Big Data Architectures : A detailed and application oriented review

Article · October 2019

CITATIONS
7

READS
14,277

2 authors:

Godson Koffi Kalipe
KIIT University
2 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Rajat Kumar Behera
KIIT, Deemed to be University
26 PUBLICATIONS **169** CITATIONS
SEE PROFILE

Big Data Architectures : A detailed and application oriented review

Godson Koffi Kalipe
School of Computer Engineering
Kalinga Institute of Industrial Technology (Deemed to be University)
Bhubaneswar, Odisha , India
godsonkalipe@gmail.com

Rajat Kumar Behera
School of Computer Engineering
Kalinga Institute of Industrial Technology (Deemed to be University)
Bhubaneswar, Odisha, India
rajat_behera@yahoo.com

***Abstract*— Big Data refers to huge amounts of heterogeneous data from both traditional and new sources, growing at a higher rate than ever. Due to their high heterogeneity, it is a challenge to build systems to centrally process and analyze efficiently such huge amount of data which are internal and external to an organization. A Big data architecture describes the blueprint of a system handling massive volume of data during its storage, processing, analysis and visualization. Several architectures belonging to different categories have been proposed by academia and industry but the field is still lacking benchmarks. Therefore, a detailed analysis of the characteristics of the existing architectures is required in order to ease the choice between architectures for specific use cases or industry requirements. The types of data sources, the hardware requirements, the maximum tolerable latency, the fitment to industry, the amount of data to be handled are some of the factors that need to be considered carefully before making the choice of an architecture of a Big Data system. However, the wrong choice of architecture can result in huge decline for a company reputation and business. This paper reviews the most prominent existing Big Data architectures, their advantages and shortcomings, their hardware requirements, their open source and proprietary software requirements and some of their real-world use cases catering to each industry. For each architecture, we present a set of specific problems related to particular applications domains, it can be leveraged to solve. Finally, a trade-off comparison between the various architectures is presented as the concluding remarks. The purpose of this body of work is to equip Big Data architects with the necessary resource to make better informed choices to design optimal Big Data systems.**

Keywords*— *Big Data Architecture , Big Data Architectural Patterns, Big Data Use Cases

I. INTRODUCTION

The non-stop growth of data, the frantic releases of new electronic devices and the data-driven decision-making trend in companies is fueling a constant demand for more efficient Big Data processing systems. The investment in Big Data architecture has been rapidly growing these past years and according to Gartner, businesses will keep investing more in IT in 2018 and 2019 focusing on IOT, Block-chain and Big Data [1,2]. 178 billion dollars were spent on Data Center Systems in 2017 and that number is expected to increase in the coming years [5]. Considering the important funds companies invest in their Big Data solutions, it appears obvious that a careful planning should be done ahead of time before the actual implementation of a solution. However, according to the McKinsey institute, many organizations, today, are facing difficulties because of the absence of architectural planning of their data management solutions [38]. They develop overlapping functionalities and are not able to achieve sustainability because they usually develop technology driven solutions.

An early and careful Big data system architecting considers a holistic data strategy while focusing on real business objectives and requirements. It is of the utmost importance to write down the current but also future needs in order to take scalability considerations into account from the earliest stages of the design of the Big data system. Once that list of use cases and requirements is made clear, a company can move forward and select among the many existing Big Data architectures the most suitable for its use.

The Lambda architecture was one of the first architectures to be proposed for Big Data processing and it has been established as the standard over time [4]. The Kappa architecture came next, followed by several other architectures [3] designed to be able to address some of the limitations of the lambda architecture for use cases where the former standard failed to offer satisfying results. In this paper, we discuss different architectures with their optimal use cases along with some of the factors that need to be considered to make the best choice from a pool of candidate architectures. The paper also highlighted whether the architecture to be adopted for a given use case should be built from scratch or incrementally constructed from an existing architecture.

The structure of this paper is described as follows. Section 2 presents an overview of the rise of Big Data and the challenges that have accelerated the need of new tools and architectures. The next section reviews the work that has been done in the Big Data field to survey the domain, propose architectures and eventually compare them. Section 4 gives, for each architecture, a brief description, its advantages and disadvantages, a set of problems it can solve, some of the fields where it can be used and the hardware and open source configuration required to set up an environment based on that architecture. An overall comparison of the architectures discussed is presented in Section 5 and Section 6 concludes the paper.

