An improved version of lambda architecture

Miguel Landry Foko Sindjoung*, Alain Bertrand Bomgni*, Elie Tagne Fute*,**, Justin Chendjou*

*Department of Mathematics and Computer Science
University of Dschang
Dschang
PO Box 67 Dschang, Cameroon
**Department of Computer Engineering
University of Buea
Buea
PO Box 63 Buea, Cameroon
miguelfoko@gmail.com, alain.bomgni@gmail.com, eliefute@yahoo.fr, chendjoujustin@gmail.com

RÉSUMÉ. La quantité de données produites de façon journalière ne cesse de s'accroitre de nos jours, notamment grâce à l'avènement des objets connectés à internet. Ces objets produisent des quantités de données très importantes (on parle alors de Big Data) et souvent très sensibles qu'il est nécessaire d'analyser en temps réel et/ou sur une période donnée, ceci notament en vue des prises de décisions. C'est dans le but de faciliter l'analyse de données sur ces deux plans que l'architecture Lambda a vue le jour. Cette architecture décrit différentes couches qui peuvent être combinées afin de traiter les données du Big Data. Dans ce document, nous présentons une version améliorée de l'architecture Lambda. Les résultats de notre implémentation montrent une bonne adéquation entre les outils que nous utilisons et le modèle proposé, ce qui fournit des résultats assez encourageants.

ABSTRACT. The amount of data produced on a daily basis is steadily increasing today, especially with the advent of Internet of Things (IoT). These Things produce very large amounts of data (Big Data) and often very sensitive that it is necessary to analyze in real time and/or over a given period, especially for decision making. It is for the purpose of facilitate the analysis of data on both these plans that the Lambda architecture has emerged. This architecture describes different layers that can be combined to process Big Data. In this paper, we present an improved version of the Lambda architecture. The results of our implementation show a good match between the tools we use and the proposed model, which provides quite encouraging results

MOTS-CLÉS : Internet des Objets, Big Data, Architecture Lambda, Traitement Temps Réel, Traitement par lot.

KEYWORDS : Internet of Things, Big Data, Lambda Architecture, Real Time Processing, Batch Processing.

1. Introduction

In recent years there has been a growing production of data due to the advent of the internet of things (IoT) [1]. IoT refers to a set of things (usually sensors) that can produce or capture data and transfer them to the internet network for immediate or subsequent processing. The amount of data produced by things that constitute the IoT is often very huge and evolves exponentially over time. The data produced are so diverse that traditional processing tools and databases are unable to manage them, it is the Big Data. Big Data refers to sets of data that have become so large that they go beyond intuition and the human capacities of analysis and even those of classical processing tools[2, 3]. Given the importance that these data often have (medical application, military, environmental, enterprise, ...), it is important to find mechanisms that allow their treatment in such a way as to reap the full benefit they provide. Some data needs to be processed in real time to immediate decision-making (for example, patient data), while others need to be studied in the long term (eg statistics produced by a company during a given period). It is in order to meet these two constraints that the Lambda architecture has been proposed, that is this architecture is intended to solve the problems of big data in real time and over time. The implementation of the Lambda architecture therefore requires a special knowledge of the appropriate tools for Big Data problem solving. Based on a study of the tools that can intervene in the processing of big data, we propose an improvement of the Lambda architecture which makes it possible to optimize the processing time of Big Data.

2. Related work

This section aims to present a state of the art on the processing of big data. We start by giving the challenges and the characteristics of big data (section 2.1), then, we present the proposed solutions processing of big data in the literature (section 2.2).

2.1. Big Data and its challenges

In this subsection, we present the characteristics of big data and the challenges it faces.

2.1.1. The characteristics of big data

The issue of big data is a hot topic today due to the amount of data that is produced daily. The plurality of data sources, their volume as well as the type of big data information makes it almost impossible to process these data using conventional data processing means. Indeed, IBM's data specialists present the characteristics of big data in a four-dimensional coordinate system [2] whose axes are volume, velocity, variety and veracity. Figure 1 illustrates these coordinates.

– Volume : it is the amount of data available for processing.

– Velocity : helps to measure the speed of generation, the processing and the aggregation of data.

– Variety : refers to different types of generated data (audio, image, videos, ...). These data can be structured or not.

– Veracity : measured or collected data from practical processes must be detected in real time (before any possibility of corruption or manipulation by an external actor).

