Lambda Architecture for Batch and Stream Processing

October 2018

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Lambda architecture is a data-processing design pattern to handle massive quantities of data and integrate batch and real-time processing within a single framework. (Lambda architecture is distinct from and should not be confused with the AWS Lambda compute service.)

This paper covers the building blocks of a unified architectural pattern that unifies stream (real-time) and batch processing. After reading this paper, you should have a good idea of how to set up and deploy the components of a typical Lambda architecture on AWS. This white paper is intended for Amazon Web Services (AWS) Partner Network (APN) members, IT infrastructure decision-makers, and administrators.

Introduction

When processing large amounts of semi-structured data, there is usually a delay between the point when data is collected and its availability in reports and dashboards. Often the delay results from the need to validate or at least identify granular data. In some cases, however, being able to react immediately to new data is more important than being 100 percent certain of the data's validity.

The AWS services frequently used to analyze large volumes of data are Amazon EMR and Amazon Athena. For ingesting and processing stream or real-time data, AWS services like Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, Spark Streaming and Spark SQL on top of an Amazon EMR cluster are widely used. Amazon Simple Storage Service (Amazon S3) forms the backbone of such architectures providing the persistent object storage layer for the AWS compute service.

Lambda architecture is an approach that mixes both batch and stream (real-time) data-processing and makes the combined data available for downstream analysis or viewing via a serving layer. It is divided into three layers: the batch layer, serving layer, and speed layer.

Figure 1 shows the batch layer (batch processing), serving layer (merged serving layer) and speed layer (stream processing).
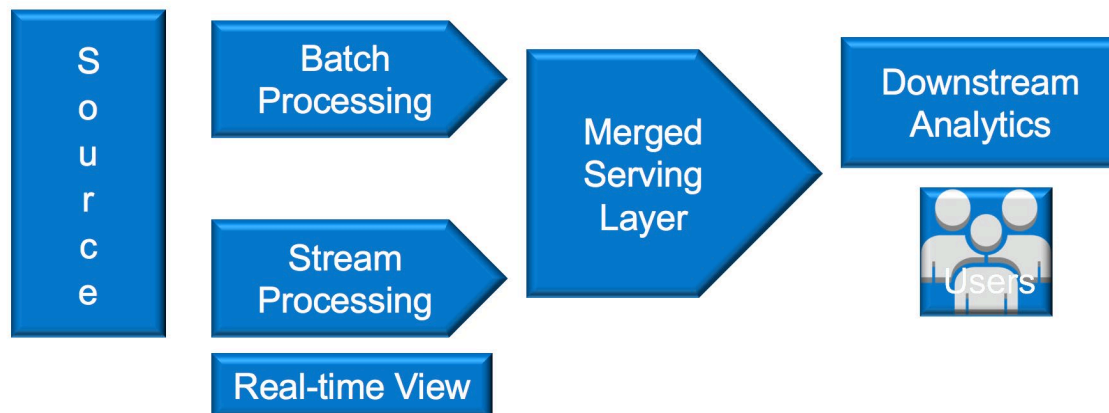

Figure 1: Lambda Architecture

In Figure 1, data is sent both to the batch layer and to the speed layer (stream processing). In the batch layer, new data is appended to the master data set. It consists of a set of records containing information that cannot be derived from the existing data. It is an immutable, append-only dataset. This process is analogous to extract, transform, and load (ETL) processing.

The results of the batch layer are called batch views and are stored in a persistent storage layer. The serving layer indexes the batch views produced by the batch layer. It is a scalable

data store that swaps in new batch views as they become available. Due to the latency of the batch layer, the results from the serving layer are out-of-date.

The speed layer compensates for the high latency of updates to the serving layer from the batch layer. The speed layer processes data that has not been processed in the last batch of the batch layer. This layer produces the real-time views that are always up-to-date.

The speed layer is responsible for creating real-time views that are continuously discarded as data makes its way through the batch and serving layers. Queries are resolved by merging the batch and real-time views. Re-computing data from scratch helps if the batch or real-time views become corrupted. This is because the main data set is append only and it is easy to restart and recover from the unstable state. The end user can always query the latest version of the data, which is available from the speed layer.

Overview

This section provides an overview of the various AWS services that form the building blocks for the batch, serving, and speed layers of lambda architecture.