II. BACKGROUND

Since In 2013, the McKinsey institute reported that there were more than 2 billion Internet users worldwide [55]. In 2018, according to an article published by Forbes, that number has jumped to 3.7 billion users who are performing

over 5 billion searches every day [61, 62]. Social media remains one of the biggest sources of the data produced in the world. According to Domo's "Data Never Sleeps 6.0" report, every minute, Internet users watch more than 4 million videos on YouTube, close to 13 million text messages are exchanged, the weather channel receives 18 million forecast requests and 97 000 hours of video content are streamed on the Internet [63]. The growth is particularly apparent with social media considering companies like Instagram, one of the most used social media platforms in the world, which has grown its active user database, between December 2016 and 2018, from 600 million to 800 million users who are now posting 95 million photos and videos everyday [64]. Companies across various industries are experiencing a similar frenetic data growth. In 2018, Amazon ships 1111 packages every minute and Uber is used to book 1389 rides every single minute [63]. Another main contributor to the data flood is the Internet of Things industry. The International Data Corporation (IDC) and Intel predict that there will be 200 billion iot devices in use by 2020 [65]. Considering that only 15 billion devices were identified in 2015 up from 2 billion in 2006, it is easier to start getting an idea of the exponential rate at which data is growing in size. And of course, all those devices transmit information across networks, sometimes carrying sensitive data intended to trigger immediate reactions. The case of the voice control feature which is now used by 8 million people every month illustrates the point [63].

From all that has been previously described, it is evident that the traditional single machines can no longer process the diverse and humongous amount of data being produced at such a high speed. Several challenges have arisen due to the birth of Big Data. They include data storage issues of course, but also for instance, the need to separate qualitative data from noise and error as fast as possible because of the volatility of the data. Other challenges are faced during the entirety of the data analysis process. During the acquisition of the data, there is a need to filter it, reduce it and associate it with metadata. There is also a need to transform structureless data and eliminate errors from it in a cleaning process before consuming it. Heterogeneous data proceeding from various sources have to be integrated into single data repositories, requiring new designs and systems more complex than the traditional ones. Additionally, to that, most use cases require that integration to be automated. Also, queries need to be easily scaled over different amounts of data and executed in a matter of second for critical use cases. Finally, there is a need to reflect on the design of specific tools to present human friendly interpretation of the data that is being generated. Another category of challenges that have led to the conception of Big Data ecosystems is the management related issues such as privacy and security among so many others [66, 67, 68].

The landscape of Big Data has kept changing since its birth and the storage devices' prices have been considerably reduced while the data collection methods have kept increasing. Nowadays, in the same system, some data arrive at a very fast rate in constant streams while others arrive in big batches periodically. That diversity has led to the creation of Big Data architectures with the intention to accommodate various data flows and solve the issues specific to each of them [69].

III. Literature Review

Many reviews have been done in the field of Big Data. Most of them cover technologies, tools, challenges and opportunities in the field [55]. They try to shed more light on the field of Big Data, present its advantages and inconvenient [56]. For the majority, they review for each of traditional Big Data processing steps from data generation to its analysis, the background, the technical challenges, and the latest advances. There has also been some work done to review Big Data analytics methods and tools [60].

Reference architectures for Big Data ecosystem have been published by top tech companies as IBM [53], Oracle [51], Microsoft [52] and the National Institute of Standards and Technology (NIST) [54]. Various approaches have been used by researchers in order to try to come up with a reference architecture that could be used across industries in a wide variety of use cases. Pekka, P. and Daniel, P. [39] have proposed a technology independent architecture based on the study of seven major Big Data use cases at top tech companies like Facebook, Linkedin or Netflix. They have decomposed the 7 reviewed architectures in a set of components which they have then classified according to their roles in 8 components forming their reference architecture. Nevertheless, not all use cases required all the components of their architecture and by considering other use cases than the reviewed ones, they have acknowledged more components might need to be added. Other authors have followed a similar approach to propose a five-layer reference architecture in [47]. Mert, O. G., & al. [40] have fetched among more than 19 million projects in the Github database and Apache Software Foundation projects list, the ones related to Big Data. They have extracted from 113 documents, including whitepapers and projects documentations across diverse industries, 241 most popular and actively developed open-source tools. The authors have then classified the tools in 11 groups constituting the components of their reference architecture. They have also discussed the suitability of different tools for their architecture's implementation taking in account factors such as timing, data size, platform independency and data-storage model requirements. Reference architectures have also been proposed to address specific issues such as security in Big Data ecosystems. An example is the Big Data Security reference architecture proposed in [50]. Their architecture was extended from the NIST reference architecture to include for each component, tools and specifications to ensure the protection of the elements of interest: encryption for data, authentication and authorization for networks and containerization and isolation for processes' execution. The authors also presented a brief and high-level comparison of their architecture with other existing reference architectures.

