PERFORMANCE TESTING APPROACH TO AWS KINESIS STREAM AND LOADRUNNER

Author

Devdutta Dasgupta

Abstract

In the wake of an increasingly digital economy, businesses are racing to build operational knowledge around the vast sums of data they produce each day. And with data now at the center of almost every business function, developing practices for working with data is critical regardless of your company's size or industry. AWS Kinesis Stream provides a platform to consume these data from various sources and process them for analytical purposes. In the current age of Digital Marketing the important aspect which comes to play is the speed at which you can process the data.

The AWS Cloud provides a broad set of infrastructure services, such as computing power, storage options, networking and databases that are delivered as a utility: on-demand, available in seconds, with pay-as-you-go pricing. Amazon Kinesis is a platform for streaming data on AWS, offering powerful services to make it easy to load and analyze streaming data, and also providing the ability for to build custom streaming data applications for specialized needs.

To ensure the timely execution as well to determine the proper infrastructure to host the Kinesis Stream it is very important to execute a through performance test of the Kinesis Stream application. The Data Message flow to kinesis Stream gets authenticated using AES256 encryption which most of performance testing tools (e.g. Loadrunner) doesn't supports. This paper presents a succinct approach to implement the performance testing steps overcoming the authentication issue and execute Load Test on Kinesis Stream. The approach explains how to overcome the authentication issue with the help of Node.js and integrate it with Loadrunner and execute the performance test. The paper also provides details on the critical metrics to be monitored from CloudWatch for analyzing the Kinesis Stream performance. AWS provides its own monitoring through service called CloudWatch. This document also provides insight on the monitoring steps using CloudWatch during the performance testing.

In Summary this paper provides an approach for testing Kinesis Stream with help of Node.js. The paper also provides benefits on the approach and how it can be utilized for other tools and testing approaches.

Introduction

What is Kinesis?

Amazon Kinesis is a platform for streaming data on AWS, offering powerful services to make it easy to load and analyze streaming data, and also providing the ability for you to build custom streaming data applications for specialized needs. Web applications, mobile devices, wearables, industrial sensors, and many software applications and services can generate staggering amounts of streaming data – sometimes Terabytes per hour – that need to be collected, stored, and processed continuously. Amazon Kinesis services enable you to do that simply and at a low cost.

AWS Kinesis Services

There are three different AWS Kinesis services offered by AWS. They are Amazon Kinesis Firehose, Amazon Kinesis Analytics and Amazon Kinesis Streams.

Amazon Kinesis Streams

Amazon Kinesis Streams enables one to build custom applications that process or analyze streaming data for specialized needs. Amazon Kinesis Streams can continuously capture and store terabytes of data per hour from hundreds of thousands of sources such as website clickstreams, financial transactions, social media feeds, IT logs, and location-tracking events.

Message Flow process in Kinesis Stream

The real time analytics data from different external sources and click stream media are required to be used for the analytic purpose to improve the business insight into the consumer buying capacities and choices.

The AWS kinesis stream was used to feed in and process these real time data and update the PostGres Database. The data from these external sources which are entering into the Kinesis Stream are secured on the run via AES 256 Encryption. This is a common norm for all AWS services which practices AES 256 bit encryption to encrypt all data on the run.

Performance Testing Requirements

For a Travel and Hospitality account the ask was to analyze the pattern of search and booking made in their line of businesses like eCommerce website, Call Center Application, Third Party Vendor and Connectivity partner. These data were required to be stored and analyze for future business decisions.

The performance test requirement was to test the performance of the Kinesis Stream to process the data and upload the same to the PostGres Database. The performance test requirement was to measure the end to end response time for processing the request and test the scalability of the Cloud Infrastructure with increasing data volume.

Since the external sources were unavailable in test environment the messages which were XML inputs directly into Kinesis stream, have to be replicated as a testing stub. As mentioned earlier AWS required AES256 authentication for allowing any messages to be posted in the Kinesis stream.

The performance testing tool for the test scenario was LoadRunner 11.5. The limitation of Loadrunner doesn't allow any such authentication which would allow posting messages in the Kinesis stream. The problem for continuing the performance testing was to devise a way to post messages to the AWS Kinesis Stream using AWS accepted authentication mechanism.

