Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2020, 7(2):59-64

Research Article ISSN: 2394 - 658X

Leveraging Lambda Architecture for Efficient Real-Time Big Data Analytics

Chandrakanth Lekkala

Email: Chan.Lekkala@gmail.com

ABSTRACT

In this era of big data, firms struggle with processing, analyzing and making sense of the "big data" in real-time, hence the ability to extract valuable information in time for decision-making. The lambda architecture has become a potent framework that allows extensive data systems to handle large and real-time data sets, both batch and streaming. Through Lambda Architecture, this paper discusses Lambda architecture and its relation to up-to-date data analysis. We will review the core architecture parts, including batch, speed, and serving layers, and then show how they work to provide a practical solution to colossal data processing and service. Moreover, we dive into the usage of Lambda Architecture by employing comprehensive big data technologies, including Apache Kafka for taking data in, Apache Hadoop for batch processing, Apache Spark for stream processing, and Apache Cassandra for serving the results. Moreover, we discuss the positive aspects and the issues that may arise with applying Lambda Architecture and even provide some real examples related to this area. The performance of the scheme we implemented shows that the Lambda Architecture models a sustainable and functional system to exploit big data for quick analytics and decision-making in all areas of life.

Key words: Lambda Architecture, Big Data, Real-time Analytics, Apache Kafka, Apache Hadoop, Apache Spark, and Apache Cassandra.

INTRODUCTION

In the modern era, the amount of data produced and garnered by organizations is beyond our wildest imaginations. Processing and analyzing this information in real time is crucial for business market share and managerial decisions. On the contrary, even though traditional data processing architectures are designed to handle high volumes and varieties of data, they may not perform well enough and prevent timely and effective analytics. The Lambda Architecture proposed by Dubuc [1] confronts this challenge by providing a joint manner to process massive data sets in both batch and real-time.

The Lambda Architecture integrates batch processing and live processing technology to increase the reliability of real-time extensive data analytics operations. It consists of three main layers: the manufacturing stage, the acquiring stage and the delivery stage. The batch layer is involved in processing the entire dataset at a time; thus, it can visualize the dataset comprehensively. The actual time layer, in contrast, deals with incoming data retrieval directly to process instantly to take immediate response and analysis. The serving tier of Hadoop combines outputs from the batch and speed layers, presenting a consolidated image of the data for both downstream applications and users.

This article will go into detail on how the architecture of Lambda can provide real-time big data analytics by aiding the process. We will cover the architecture's fundamental attributes and how it is implemented using the widely used big data technologies. We will also discuss the advantages and disadvantages of architecture. On top of that, we will give some examples of real-world use cases for Lambda Architecture and industry applications to show how practical Lambda Architecture is in different areas of work.

PROBLEM STATEMENT

The exponential growth of data, which poses significant threats to organizations seeking to use big data for real-time decision-making, is now the main issue. Usually, traditional information systems do not work when there is too much data to process quickly, and it is unclear how it should be processed [2]. Having a scalable fault-tolerant infrastructure that can analyze data in real-time is now critical.

LAMBDA ARCHITECTURE OVERVIEW

Batch Layer

In this case, the batch layer can process the whole dataset batch-wise. It processes the raw data from different sources using complex transformations and calculations to provide a complete view. The batch layer usually relies on a framework such as HDFS or Apache Hadoop to distribute the processing of large datasets across a farm of computer nodes [3].

Speed Layer

The data-layer part of the machine performs real-time analysis of incoming data, enabling instant analysis and action. It captures data from streaming sources and performs incremental model updates to the live processing layer based on the most recent data. The pace layer employs stream processing frameworks like Apache Spark Streaming [4] and Apache Storm [5], which are efficient in real-time data processing.

Serving Layer

The serving layer pulls together the results from both the speed and batch layers, which makes an easy single view for the applications and users, which consume the data. One can query and extract the output from the processed data through the access layer. The local tier uses distributed data sets like Apache Cassandra [6] or Apache HBase [7] to manage and serve the data.