Each of the layers in the Lambda architecture can be built using various analytics, streaming, and storage services available on the AWS platform.
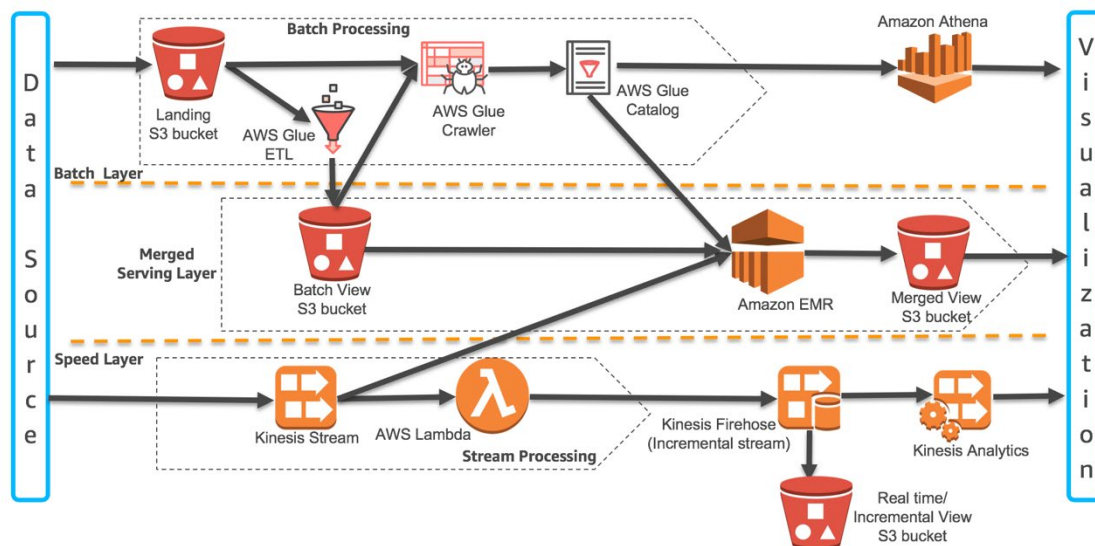

Figure 2: Lambda Architecture Building Blocks on AWS

The batch layer consists of the landing Amazon S3 bucket for storing all of the data (e.g., clickstream, server, device logs, and so on) that is dispatched from one or more data sources. The raw data in the landing bucket can be extracted and transformed into a batch view for analytics using AWS Glue, a fully managed ETL service on the AWS platform. Data analysis is performed using services like Amazon Athena, an interactive query service, or managed Hadoop framework using Amazon EMR. Using Amazon QuickSight, customers can also perform visualization and one-time analysis.

The speed layer can be built by using the following three options available with Amazon Kinesis:

- Kinesis Data Streams and Kinesis Client Library (KCL) – Data from the data source can be continuously captured and streamed in near real-time using Kinesis Data Streams. With the Kinesis Client Library (KCL), you can build your own application that can preprocess the streaming data as they arrive and emit the data for generating incremental views and downstream analysis.

- Kinesis Data Firehose – As data is ingested in real-time, customers can use Kinesis Data Firehose to easily batch and compress the data to generate incremental views. Kinesis Data Firehose also allows customer to execute their custom transformation logic using AWS Lambda before delivering the incremental view to Amazon S3.

- Kinesis Data Analytics – This service provides the easiest way to process the data that is streaming through Kinesis Data Stream or Kinesis Data Firehose using SQL. This enables customers to gain actionable insight in near real-time from the incremental stream before storing it in Amazon S3.

Finally, the serving layer can be implemented with Spark SQL on Amazon EMR to process the data in Amazon S3 bucket from the batch layer, and Spark Streaming on an Amazon EMR cluster, which consumes data directly from Amazon Kinesis streams to create a view of the entire dataset which can be aggregated, merged or joined. The merged data set can be written to Amazon S3 for further visualization. Both of these components are part of the same code base, which can be invoked as required, thus reducing the overhead of maintaining multiple code bases.

The metadata (e.g., table definition and schema) associated with the processed data is stored in the AWS Glue catalog to make the data in the batch view immediately available for queries by downstream analytics services in the batch layer. Customer can use a Hadoop based stream processing application for analytics, such as Spark Streaming on Amazon EMR.

Data Ingestion

The data ingestion step comprises data ingestion by both the speed and batch layer, usually in parallel. For the batch layer, historical data can be ingested at any desired interval. For the speed layer, the fast-moving data must be captured as it is produced and streamed for analysis. The data is immutable, time tagged or time ordered. Some examples of high velocity data include log collection, website clickstream logging, social media stream, and IoT device event data.

