Lambda architecture: Working, advantages limitations, and its applications

Athira Soman
athirasoman94@gmail.com
St. Joseph's College of Engineering and Technology,
Palai, Kerala

Smitha Jacob
hodcs@sjcetpalai.ac.in
St. Joseph's College of Engineering and Technology,
Palai, Kerala

ABSTRACT

The amount of data we are now generating is astonishing. Data has also evolved dramatically in recent years, in type, volume and velocity. There are several frameworks available for handling the big data applications. The Lambda Architecture is data processing framework that can handle both batch and stream processing. This paper gives an overview of the Lambda Architecture. Aside from a brief introduction to the Lambda Architecture, major current and envisaged fields of application, as well as advantages, and limitations of use are discussed.

Keywords: Big data processing, Batch processing, Stream processing, Lambda architecture.

1. INTRODUCTION

Over the past few years, data has increased in a large scale in various fields. Under the explosive boom of worldwide facts, the term of massive data is in particular used to describe considerable datasets. As compared with traditional datasets, massive facts generally include loads of unstructured statistics that need extra real-time analysis. Further, huge information also brings about new opportunities for discovering new values, enables us to advantage an in-depth knowledge of the hidden values, and also incurs new demanding situations, e.g., the way to efficaciously arrange and control such datasets.

Data analytics are important for choice guide machine, amassing and processing actual information to assist records to help discover data patterns. Processing frameworks and processing engines are accountable for computing over information in a data system.”Big data” is an expression that was begat to allude to measures of datasets that are so substantial, customary information preparing programming essentially can’t oversee them. For instance, Big data is utilized to choose slants in financial matters, and those patterns and examples are utilized to anticipate what will occur later on. These immense measures of information require stronger computer programming for preparing, best dealt with by information handling systems. Fig.1 shows the main categories of data processing frameworks. There are three categories, batch data processing, stream data processing and hybrid data processing frameworks.

Batch processing has a long history inside the huge information world. Clump handling incorporates working over a vast, static dataset and restoring the outcome at a later time when the calculation is finished. The most famous batch processing framework is Apache Hadoop. Apache Hadoop framework is an open-source community that completely gives the batch processing used for the distributed storage and processing of big data sets.
Stream processing frameworks ascertain above information as it enters the framework. This needs a disparate handling show than the clump case. Rather than noteworthy tasks to apply to a whole dataset, stream processors characterize forms that will be connected to each different information thing as it goes through the framework. Apache Storm is a stream processing system that spotlights on massively low idleness and is perhaps the best alternative for workloads that require close constant preparing. It can deal with vast amounts of information with and bring comes about with less inactivity than different arrangements. Apache Storm and Samza are the main stream processing frameworks.

The hybrid processing frameworks can manage both batch and stream workloads. Apache Spark and Flink are the main hybrid data processing frameworks. Spark provides high speed batch processing and micro-batch processing for streaming. It has wide support, integrated libraries and tooling, and flexible integrations. Flink provides true stream processing with batch processing support.

Lambda architecture is an information processing engineering intended to deal with huge amounts of information by exploiting both batch and stream-processing strategies. This way to deal with design endeavors to adjust dormancy, throughput, and adaptation to internal failure by utilizing cluster handling to give far reaching and precise perspectives of group information, while at the same time utilizing constant stream processing to give perspectives of online information.

2. LAMBDA ARCHITECTURE

The lambda architecture unifies online and batch processing within a single framework. The pattern is suited to applications where there are time delays in data collection and availability through dashboards, requiring data validity for online processing as it arrives. The pattern also allows for batch processing for older data sets to find behavioral patterns as per user needs.

2.1 Batch Layer

The batch layer precomputes results using a distributed processing system that can handle very large quantities of data. The batch layer aims at perfect accuracy by being able to process all available data when generating views. This means it can fix any errors by recomputing based on the complete data set, then updating existing views. Output is typically stored in a read-only database, with updates completely replacing existing precomputed views.

Apache Hadoop is the de facto standard batch-processing system used in most high-throughput architectures.