Performance Testing Approach

Performance Test Script Development

Each user who has access to AWS gets a unique access key and secret access keys. These are the login credentials used to generate signature keys which provides valid authentication to interact with the AWS infrastructure. Messages posted with valid signature are accepted by AWS Kinesis stream.

AWS provides its own development package (aws-sdk) which creates signature using the valid user access key and secret access keys. The javascript developed using the package is executed with help of node.js to send messages to the kinesis stream.

Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

We need to install node.js in the Load Generator for developing the test scripts and to execute the performance test using the Load Generator

AWS-SDK

AWS SDK for JavaScript in Node.js helps take the complexity out of coding by providing JavaScript objects for AWS services including Amazon S3, Amazon EC2, DynamoDB, Amazon SWF and many other services. The single, downloadable package includes the AWS JavaScript Library and documentation.

Once node.js is installed we need to add the installation directory to the PATH system variable so that it can be called from any place.

We need to install aws sdk in the Load Generator machine to develop the test script and to execute the performance test using the Load Generator.

Installation steps for aws sdk

- Open Command prompt and navigate to script folder (say C:\Scripts)
- Execute the below query.

```
npm install aws-sdk
```

- Under C:\Scripts you a sdk package gets copied under the folder name “node_modules”.
- Within this folder you would get all required aws-sdk javascript objects.
- We need to copy the node_modules folder in our LR test script folder.

Once Node.js and aws sdk are installed and configured a javascript is developed which pushes the sample XML messages to the Kinesis stream. As a part of data setup for the performance testing a bulk volume of sample XML messages were created which would be pushed to the Kinesis stream.

```
'use strict';

var aws = require('aws-sdk');

aws.config.update({accessKeyId: '<your access key>', secretAccessKey: '<your secret access key>'});   //Provide Access key and Secret Access key of the Valid User

var kinesis = new aws.Kinesis({region : 'us-west-2'});

   var XMLPath = process.argv[2];          //The name of the XML file which need to be posted and is passed as second argument in our LR script

   var fs = require('fs');

   var fileData = fs.readFileSync(XMLPath, 'utf8');

   var recordParams = {

     Data : fileData,

     PartitionKey : 'test',

     //StreamName : "DataReplay"

     StreamName : "<Kinesis Stream Name>"  //Provide the Kinesis stream name where data need to be posted

   };

   kinesis.putRecord(recordParams, function(err, data) {

     if (err)

       console.log('Error encountered for data file:' + err + ': ' + XMLPath);

     else

       console.log('Successfully sent data file to Kinesis.' + ': ' + XMLPath);

}
```

The file is saved as DataUpload.js and is placed in Loadrunner test script folder.

Now the test script is developed with the following code:

```
web_js_run( “File=DataUpload.js”,
                SOURCES,
                “File={FilePath}”,
```

```
LAST);
```

The parameter file “FilePath” contains the list of all files along the path and the parameter settings is set as Unique each iteration, to ensure unique file is pushed to Kinesis stream every time.

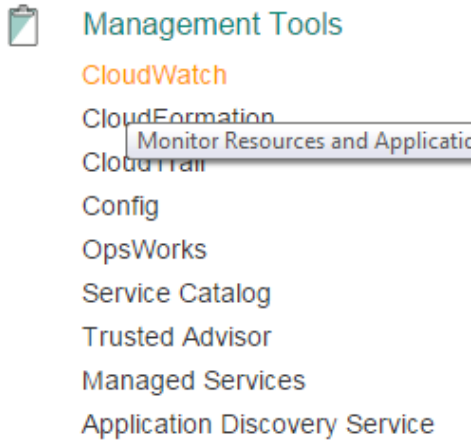
Performance Test Execution and Monitoring

Once the test script is developed the test scenario is developed replicating the production message volume processed per hour. The test is executed and the required volumes of messages are pushed to the Kinesis Stream. The total number of virtual users required to push the messages are calculated after initial smoke test and calculating the value of (t + R) using Little law.