This fast data is captured and ingested as part of the speed layer using Amazon Kinesis Data Streams, which is the recommended service to ingest streaming data into AWS. Kinesis offers key capabilities to cost effectively process and durably store streaming data at any scale. Customers can use Amazon Kinesis Agent, a pre-built application, to collect and send data to

an Amazon Kinesis stream or use the Amazon Kinesis Producer Library (KPL) as part of a custom application.

For batch ingestions, customers can use AWS Glue or AWS Database Migration Service to read from source systems, such as RDBMS, Data Warehouses, and No SQL databases.

Data Transformation

Data transformation is a key step in the Lambda architecture where the data is manipulated to suit downstream analysis. The raw data ingested into the system in the previous step is usually not conducive to analytics as is. The transformation step involves data cleansing that includes de-duplication, incomplete data management, and attribute standardization. It also involves changing the data structures if necessary, usually into an OLAP model to facilitate easy querying of data.

Amazon Glue, Amazon EMR, and Amazon S3 form the set of services that allow users to transform their data. Kinesis analytics enables users to get a view into their data stream in real-time, which makes downstream integration to batch data easy. Let's dive deeper into data transformation and look at the various steps involved:

1. The data ingested via the batch mechanism is put into an S3 staging location. This data is a true copy of the source with little to no transformation.

2. The AWS Glue Data Catalog is updated with the metadata of the new files. The Glue Data Catalog can integrate with Amazon Athena, Amazon EMR and forms a central metadata repository for the data.

3. An AWS Glue job is used to transform the data and store it into a new S3 location for integration with real-time data. AWS Glue provides many canned transformations, but if you need to write your own transformation logic, AWS Glue also supports custom scripts.

4. Users can easily query data on Amazon S3 using Amazon Athena. This helps in making sure there are no unwanted data elements that get into the downstream bucket. Getting a view of source data upfront allows development of more targeted metrics. Designing analytical applications without a view of source data or getting a very late view into the source data could be risky. Since Amazon Athena uses a schema-on-read approach instead of a schema-on-write, it allows users to query data as is and eliminates the risk.

5. Amazon Athena integrates with Amazon QuickSight, which allows users to build reports and dashboards on the data.

6. For the real-time ingestions, the data transformation is applied on a window of data as it passes through the steam and analyzed iteratively as it comes into the stream. Amazon Kinesis Data Streams, Kinesis Data Firehose and Kinesis Data Analytics allow you to ingest, analyze, and dump real-time data into storage platforms, like Amazon S3, for integration with batch data. Kinesis Data Streams interfaces with Spark

streaming which is run on an Amazon EMR cluster for further manipulation. Kinesis Data Analytics allows you to run analytical queries on the data stream in real-time, which allows you to get a view into the source data and make sure aligns with what is expected from the dataset.

By following the preceding steps, you can create a scalable data transformation platform on AWS. It is also important to note that Amazon Glue, Amazon S3, Amazon Athena and Amazon Kinesis are serverless services. By using these services in the transformation step of the Lambda architecture, we can remove the overhead of maintaining servers and scaling them when the volume of data to transform increases.

Data Analysis

In this phase, you apply your query to analyze data in the three layers:

- **Batch Layer** – The data source for batch analytics could be the raw master data set directly or the aggregated batch view from the serving layer. The focus of this layer is to increase the accuracy of analysis by querying a comprehensive dataset across multiple or all dimensions and all available data sources.

- **Speed Layer** – The focus of the analysis in this layer is to analyze the incoming streaming data in near real-time and to react immediately based on the analyzed result within accepted levels of accuracy.

- **Serving Layer** – In this layer, the merged query is aimed at joining and analyzing the data from both the batch view from the batch layer and the incremental stream view from the speed layer.

