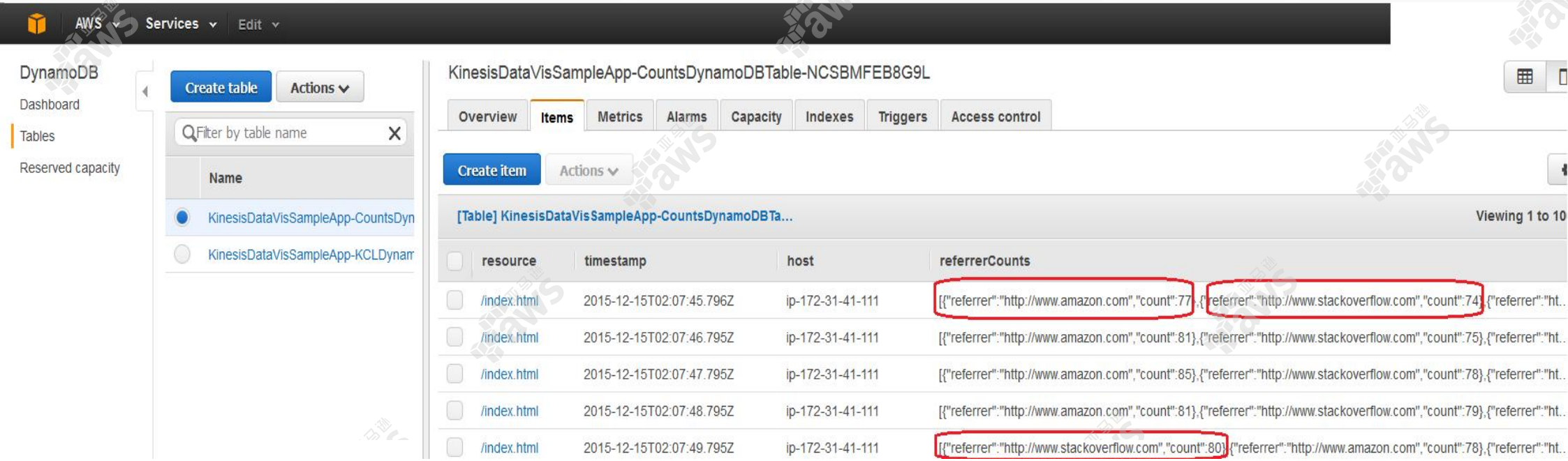
2. 原始数据写入Kinesis Stream

1. 计数程序，写入DynamoDB中

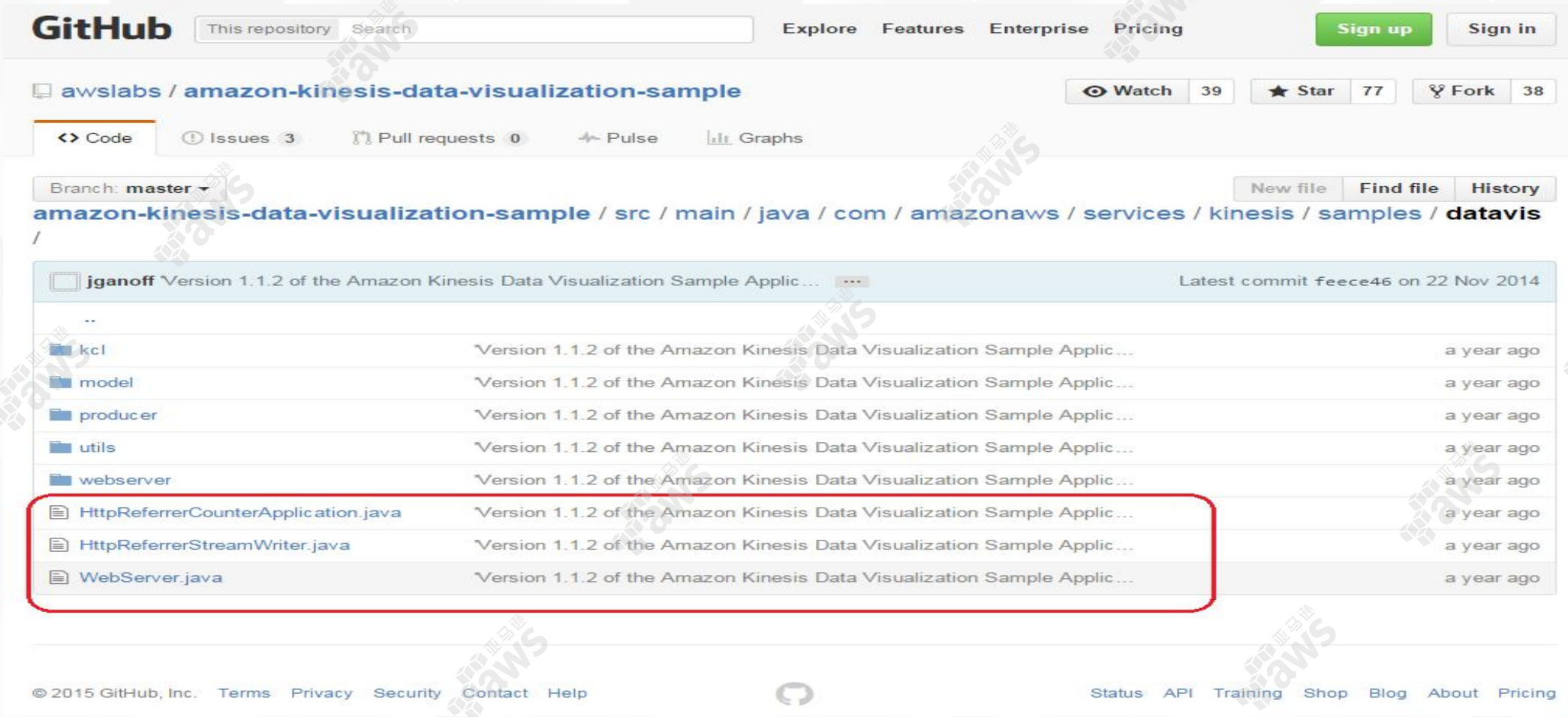
3. WebServer负责从DynamoDB中读取数据，然后实时显示出来

DynamoDB表


AWS
Services
Edit
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Actions
Filter by table name
Name
KinesisDataVisSampleApp-CountsDyn
KinesisDataVisSampleApp-KCLDynam
KinesisDataVisSampleApp-KCLDynamoDBTable-1Q0B7RXBDH7FF
Overview
Items
Metrics
Alarms
Capacity
Indexes
Triggers
Access control
Create item
Actions
Scan: [Table] KinesisDataVisSampleApp-KCLDynamoDB...
leaseKey
checkpoint
leaseCounter
leaseOwner
ownerSwitche
shardId-000000000000
LATEST
1710
7b0af5bd-adf1-4c60-8c10-cd0597a4ce94
0
shardId-000000000001
49557286285182281785415826888121645070681483869480288274
1803
7b0af5bd-adf1-4c60-8c10-cd0597a4ce94
0


AWS
Services
Edit
DynamoDB
Dashboard
Tables
Reserved capacity
Create table
Actions
Filter by table name
Name
KinesisDataVisSampleApp-CountsDyn
KinesisDataVisSampleApp-KCLDynam
KinesisDataVisSampleApp-CountsDynamoDBTable-NCSBMFEB8G9L
Overview
Items
Metrics