The monitoring of the AWS Cloud Infrastructure is completely governed by CloudWatch.

AWS CloudWatch

Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources. You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health.



The CloudWatch provides following metrics:

- Basic Stream level Metrics
  - GetRecords.Bytes
  - GetRecords.IteratorAge
  - GetRecords.IteratorAgeMilliseconds
  - GetRecords.Latency
  - GetRecords.Records
  - GetRecords.Success
  - IncomingBytes
  - IncomingRecords
  - PutRecord.Bytes
  - PutRecord.Latency
  - PutRecord.Success
  - PutRecords.Bytes
  - PutRecords.Latency
  - PutRecords.Records
  - PutRecords.Success

    - ReadProvisionedThroughputExceeded
    - WriteProvisionedThroughputExceeded
- Enhanced Shard-level Metrics
    - IncomingBytes
    - IncomingRecords
    - IteratorAgeMilliseconds
    - OutgoingBytes
    - OutgoingRecords
    - ReadProvisionedThroughputExceeded
    - WriteProvisionedThroughputExceeded

The highlighted metrics are the AWS recommended Kinesis Streams Metrics.

Apart from the above mentioned metrics pertaining to Kinesis stream, other inter-linked applications (such as PostGresDB in our case) are also monitored using CloudWatch. CloudWatch provides us different metrics such as CPU, Memory, I/O etc. pertaining to RDS (Relational Database System) to be monitored.

CloudWatch also provides us the feature to create our own dashboard where we can define the metrics we need to monitor during our test. CloudWatch also provides option for collecting custom metrics generated by any code that need to be monitored.

CloudWatch also provides the logs for all services running in the AWS for which logging is enabled. This helps in identifying any errors that might have generated during the load test. This is helps us to debug any issue that might have excavated from the test. These features can be provisioned to capture required metrics during the performance test as well as help in deep diving into the application.

Common Issues identified and tuning recommended during the Load Test:

- Messages are getting throttled and as a result getting delayed in processing of the results.
- The PostGresDB CPU was running high and as a result data insertion was taking time.
- Optimum number of thread count was established for best processing the messages in parallel.

Benefits of the Solution

- Processing these data in a timely fashion holds the keys to success for any Business Organization. Performance Testing this data processing helps boosting Business confidence for ensuring timely processing of data and also to uncover any scalability issues in the application if any. It also helps to uncover any unforeseen issues which might cause issue in real production peak load.

- The performance testing of the AWS Kinesis Stream helped the business successfully analyze the search data used by the customer while using the Client e-commerce application. The implementation of the approached paved the path to gather the data logs across different line of business.

- The approach outlined here is based on utilizing the features provided by AWS SDK to interact with AWS services in a Node.js environment using javascript. This approach is extendible to other performance testing tools and with other automation tools which can execute Javascript.

Conclusion

To successfully conduct the performance testing for any interfaces hosted on AWS Cloud, we first need to establish a secure communication to AWS by authenticating the message or request. It is similar to the establishing a secure certificate for any SSL application. This authentication can be established by making use of aws-sdk packages running over Node.js platform.

AWS CloudWatch can be leveraged for monitoring all the servers and services hosted under AWS Cloud. For successful execution of performance testing AWS CloudWatch must be utilized to full extent.

The test data / files used in the process also determine the actual load applied on the system during the execution phase. They should be as close to the actual Production data / files as possible.

References

- https://aws.amazon.com/kinesis/?sc_channel=PS&sc_campaign=kinesis_2016&sc_publisher=google&sc_medium=awns_kinesis_b&sc_content=kinesis_e&sc_detail=aws%20kinesis&sc_category=kinesis&sc_segment=103255132002&sc_matchtype=e&sc_country=US&s_kwcid=AL!4422!3!103255132002!e!!g!!aws%20kinesis&ef_id=WEHJfwAAAFwLz68g:20170105201605:s
- https://aws.amazon.com/sdk-for-node-js/
- http://docs.aws.amazon.com/streams/latest/dev/monitoring-with-cloudwatch.html
- http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Monitoring.html#monitoring-cloudwatch