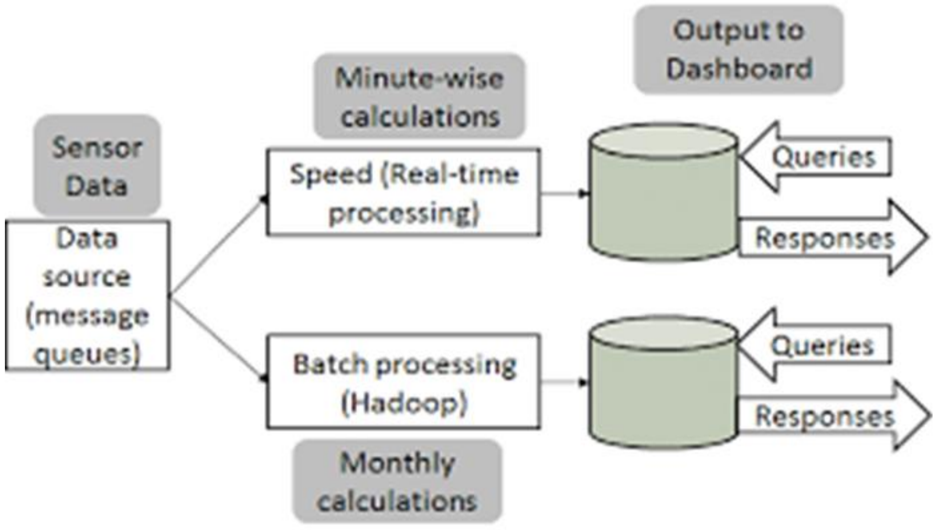

Figure 1: diagram depicting the three layers of the Lambda Architecture and their interactions

IMPLEMENTING LAMBDA ARCHITECTURE FOR REAL-TIME ANALYTICS

Data Ingestion with Apache Kafka

Apache Kafka [8] helps process reliable and scalable real-time event flows instead of a distributed streaming platform for analytics. It functions as a mediator, allowing data generators to broadcast streams and data consumers to register themselves for the streams. Due to fault tolerance, high throughput, and low latency features, Kafka becomes ideal for loading large volumes of real-time data into lambda architecture

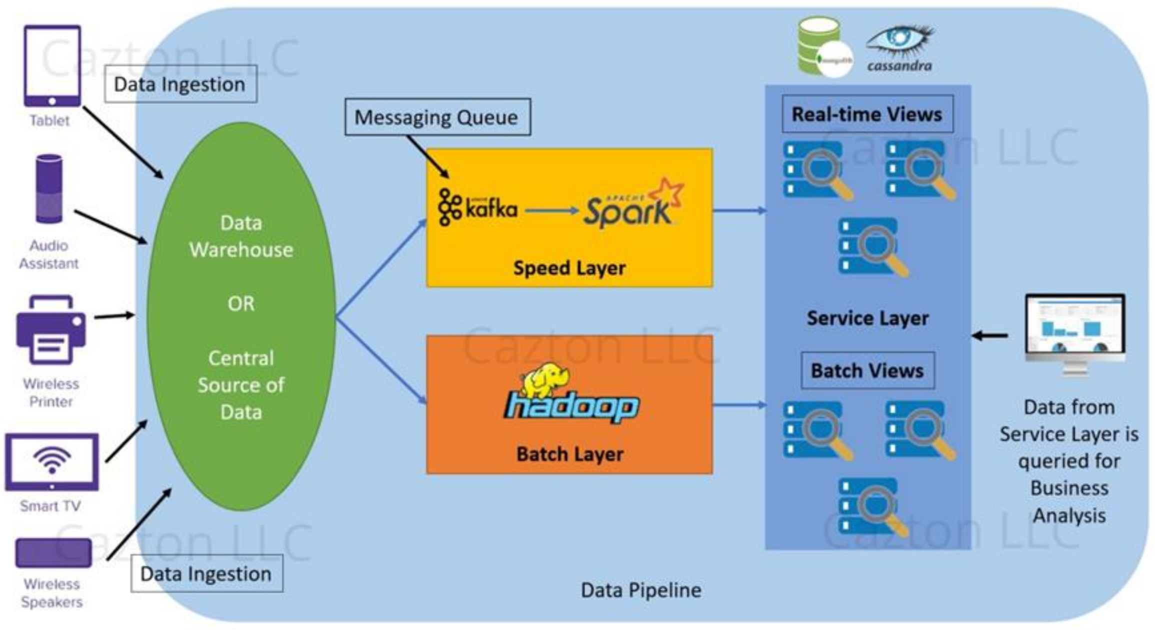

Figure 2: diagram illustrating the role of Apache Kafka in data ingestion within the Lambda Architecture.