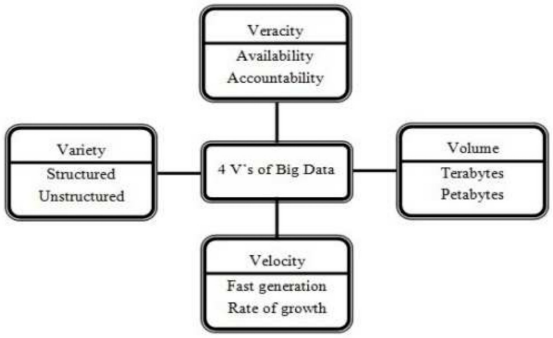

Figure 1 – Big Data's characteristics [1]

2.1.2. Big Data challenges

From the characteristics mentioned in section 2.1.1, several challenges are to be taken up. Indeed, the development of social networks, the multi media, e-commerce, IoT and cloud computing have exploded considerably the volume of data produced [1]. In addition, the need to analyze in real time the data generated by their platforms for the companies renders the traditional processing systems unusable. In this context, new challenges and research problems are encountered [4, 5], among them we have :

– Data management and storage : Because big data uses very large volumes of data that grow exponentially, today's data management systems can not meet the demand because of their limited storage capacity. Moreover, the existing algorithms are not always able to process big data, this because of the heterogeneity of these data. It is therefore interesting to study the possibility of using NoSQL for data backup.

– Data transmission and curation : The amount of data is huge, it is necessary to have an important bandwidth for the transmission of the latter, especially when we know that most of the time they are data from the IoT.

– Data analysis and processing : Response time is an important factor when using big data, especially because the applications that generate these data are mostly very sensitive. In addition, to have real time responses, the processing that is performed on the data must be able to handle very large volumes of data that are inherited. The need to have architectures and tools capable of doing such treatments arises for this purpose.

– Confidentiality and data security : military, medical and many other applications generate confidential data and must be treated with maximum security so that there is no information leak. The majority of data management policies are mostly efficient on static data, but in the context of big data, data varies on a daily basis. Confidentiality and security in the processing of big data is therefore a major objective.

2.2. Current data processing solutions

Data analytics are essential to plan and create decision support systems for optimising the underlying infrastructure. This involves not only processing of the online data, in search for certain events, but also the historical data sources which may be needed to find data patterns which influence decisions. Cloud providers are paramount for the availability and durability to their resources but present various challenges. For instance,

for availability, data is often replicated across multiple servers in different geographical locations, sometimes in untrustworthy locations [6].

Bruns [7] discussed how the current Twitter APIs were extended for third party researchers to deploy their own data analysis on twitter feeds in order to enhance business practices. However unique solutions that allow multiple users of varying backgrounds to write and deploy optimised data processing applications is still needed.

IoT and cloud computing are source of very large volumes of diverse data. Some of the data they produce needs to be analyzed as they arrive (real time processing) while others need to be studied carefully over long period (batch processing). Given the applications from which these data come most often (environment surveillance, monitoring of patients, military applications, online sales companies, ...), it is imperative to find mechanisms that allow not only a real time analysis but also batch processing. It is in the context of performing a data analysis on the two previous plans that $N\ Marz$ et $al.$ [8] proposed the lambda architecture.

The Lambda architecture is a software design pattern that combines both real-time processing and batch processing of big data in a single framework [9]. The figure 2 presents the basic architecture of this design pattern.

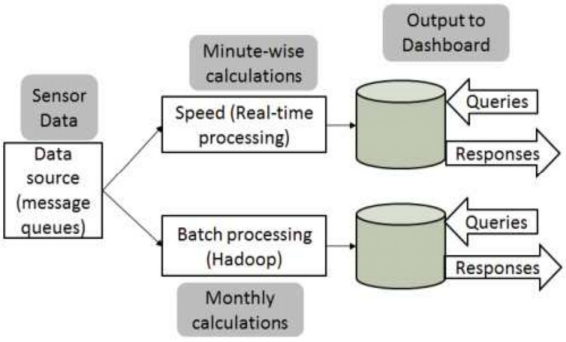

Figure 2 – Basic lambda architecture for real time and batch processing [9]

It caters as three layers (1) Batch processing for pre-computing large amounts of data sets (2) Speed or real time computing to minimize latency by doing real time calculations as the data arrives and (3) a layer to respond to queries, interfacing to query and provide the results of the calculations.

3. An improved version of Lambda architecture

In this section, we present an improved version of the basic Lambda architecture. Indeed, the idea of our solution comes from the fact that the basic architecture does not integrate data ingestion layer, moreover the architecture as presented does not show in a clear way how the old data are obtained for batch processing (it seems that both layers process data in real time). The architecture that we propose is presented in the figure 3.