Alarms
Capacity
Indexes
Triggers
Access control
Create item
Actions
[Table] KinesisDataVisSampleApp-CountsDynamoDBTa...
Viewing 1 to 10
resource
timestamp
host
referrerCounts
/index.html
2015-12-15T02:07:45.796Z
ip-172-31-41-111
[{"referrer":"http://www.amazon.com","count":77},{"referrer":"http://www.stackoverflow.com","count":74},{"referrer":"ht..
/index.html
2015-12-15T02:07:46.795Z
ip-172-31-41-111
[{"referrer":"http://www.amazon.com","count":81},{"referrer":"http://www.stackoverflow.com","count":75},{"referrer":"ht..
/index.html
2015-12-15T02:07:47.795Z
ip-172-31-41-111
[{"referrer":"http://www.amazon.com","count":85},{"referrer":"http://www.stackoverflow.com","count":78},{"referrer":"ht..
/index.html
2015-12-15T02:07:48.795Z
ip-172-31-41-111
[{"referrer":"http://www.amazon.com","count":81},{"referrer":"http://www.stackoverflow.com","count":79},{"referrer":"ht..
/index.html
2015-12-15T02:07:49.795Z
ip-172-31-41-111
[{"referrer":"http://www.stackoverflow.com","count":80},{"referrer":"http://www.amazon.com","count":78},{"referrer":"ht..