BATCH PROCESSING WITH APACHE HADOOP

Apache Hadoop, or a distributed data processing platform, allows workers to be spread across multiple machine clusters, and batch processing can be performed. At the same time, vast amounts of data are being processed. It consists of two main components: the HDFS (Hadoop Distributed File System) for saving data and the Map Reduce paradigm to process it. Hadoop's scalability capability and fault tolerance support the batch layer, making it capable of processing the entire data set

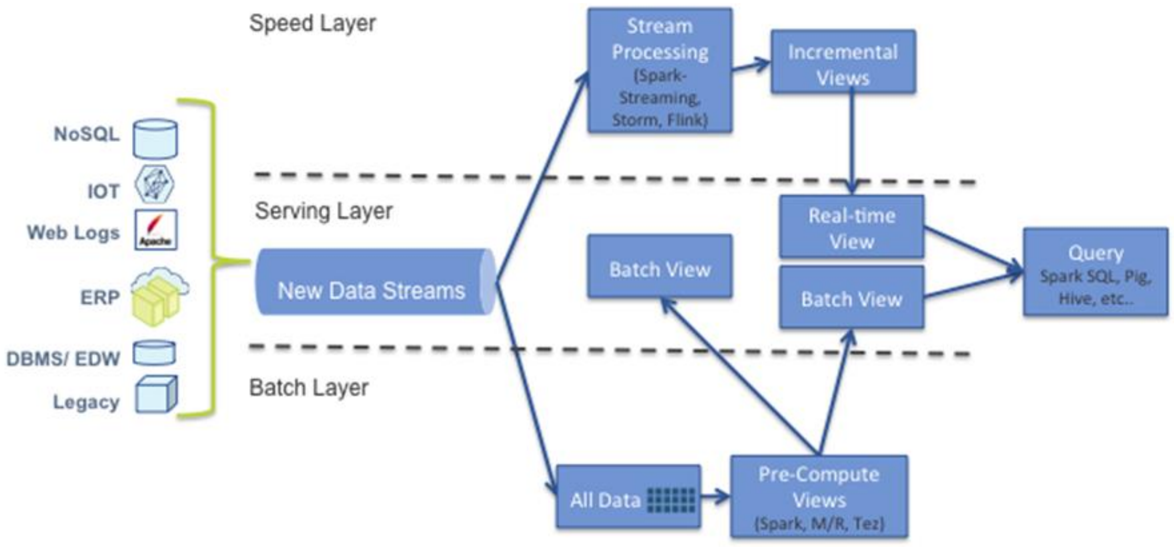

Figure 3: showing the batch processing workflow using Apache Hadoop in the Lambda Architecture.

STREAM PROCESSING WITH APACHE SPARK

Apache Spark [4] is a high-speed general-purpose cluster computing system capable of providing one unified engine for data processing of any scale. One of the components of Spark, Streaming Spark (SS), facilitates the processing of streaming data in real time. It consumes data from several sources, including Kafka, and works with them as micro-batch streams, so it is possible to analyze them in parallel with the data's input. Spark's in-memory computing capabilities and big data frameworks and libraries make it's choice a no-brainer for working on stream processing in the layer of the lambda architecture that is for speed.