When data is generated, it is intercepted and ingested by a data-ingestion tool (1). Once the data has been ingested, it is directly made available to a real-time processing tool (2) and at the same time kept in a distributed database for subsequent batch processes (3). During real-time processing, data is regularly processed (4) and the results forming

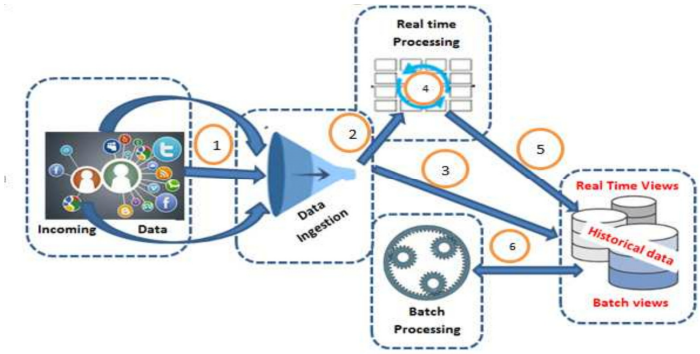

Figure 3 – Our improved lambda architecture

real-time views are stored in a distributed database (5). At considerable time intervals (monthly for example), batch processes are started on the historical data (6) in order to obtain results which will constitute batch views. The batch and real-time views that make up the service layer are therefore merged to answer different user requests.

4. Simulations and results

We present the tools used for our implementation in this section, after which we present our obtained results.

4.1. Big Data Tools used for the simulations

Figure 4 presents the new architecture with tools we used for the implementation. We

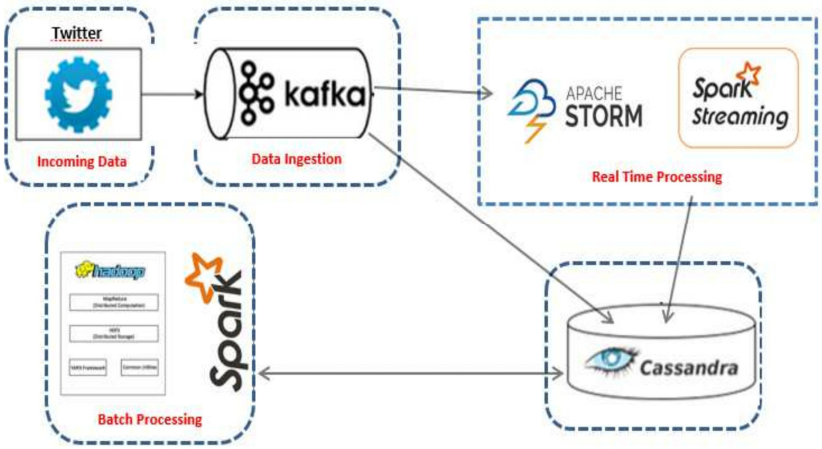

Figure 4 – Our implementation

listen a twitter account as incomming data source in our system, our goal is to count the number of tweets (messages) that arrive in the said account for a given period.

– We use Apache Kafka as data ingestion tool. Kafka[10] is a distributed messaging system that receives and distributes large volumes of data with low latency. It operates according to the producer/consumer model where the data is considered as topics. This means that the producer publishes the topics while the consumer consumes them. The communication between the producer and the consumer is via the HTTP protocol.

– At the batch processing level, we use both Apache Hadoop and Spark, then we make a comparison of results obtained by these two tools. the Hadoop framework[11] with its HDFS, MapReduce and Yarn components enables batch processing. HDFS is a distributed file system that replicates and stores data in cluster machines. MapReduce is a framework for processing and analyzing large volumes of data and Yarn is a framework that aims to separate resource management from the programming model. Although Hadoop is adapted to handle large volumes of data in the context of big data, there are situations where we need the data to remain a little more in memory, in this case, we can think of use of Apache Spark [12] which is a framework to manage large volumes of data just like Hadoop, but with lower latency. It is also important to note that Spark is compatible with the data backup tools used by Hadoop.

– In real time layer, we also make two implementations : one with Apache Storm and another with Spark Streaming and we compare the results. Apache Storm [13] is a popular open source distributed system for processing real-time data. One of the disadvantages of Storm is that it is not able to dynamically optimize between the nodes of the Storm cluster, but that is part of future work in the field. Spark Streaming [14], an extension of Apache Spark is also a distributed system allowing the processing of data in real time. It has a different philosophy than storm. Indeed, in streaming, the received data is stored for a specific time in memory then processed, and returned in Spark RDD (Resilient Distributed DataSet). The disadvantage here is the size of the data to be stored in memory, if it is too short, it can generate multiple RDDs. In addition, in the majority of cases, the data is received through the network, so to ensure the fault tolerance of the data received, Streaming replicates the data through the active nodes.