There have been several industry specific propositions too, based on the set of requirements of special use cases. Architectural solutions have been proposed in the field of Supply Chain Management [48], Intelligent Transportation Systems [46], telecommunications [45], healthcare [44], communication networks security (for fault detection and monitoring) [43], smart grids in electrical networks [42], Higher education and universities [41]. Those architectures all reuse all or some of the layers defined in the common reference architectures namely: the data sources layer, the

extraction/collection/aggregation layer, the storage layer, the analysis layer and the visualization layer. Each of these industry specific architectures defines its layers' components in terms of the technological tools or features required by the use case.

Existing architectures have extensively been documented over time as they gained popularity. The biggest part of the existing research focuses on two of the most popular ones: The Lambda and Kappa architectures. Zhelev and Rozeva [5] worked to equip data architects with decision-making information by reviewing cloud types, data persistence options, data processing paradigms and tools and also briefly both the Lambda and Kappa architecture specifying each one's strengths and flaws and mentioning in which situation, each one would be suitable to use. Other works have presented both Lambda and Kappa architecture along with some of their strengths and weaknesses [6]. The authors have also presented a short comparison of both architectures before proposing a new architecture to overcome the deficiencies of both the previously discussed ones. The most exhaustive work has been done in [7] where seven popular architectures were described with the software requirements necessary to implement them. Our aim is to extend the work done in [7], by describing not only existing related use cases but also a set of specific problems each architecture can solve given an industrial context.

From an industrial application point of view, a lot of work has been done to provide exposure on how Big Data can be leveraged to provide better services or increase business profit in various fields [8, 9, 10]. [8] provides insights on the kind of hardware required to build a Big Data processing system discussing electric energy, storage, processing and network requirements at a very high level. None of the existing addressed detailed hardware requirements or attempted to classify use cases and target problems architecture wise.

There does not exist yet to the best of our knowledge any reference document using which, a Big Data System architect can be guided to choose among the most popular Big Data architectures knowing the industry of application, the existing hardware architecture, the budget allotted to purchasing new components and the problems the system is expected to solve.

IV. Big Data Architectures

Big Data architectures are designed to manage the ingestion, processing, visualization and analysis of data that are too large or too complex to handle with traditional tools. From one organization to the other, that data might consist of hundreds of gigabytes or hundreds of terabytes. In the context of this paper, the minimum amount we consider as Big data is 1 TB.

A Big data architecture determines how the collection, storing, analysis and visualization of data is done. We also refer to it to define how to transform structured, unstructured and semi-structured data for analysis and reporting. We discuss in this section, five of the most prominent Big Data architectures that have gained recognition in the industry over the years.

A. Lambda Architecture

The lambda architecture is an approach to big data processing that aims to achieve low latency updates while maintaining the highest possible accuracy. It is divided in 3 layers.

The first, "the batch layer" is composed of a distributed file system which stores the entirety of the collected data. The same layer stores a set of predefined functions to be run on the dataset to produce what is called a batch view.

Those views are stored in a database constituting the "serving layer" from which they can be queried interactively by the user.

The third layer called "speed layer" computes incremental functions on the new data as it arrives in the system. It processes only data which is generated between two consecutive batch views re-computation producing and it produces real-time views which are also stored in the serving layer. The different views are queried together to obtain the most accurate possible results. A representation of this architecture is given in Figure 1.

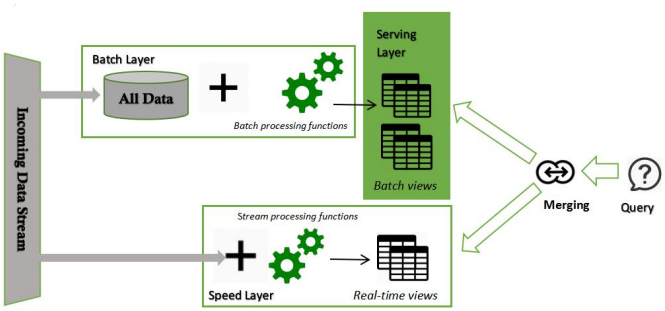

Fig. 1. Lambda Architecture

1) Advantages

Nathan Marz proposed the Lambda architecture (LA) with as first objective to palliate the problems encountered while using fully incremental systems. Such kinds of system have exhibited problems such as operational complexities (online compaction for example), the need to handle eventual consistency in highly available systems and the lack of human fault tolerance. On the contrary, a lambda architecture-based system provides better accuracy, higher throughput and lower latency for reads and updates simultaneously without compromise on data consistency. A LA based architecture is also more resilient thanks to the Distributed File System used to store the master dataset, mostly because it is less subject to human errors (such as unintended bulk deletions) than a traditional RDBMS. Finally, the lambda architecture helps achieve the main requirements of a reliable Big Data system among which are robustness and fault tolerance provided through the batch layer. Each layer of the architecture is scalable independently and the lambda architecture can be easily generalized or extended for a great number of use cases while requiring only minimal maintenance [4]. This architecture provides both real-time data analysis through the ad-hoc querying of real-time views and historical data analysis [11].