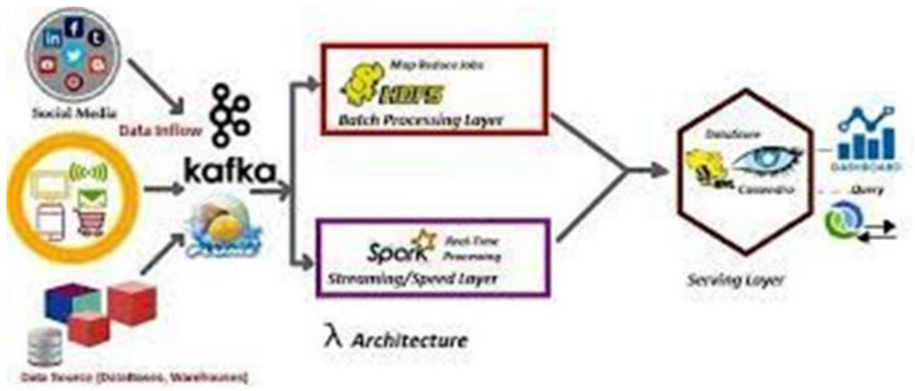

Figure 4: flow diagram displaying how Spark Apache stream processing operates in the lambda architecture

SERVING RESULTS WITH APACHE CASSANDRA

Apache Cassandra [5] is an open-source, distributed NoSQL database built to run across multiple commodity servers and handle massive amounts of structured data. Thus, it is high scalability, fault-tolerant and provides low latency for read/write operations. Cassandra's fast data processing capacity lends itself well to returning results from both the batch-oriented and the speed-oriented layers in the Lambda Architecture for the serving layer.

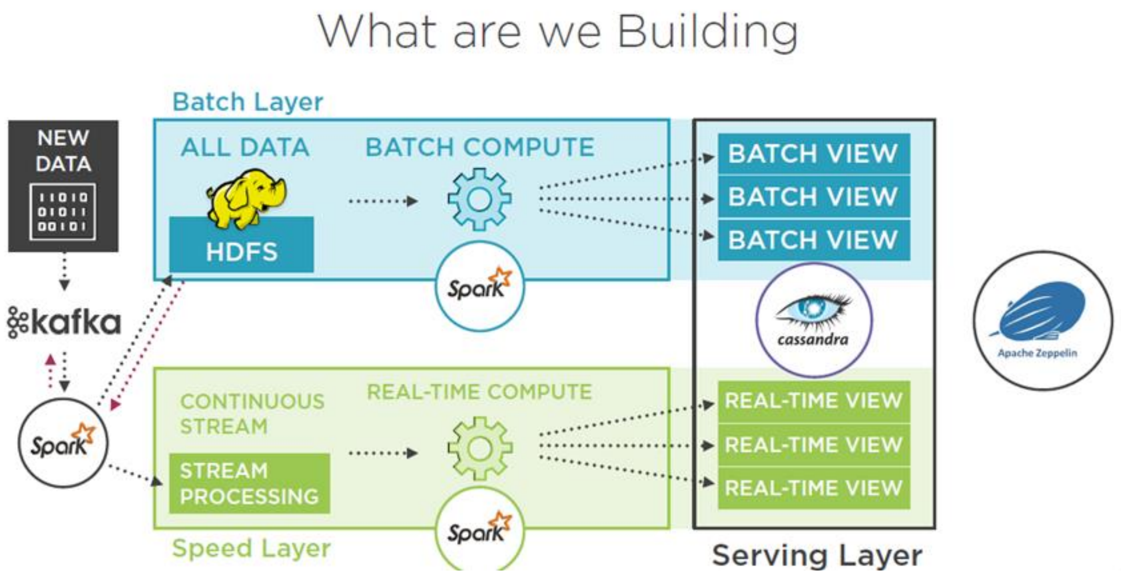

Figure 5: Diagram detailing how Apache Cassandra works in the processing layer in the Lambda Architecture.

BENEFITS AND CHALLENGES

The Lambda Architecture brings several advantages when dealing with real-time big data applications. It delivers scalable, fault-tolerant mapping of large data sets to tackle the growing volume and pace problem. That provides batch and with-time processing architecture, so it can look at data from every angle and react immediately to the sights and actions needed.

Nevertheless, the Lambda Architecture, while having challenges, also has some obstacles. An example is that it requires the management and maintenance of multiple components, including the batch, speed, and serving layers, which are complications to the process. Maintaining data uniformity and synchronization among various layers might become a serious problem, specifically when data changes, record updates, or removal occurs. Additionally, architecture is subject to the creation of latencies, owing to the need to copy and coordinate the data between the layers.