2.2 Speed Layer

The speed layer processes data streams in real time and without the requirements of fix-ups or completeness. This layer sacrifices throughput as it aims to minimize latency by providing real-time views into the most recent data. Essentially, the speed layer is responsible for filling the "gap" caused by the batch layer's lag in providing views based on the most recent data. This layer's views may not be as accurate or complete as the ones eventually produced by the batch layer, but they are available almost immediately after data is received, and can be replaced when the batch layer's views for the same data become available.
Stream-processing technologies typically used in this layer include Apache Storm, SQLstream and Apache Spark. Output is typically stored on fast NoSQL databases.

2.3 Service Layer

Output from the batch and speed layers are stored in the serving layer, which responds to ad-hoc queries by returning precomputed views or building views from the processed data.

Examples of technologies used in the serving layer include Druid, which provides a single cluster to handle output from both layers. Dedicated stores used in the serving layer include Apache Cassandra, Apache HBase, MongoDB, VoltDB or Elasticsearch for speed-layer output, and Elephant DB (https://github.com/nathanmarz/elephantdb), Apache Impala or Apache Hive for batch-layer output.

3. ADVANTAGES OF LAMBDA ARCHITECTURES

As a result, emphasizes retaining the input data unchanged. Also, the discipline of modeling data transformation. Moreover, this is one of the things that make large MapReduce workflows tractable. As it enables you to debug each stage independently.

This highlights the problem of reprocessing data. As the reprocessing process is one of the key challenges of stream processing. Also, by this process, input data over again to re-derive output. This is a completely obvious but often ignored requirement. Also, a code will always change.

During Big Data management, the very fact that another layer is getting added to architecture brings with it many benefits. Few of the advantages are:

- Processing of data is accurate and precise with no loss to information (alerts, insights etc.) by the real time layer.
- Introduction of a new layer is balanced by reduction in the random write storage needs.
- There is an option to switch data at predefined intervals and also version it, by the batch write storage.
- Opportunity to recover from manual errors due to the introduction of the data sink of raw data.
- Enhance data extraction or learning algorithms to apply on the entire dataset.
- Build immutability and re-computation into the system.
- Efficient architecture in which the batch and stream processing work together to answer multiple use cases.
- Execution of ad-hoc queries against any type of data to obtain end results.

4. LIMITATIONS OF LAMBDA ARCHITECTURE

There is a problem with the Lambda Architecture. That is to maintain the code. Also, that needs to produce the same result in two complex distributed systems. That is exactly as painful as it seems like it would be. To do programming in frameworks like Storm and Hadoop is complex. Also, the code ends up being towards the framework it runs on.

Why can’t the stream processing system be improved to handle the full problem set in its target domain?

To fix this we have only one approach that is we need to have a language or either framework. Moreover, that abstracts over both the real-time and batch framework. You can easily write your code using this higher level framework. Then it “compiles down” to stream processing or MapReduce under the covers. “Summingbird” is an only framework that can easily do this. Furthermore, this will definitely make things a little better, but I don’t think it solves the problem.

5. APPLICATIONS OF LAMBDA ARCHITECTURE

As we know it is an emerging paradigm in Big Data computing. However, log ingestion and accompanying analytics are use cases of Lambda-based applications. Moreover, log messages often are created at a high velocity. Also, they are immutable. Also, we can call it as the “fast data”. The ingestion of each log message does not require a response to the entity that delivered the data. It is a one-way data pipeline.

For example, we can say that the analytics for website click logs could be counting page hits and page popularity.

Lambda architecture can be applied to any Big Data architecture, specially immutability and re-computation. Hadoop is one framework that allows you to store your data and then append new records to the master dataset. You need not have a complicated system to find and update individual records, you can now just append new immutable records to your master set. In the case of immutable records, it is a version of a record at a particular instance. Previous state can always be maintained since new versions get created but the older ones are never gone. Hence, handling of bad records is much easier in this architecture.

6. CONCLUSION

The paper gave an overview of the current state and trends of Big Data processing Framework (Lambda Architecture). Even though numerous limitations and unresolved issues still hinder the widespread application of Lambda Architecture. The main advantages of Lambda Architecture is it can handle both volume and velocity requirements of Big Data applications. Lambda Architecture can perform both batch and stream processing into a single framework. Lambda architecture allows multiple data processing scripts
which are tailored to specific data sets. For online processing stream processing can be used to perform calculations as data arrives and batch processing scripts can be created to run on data stored from before. Lambda Architecture is designed to overcome some limitations of existing data processing framework. It takes advantages from both batch and stream processing.

7. REFERENCE