代码整体解析


GitHub
This repository
Search
Explore
Features
Enterprise
Pricing
Sign up
Sign in
awslabs / amazon-kinesis-data-visualization-sample
Watch 39
Star 77
Fork 38
Code
Issues 3
Pull requests 0
Pulse
Graphs
Branch: master
New file
Find file
History
amazon-kinesis-data-visualization-sample / src / main / java / com / amazonaws / services / kinesis / samples / datavis /
jganoff Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic...
Latest commit feece46 on 22 Nov 2014
..
kcl Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
model Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
producer Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
utils Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
webserver Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
HttpReferrerCounterApplication.java Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
HttpReferrerStreamWriter.java Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
WebServer.java Version 1.1.2 of the Amazon Kinesis Data Visualization Sample Applic... a year ago
© 2015 GitHub, Inc. Terms Privacy Security Contact Help
Status API Training Shop Blog About Pricing

代码分析: 1. Stream Writer

zon-kinesis-data-visualization-sample/blob/master/src/main/java/com/amazonaws/services/kinesis/samples/datavis/HttpReferrerStreamWriter

```
72
73      AWSCredentialsProvider credentialsProvider = new DefaultAWSCredentialsProviderChain();
74      ClientConfiguration clientConfig = SampleUtils.configureUserAgentForSample(new ClientConfiguration());
75      AmazonKinesis kinesis = new AmazonKinesisClient(credentialsProvider, clientConfig);
76      kinesis.setRegion(region);
77
78      // The more resources we declare the higher write IOPS we need on our DynamoDB table.
79      // We write a record for each resource every interval.
80      // If interval = 500ms, resource count = 7 we need: (1000/500 * 7) = 14 write IOPS minimum.
81      List<String> resources = new ArrayList<>();
82      resources.add("/index.html");
83
84      // These are the possible referrers to use when generating pairs
85      List<String> referrers = new ArrayList<>();
86      referrers.add("http://www.amazon.com");
87      referrers.add("http://www.google.com");
88      referrers.add("http://www.yahoo.com");
89      referrers.add("http://www.bing.com");
90      referrers.add("http://www.stackoverflow.com");
91      referrers.add("http://www.reddit.com");
92
93      HttpReferrerPairFactory pairFactory = new HttpReferrerPairFactory(resources, referrers);
94
95      // Creates a stream to write to with 2 shards if it doesn't exist
96      StreamUtils streamUtils = new StreamUtils(kinesis);
97      streamUtils.createStreamIfNotExists(streamName, 2);
98      LOG.info(String.format("%s stream is ready for use", streamName));
99
100     final HttpReferrerKinesisPutter putter = new HttpReferrerKinesisPutter(pairFactory, kinesis, streamName);
101
102     ExecutorService es = Executors.newCachedThreadPool();
103
104     Runnable pairSender = new Runnable() {
105         @Override
106         public void run() {
107             try {
108                 putter.sendPairsIndefinitely(DELAY_BETWEEN_RECORDS_IN_MILLIS, TimeUnit.MILLISECONDS);
109             } catch (Exception ex) {
110                 LOG.warn("Thread encountered an error while sending records. Records will no longer be put by this thread.",
111                         ex);
112             }
113         }
114     };
115
116     for (int i = 0; i < numberOfThreads; i++) {
117         es.submit(pairSender);
118     }
```