USE CASES AND INDUSTRY APPLICATIONS

The Lambda Architecture has found applications in various domains;

- Fraud Detection: In real-time, multi-tier fraud, detection systems can apply Lambda Architecture to process massive amounts of transaction data to update current data, detect, and flag suspicious activities for immediate action [9].
- Recommendation Engines: E-commerce platforms can follow the Lambda Architecture, i.e., processing user behavior data to generate real-time product recommendations, enhancing user experience and boosting sales.
- IoT Analytics: The Lambda architecture would be an excellent choice for the fast processing and analysis of the data from the Internet of Things (IoT) devices in real time, making it a valuable tool in applications like predictive maintenance, energy optimization, and real-time monitoring [11].
- Social Media Analytics: Social media platforms can utilize Lambda architecture by processing and analyzing user-generated content in real-time, lightening the way to sentiment analysis, trend detection and real-time user engagement [12].

CONCLUSION

Lambda architecture is a robust architecture (framework) that helps build strong and fault-tolerant extensive data systems that can manage batch and real-time processing. Lambda architecture helps bring together the advantages of batch and streaming processing by which organizations can conduct real-time analytics and make to-the-minute decisions using large amounts of data.

As with implementing the Lambda Architecture with technologies such as Apache Kafka for data taking, Apache Hadoop for batch processing, Apache Spark for stream processing, and Apache Cassandra for serving results, organizations can construct these real-time analyzing systems that are also robust and efficient. Although architecture might be a challenge, it provides a solid ground for real-time data processing, scalability, and fault tolerance in big data domains, making it a valuable approach for harnessing big data.

As data gets more extensive in volume, velocity, and variability, the Lambda Architecture will remain a busy and pivotal part of the machinery, which will help to utilize comprehensive data analysis for real-time analysis and decision-making. Future studies can be directed at overcoming the difficulties caused by the architecture, like data consistency and latency, and searching for the best way to exploit new technologies within the framework

REFERENCES

[1]. Warren, J. and Marz, N., 2015. Big Data: Principles and best practices of scalable realtime data systems. Simon and Schuster.

[2]. Dean, J. and Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), pp.107-113.

[3]. Benbrahim, H., Hachimi, H. and Amine, A., 2019. Comparison between Hadoop and Spark. In Proceedings of the International Conference on Industrial Engineering and Operations Management (Vol. 2019, pp. 690-701).

[4]. Aziz, K., Zaidouni, D. and Bellafkih, M., 2019. Leveraging resource management for efficient performance of Apache Spark. Journal of Big Data, 6(1), p.78.

[5]. Fu, M., Mittal, S., Kedigehalli, V., Ramasamy, K., Barry, M., Jorgensen, A., Kellogg, C., Lu, N., Graham, B. and Wu, J., 2015. Streaming@ Twitter. IEEE Data Eng. Bull., 38(4), pp.15-27.

[6]. Cheng, L., Hu, Y. and Lee, P.P., 2019, November. Coupling decentralized key-value stores with erasure coding. In Proceedings of the ACM Symposium on Cloud Computing (pp. 377-389).

[7]. George, L., 2011. HBase: the definitive guide: random access to your planet-size data. " O'Reilly Media, Inc.".

[8]. Kreps, J., Narkhede, N. and Rao, J., 2011, June. Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB (Vol. 11, No. 2011, pp. 1-7).

[9]. Tyagi, A.K., 2019, February. Machine learning with big data. In Machine Learning with Big Data (March 20, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India.

[10]. Kalipe, G.K. and Behera, R.K., 2019. Big Data Architectures: A detailed and application oriented review. Int. Journal Innov. Technol. Explor. Eng, 8, pp.2182-2190.

[11]. Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I.A.T., Siddiqa, A. and Yaqoob, I., 2017. Big IoT data analytics: architecture, opportunities, and open research challenges. ieee access, 5, pp.5247-5261.

[12]. Mane, S.B., Sawant, Y., Kazi, S. and Shinde, V., 2014. Real time sentiment analysis of twitter data using hadoop. IJCSIT) International Journal of Computer Science and Information Technologies, 5(3), pp.3098-3100.