– Finally, we use Apache Cassandra as distributed database. Apache Cassandra[14] is a distributed storage system for managing very large amounts of structured data spread across the cluster. It provides a highly available, scalable, fault-tolerant, consistent service and is a column-oriented database.

4.2. The obtained results

We make our simulations in a laptop core i3, 4 CPU, 2.4 Ghz; 8 Go of RAM with Ubuntu 14.04, 64 bits as Operating System. We have in our environment a single node in the Hadoop cluster on which we have a NameNode (Master) and a Datanode (Slave). We also have a single supervisor and a single Nimbus Storm where we have our Storm topology constitute by a Spout and three Bolts. The first bolt makes the split operations on tuples. The second makes the filter operations and the last one makes agregation operations. Our datacenter is a Cassandra cluster constitutes by a node.

Figure 5 shows that Storm processes data faster than Spark. Indeed, Storm processes the set of tweets received (220,000) in 1215 seconds (181 Tweets/second) while Spark processes the same tweets in 2475 seconds (90 Tweets/second). This means that Storm's processing speed is twice that of Spark. This allows us to deduce that Storm is better suited for the real-time processing of big data.

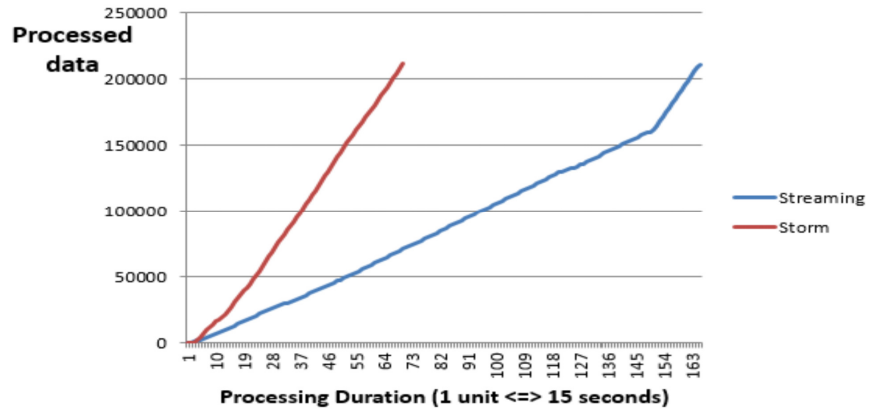

Figure 5 – Performance comparison between Apache Storm and Spark Streaming

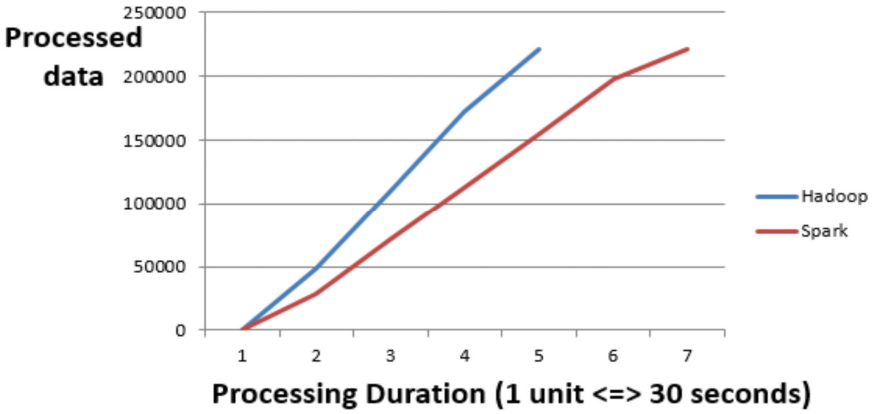

Figure 6 – Performance comparison between Apache Hadoop and Spark

Figure 6 itself presents a comparative curve representing the processed data per unit of time between Hadoop and Spark. On this curve, we notice that Hadoop uses 150 seconds to process the 220,000 tweets present in Cassandra (a speed of 1460 Tweets/s), while Spark takes 210 seconds to process the same amount of data. This allows us to say that Hadoop is faster in batch processing than Spark.

The previous results allow us to conclude that for the implementation of the improved version of the Lambda architecture we present, it is recommended to use Apache Storm as real-time processing tool, and to use Hadoop as batch processing tool.