1. 访问的Resources

2. 访问的来源referrers，程序中会随机选择一个

3. 初始化一个发送数据到Kinesis的对象

4. 指定线程函数，向Kinesis发送记录

5. 设置几个线程同时执行向Kinesis发送数据

代码分析: 2. Referrer Counter

on-kinesis-data-visualization-sample/blob/master/src/main/java/com/amazonaws/services/kinesis/samples/datavis/HttpReferrerCounterAp

```
        }

        String applicationName = args[0];
        String streamName = args[1];
        String countsTableName = args[2];
        Region region = SampleUtils.parseRegion(args[3]);

        AWSCredentialsProvider credentialsProvider = new DefaultAWSCredentialsProviderChain();
        ClientConfiguration clientConfig = SampleUtils.configureUserAgentForSample(new ClientConfiguration());
        AmazonKinesis kinesis = new AmazonKinesisClient(credentialsProvider, clientConfig);
        kinesis.setRegion(region);
        AmazonDynamoDB dynamoDB = new AmazonDynamoDBClient(credentialsProvider, clientConfig);
        dynamoDB.setRegion(region);

        // Creates a stream to write to, if it doesn't exist
        StreamUtils streamUtils = new StreamUtils(kinesis);
        streamUtils.createStreamIfNotExists(streamName, 2);
        LOG.info(String.format("%s stream is ready for use", streamName));

        DynamoDBUtils dynamoDBUtils = new DynamoDBUtils(dynamoDB);
        dynamoDBUtils.createCountTableIfNotExists(countsTableName);
        LOG.info(String.format("%s DynamoDB table is ready for use", countsTableName));
        String workerId = String.valueOf(UUID.randomUUID());
        LOG.info(String.format("Using working id: %s", workerId));
        KinesisClientLibConfiguration kclConfig =
                new KinesisClientLibConfiguration(applicationName, streamName, credentialsProvider, workerId);
        kclConfig.withCommonClientConfig(clientConfig);
        kclConfig.withRegionName(region.getName());
        kclConfig.withInitialPositionInStream(InitialPositionInStream.LATEST);

        // Persist counts to DynamoDB
        DynamoDBCountPersister persister =
                new DynamoDBCountPersister(dynamoDBUtils.createMapperForTable(countsTableName));

        IRecordProcessorFactory recordProcessor =
                new CountingRecordProcessorFactory<HttpReferrerPair>(HttpReferrerPair.class,
                        persister,
                        COMPUTE_RANGE_FOR_COUNTS_IN_MILLIS,
                        COMPUTE_INTERVAL_IN_MILLIS);

        Worker worker = new Worker(recordProcessor, kclConfig);

        int exitCode = 0;
        try {
            worker.run();
        } catch (Throwable t) {
```

1. 使用KCL创建worker来处理数据流

2. 将结果写入到DynamoDB表

3. 指定记录处理流程

4. 让worker持续的运行记录处理流程

代码分析: 3. WebServer

```
41          *
42          * @param args Expecting 4 arguments: Port number, File path to static content, the name of the
43          *         DynamoDB table where counts are persisted to, and the AWS region in which these resources
44          *         exist or should be created.
45          * @throws Exception Error starting the web server.
46          */
47         public static void main(String[] args) throws Exception {
48             if (args.length != 4) {
49                 System.err.println("Usage: " + WebServer.class
50                         + " <port number> <directory for static content> <DynamoDB table name> <region>");
51                 System.exit(1);
52             }
53             Server server = new Server(Integer.parseInt(args[0]));
54             String wwwroot = args[1];
55             String countsTableName = args[2];
56             Region region = SampleUtils.parseRegion(args[3]);
57
58             // Servlet context
59             ServletContextHandler context =
60                     new ServletContextHandler(ServletContextHandler.NO_SESSIONS | ServletContextHandler.NO_SECURITY);
61             context.setContextPath("/api");
62
63             // Static resource context
64             ResourceHandler resources = new ResourceHandler();
65             resources.setDirectoriesListed(false);
66             resources.setWelcomeFiles(new String[] { "graph.html" });
67             resources.setResourceBase(wwwroot);
68
69             // Create the servlet to handle /GetCounts
70             AWSCredentialsProvider credentialsProvider = new DefaultAWSCredentialsProviderChain();
71             ClientConfiguration clientConfig = SampleUtils.configureUserAgentForSample(new ClientConfiguration());
72             AmazonDynamoDB dynamoDB = new AmazonDynamoDBClient(credentialsProvider, clientConfig);
73             dynamoDB.setRegion(region);
74             DynamoDBUtils dynamoDBUtils = new DynamoDBUtils(dynamoDB);
75             context.addServlet(new ServletHolder(new GetCountsServlet(dynamoDBUtils.createMapperForTable(countsTableName))),
76                     "/GetCounts/*");
77
78             HandlerList handlers = new HandlerList();
79             handlers.addHandler(context);
80             handlers.addHandler(resources);
81             handlers.addHandler(new DefaultHandler());
82
83             server.setHandler(handlers);
84             server.start();
85             server.join();
86         }
87     }
```