This suggested architecture on the AWS platform includes Amazon Athena for the batch layer and Amazon Kinesis Data Analytics for the speed layer. For the serving layer, we recommend using Spark Streaming on an Amazon EMR cluster to consume the data from Amazon Kinesis Data Streams from the speed layer, and using Spark SQL on an Amazon EMR cluster to consume data from Amazon S3 in the batch layer. Both of these components are part of the same code base, which can be invoked as required, thus reducing the overhead of maintaining multiple code bases. The sample code that follows highlights using Spark SQL and Spark streaming to join data from both batch and speed layers.

```
import . . .

DataFrame dataFromS3 = sqlContext.read().json("s3://").toDF();
dataFromS3.registerTempTable("batchData");
. . . The final step in the workflow for the lambda architecture is visualization of
val ssc = new StreamingContext(sc, …)

val kinesisStreams = (0 until numStreams).map { i => KinesisUtils.createStream(ssc, appName, streamName, endpointUrl,
regionName, InitialPositionInStream.LATEST, kinesisCheckpointInterval, StorageLevel.MEMORY_AND_DISK_2) }

val unionStreams = ssc.union(kinesisStreams)

unionStreams.foreachRDD((rdd:RDD[String])=>{
        . . .
        rdd.toDF().registerTempTable("streamData")

        val mergedResult = sqlContext.sql("SELECT ... FROM streamData s JOIN batchData b ON a.data = b.data ...")
        mergedResult.save("s3://... ", "parquet", SaveMode.Overwrite)
}})

ssc.start()
```

Querying Batch
data from Amazon S3 using Spark SQL

Querying
data from Kinesis Stream using
Spark Streaming

Merge query
using Spark SQL

Figure 2: Sample Code

Visualization

The final step in the Lambda architecture workflow is metrics visualization. The visualization layer receives data from the batch, stream, and the combined serving layer. The purpose of this layer is to provide a unified view of the analysis metrics that were derived from the data analysis step.

Batch Layer: The output of the analysis metrics in the batch layer is generated by Amazon Athena. Amazon QuickSight integrates with Amazon Athena to generate dashboards that can be used for visualizations. Customers also have a choice of using any other BI tool that supports JDBC/ODBC connectivity. These tools can be connected to Amazon Athena to visualize batch layer metrics.

Stream Layer: Amazon Kinesis Data Analytics allows users to build custom analytical metrics that change based on real-time streaming data. Customers can use Kinesis Data Analytics to build near real-time dashboards for metrics analyzed in the streaming layer.

Serving Layer: The combined dataset for batch and stream metrics are stored in the serving layer in an S3 bucket. This unified view of the data is available for customers to download or connect to a reporting tool, like Amazon QuickSight, to create dashboards.

Security

As part of the AWS Shared Responsibility Model, we recommend customers use the AWS security best practices and features to build a highly secure platform to run Lambda architecture on AWS. Here are some points to keep in mind from a security perspective:

- **Encrypt end to end.** The architecture proposed here makes use of services that support encryption. Make use of the native encryption features of the service whenever possible. The server side encryption (SSE) is the least disruptive way to

encrypt your data on AWS and allows you to integrate encryption features into your existing workflows without a lot of code changes.

- **Follow the rule of minimal access when working with policies**. Identity and access management (IAM) policies can be made very granular to allow customers to create restrictive resource level policies. This concept is also extended to S3 bucket policies. Moreover, customers can use S3 object level tags to allow or deny actions at the object level. Make use of these capabilities to ensure the resources in AWS are used securely.

- **When working with AWS services, make use of IAM role instead of embedding AWS credentials.**

- **Have an optimal networking architecture in place by carefully considering the security groups, access control lists (ACL), and routing tables that exist in the Amazon Virtual Private Cloud (Amazon VPC).** Resources that do not need access to the internet should not be in a public subnet. Resources that require only outbound internet access should make use of the network address translation (NAT) gateway to allow outbound traffic. Communication to Amazon S3 from within the Amazon VPC should make use of the VPC endpoint for Amazon S3 or a AWS private link.

Getting Started

Refer to the AWS Big Data blog post [Unite Real-Time and Batch Analytics Using the Big Data Lambda Architecture, Without Servers!](#) which provides a walkthrough of how you can use AWS services to build an end-to-end Lambda architecture.

Conclusion

The Lambda architecture described in this paper provides the building blocks of a unified architectural pattern that unifies stream (real-time) and batch processing within a single code base. Through the use of Spark Streaming and Spark SQL APIs, you implement your business logic function once, and then reuse the code in a batch ETL process as well as for real-time streaming processes. In this way, you can quickly implement a real-time layer to complement the batch-processing one. In the long term, this architecture will reduce your maintenance overhead. It will also reduce the risk for errors resulting from duplicate code bases.

Contributors

The following individuals and organizations contributed to this document:

- Rajeev Srinivasan, Solutions Architect, Amazon Web Services

- Ujjwal Ratan, Solutions Architect, Amazon Web Services

Further Reading

For additional information, see the following:

- [AWS Whitepapers](#)

- [Data Lakes and Analytics on AWS](#)

Document Revisions

Date	Description
October 2018	Update.
May 2015	First publication.