5. Conclusion and open issues

The lambda architecture [8] is a design pattern that combines real-time processing and batch processing for analyzing big data. Its basic presentation did not include some important aspects for its concrete implementation. In this paper, we are involved in making

a modification on this architecture. Our contribution thus facilitates its implementation. Using tweets from a twitter account as a source of data, we developed an implementation of the new version of lambda architecture, after which we made a comparison between the tools that are used at the real-time and batch layers. The results of our implementation shows that in the implementation of lambda architecture, if we want to have low latency, it is better to use Storm as real-time processing tools and Hadoop as batch processing tools.

Although the results of our implementation are pretty satisfactory, it would be interesting to see the behavior of our implementation when the incoming data is of several varieties and more than the one we used, that is, what will happen if we have 100 000 000 of tweets that arrives per second ? Will the results we obtained be the same ? It might also be interesting to make a comparative study between different data ingestion tools in order to see the real impact of data ingestion in the architecture we proposed. Another perspective would be to ensure that during the data processing by our architecture the security and fault tolerance aspects are taken into account because in the current state this is not the case.

6. Bibliographie

[1] D. P. ACHARJYA, « A survey on big data analytics : challenges, open research issues and tools », *International Journal of Advanced Computer Science and Applications*, vol. 7, n° 2, 2016.

[1] GEORGIOS SKOURLETOPOULOS, CONSTANDINOS X. MAVROMOUSTAKIS, GEORGE MASTORAKIS, JORDI MONGAY BATALLA, CIPRIAN DOBRE, SPYROS PANAGIOTAKIS, EVANGELOS PALLIS, « Big data and cloud computing : A survey of the state-of-the-art and research challenges », *Advances in mobile cloud computing and big data in the 5G Era*, 2017.

[2] SHEN YIN, OKYAY KAYNAK, « Big data for modern Industry : Challenges and Trends », *Proceedings of the IEEE*, vol. 103, n° 2, 2015.

[3] H. HU, Y. WEN, T. -S. CHUA, X. LI, « Towards scalable systems for big data analytics : a technology tutorial », *IEEE Access*, vol. 2, page 652-687, 2014.

[4] H. DEMIRKAN, D. DELEN, « Leveraging the capabilities of service-oriented decision support systems : putting analytics and big data in cloud », *Support Sys*, vol. 55 page 412-421, 2013.

[5] C. L. PHILIP CHEN, CHUN-YANG ZHANG, « Data-intensive applications, challenges, techniques and technologies : a survey on big data », *Information System*, vol. 275 page 314-347, 2014.

[6] M. DIKAIAKOS, G. PALLIS, D. KATSAROS, P. MEHRA, A. VAKALI, « Cloud Computing : Distributed Internet computing for IT and Scientific Research », *IEEE Internet Computing*, 2009.

[7] A. BRUNS, Y. LIANG, L. EUGENE, « Tools and methods for capturing Twitter data during natural disasters. », *First Monday, [S.I]*, 2012.

[8] N. MARZ, J. WARREN, « Big data : principles and the best practices of scalable realtime data systems », *Manning Publications*, 2013.

[9] MARIAM KIRAN, PETER MURPHY, INDER MONGA, JON DUGAN, SARTAJ SINGH BAVEJA, « Lambda architecture for cost-effective batch and speed big data processing », *2015 IEEE International Conference on Big Data (Big Data)*, page 2785-2792, 2015.

[10] , « Apache kafka, Available online », *http://kafka.apache.org/*, , accessed on 21 March 2018.

[11] SHVACHKO K, Kuang H, Radia S, Chansler R., « The hadoop distributedfile system. In : Mass storage systems and technologies (MSST) », *2010 IEEE 26th Symposium on Incline Villiage, Nevada, USA*, page 1-10, May 2010, http://dx.doi.org/10.1109/MSST.2010.5496972.

[12] , « Apache spark, Available online », *https://spark.apache.org/*, , accessed on 21 March 2018.

[13] TOSHNIWAL A., Taneja S., Shukla A., Ramasamy K., Patel JM., Kulkarni S., Jackson J., Gade K., Fu M., Donham J., « Storm@ twitter », *In : Proceedings of the 2014 ACM SIGMOD international conference on management of data. Snowbird, Utah, USA : ACM*, page 147-56, 2014.

[14] MANUEL DIAZ, Christian Martin, Bartolomé Rubio, « State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing », *Journal of Network and Computer Applications*, page 99-117, 2016.