1. 启动一个HTTP服务器, 使它监听在指定的端口

2. 设置执行的动作是从DynamoDB表中读取计数

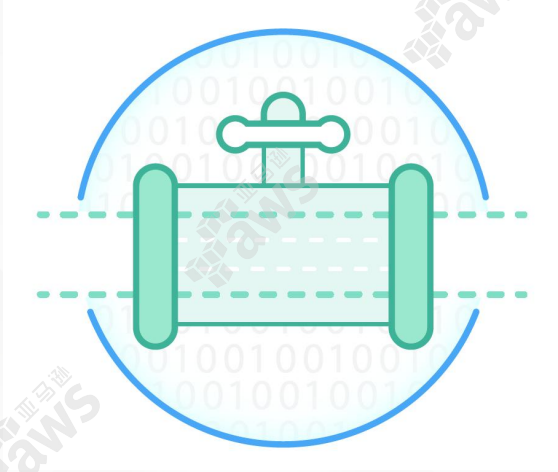
3. 启动这个服务器，持续的执行查询计数和显示结果的动作

Amazon Kinesis

轻松实现实时数据流处理

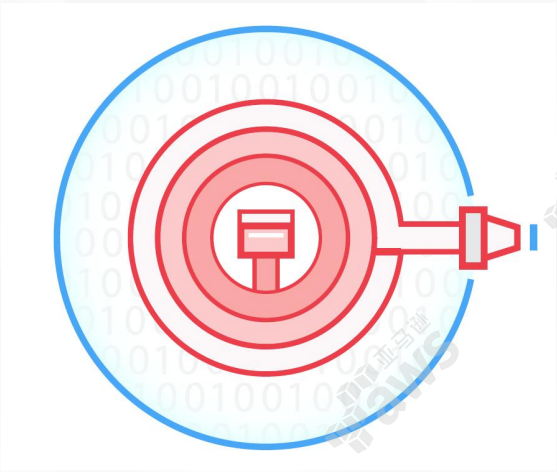
Amazon Kinesis:

完善的功能体系可以帮助您在AWS平台上轻松实现数据流的捕获，传送和分析处理



- **Amazon Kinesis Streams**

使用Kinesis Streams可以使用户轻松创建自己的应用来处理和分析数据流



- **Amazon Kinesis Firehose**

轻松实现装载数据到静态对象存储S3 和数据仓库Redshift

- **Amazon Kinesis Analytics**

只需要使用您熟知SQL语句，即可实现数据流上的数据查询

Amazon Kinesis 带给用户的价值

简单易用

只需要创建一个Stream，设置好期望的数据规模就可以开始进行实时数据收集和处理了，免除了维护等与用户业务无关的无差异的工作

低延时

相对于批处理任务带来的分钟级甚至小时级的延时来说，Kinesis使用户在处理实时数据仅需秒级时延成为可能

吞吐量弹性扩展

Kinesis支持弹性扩展，可以根据数据规模调整数据流容量，满足了用户的业务需求

与其他服务完美集成

Kinesis可以可靠的帮助用户实时收集，处理和转换数据，并可以根据业务需要将数据选择性地存储到S3, Redshift或是DynamoDB

轻松构建实时应用

Kinesis客户端程序可以帮助开发人员轻松构建实时数据处理应用程序

低成本

弹性扩展可以满足不同数据规模的用户需求，按需付费的模式使成本最优化

问题

① 哪个Kinesis的功能可以帮助用户将实时数据流写入S3或是Redshift

② 一个Kinesis Streams中的Shard可以支持多大的数据读取速度

③ Kinesis Client Library和Kinesis Producer Library都提供了哪些好处

Thank You