2) Drawbacks

The main challenge that comes with the Lambda Architecture is maintaining the synchronization of the batch and speed layers. It consists in regularly discarding the recent data from the speed layer once they have been committed to the immutable dataset in the batch layer.
Another limitation to keep in mind is the fact that only analytical operations are possible from the serving layer; no transactional operation is possible. Finally, one of the major disadvantages of this architecture is the need to maintain two similar code bases: one in the speed layer and another in the batch layer to perform the same computation on different sets of data. That implies redundancy and it requires two different sets of skills in order to write the logic for the streaming and for the batch data [3].

3) Use Cases

Several companies spanning across multiple industries have adopted the Lambda Architecture over time. Many of them are referenced in [29] where specific use cases and best practices around the lambda architecture are collected and made available to those who are interested to work with it.
A particularly suitable application of the Lambda architecture is found in Log ingestion and analytics. The reason is that log messages are immutable and often generated at a high speed in systems that need to offer high availability [12]. The Lambda Architecture is preferred in cases where there is an equal need for real-time/fluid analysis of incoming data and for periodic analysis of the entire repository of data collected. Social media and especially tweets analysis is a perfect example of such an application [12]. But the Lambda architecture can be used in other types of systems to keep track of users subscribing to a meet-up online for instance [13]. The system in [13] is based on the Azure platform and HDInsight Blob Storage is used to permanently store the data and compute the batch views every 60 seconds while a Redis key-value storage is used to persist and display the new registrations between two computation of batch views. The serving layer returns a combination of the results of the two other layers in real-time, via REST webservices, always providing up-to-date information without much overhead. [14] presents an Amazon EC2 based system processing data from various sensors across a city in order to make efficient decisions. While some of those decisions require an on-the-fly analyses of the sensed data, others require that the analyses be performed on massive batches of data accumulated over a long period of time. In such a case, the Lambda architecture, again, reveals itself to be ideal to achieve both objectives.
The lambda architecture is a good choice when data loss or corruption is not an option and where numerous clients expect a rapid feedback, for example, in the case of fraudulent claims processing system [15]. Here, the speed layer using Spark runs in real-time a machine learning model that detects whether a claim is genuine or needs further checking. In that manner, the overall processing time per claim from a user's point of view is considerably reduced.

4) Software requirements

Batch layer. The requirements of the batch layer make Hadoop the most suitable framework to use for its implementation. HDFS provide the perfect append-only technology to accommodate the master dataset. MapReduce, PIG and Hive can be used to develop the batch functions.
Speed layer. The speed layer can be implemented using real-time processing tools such as Storm or S4. Spark Streaming can also be used although it treats data in micro-batches rather than in real streams. The advantage is that the Spark code can be reused of in the batch layer [30].
Serving layer. Any random-access NoSQL database can host the real-time and batch views. Some examples are: HBase, CouchDB, Voldemort or even MongoDB. Cassandra is particularly preferred because of the write-fast option that it provides.
Queuing system. A queuing system is necessary to ensure asynchronous and fault-tolerant transmission of the real-time data to the batch and speed layer. Popular options include Apache Kafka or Flume.

5) Hardware requirements

The hardware requirements presented here are estimated for 1 TB of data. For the calculation, we use a method detailed in [15]. In order to exploit this, one can make the naïve assumption that the hardware requirements grow proportionally with the amount of data to process. The data in the batch layer is usually not stored in a normalized form thus some additional storage space is required, approximately 30% of the original size of the data amounting to a total of 1.3 TB in our case.

TABLE I: LAMBDA ARCHITECTURE HARDWARE REQUIREMENTS

Batch layer	1 replicated master node (6 cores CPU, 4 GB memory, RAID-1 storage, 64-bit operating system) 2 worker nodes (12 cores CPU, 4 GB memory, 2 TB storage, 1 GbE NIC) 1 dedicated resource manager (YARN) node (4 GB memory, and 4core)
Speed layer	Shares the Hadoop node
Serving layer	2 nodes (1TB, 4 cores, 16 GB memory)

Each worker node's raw storage per node (rpsn) was calculated using the formula in equation (1). 2% of the total storage per node (tspn) is reserved for the Operating System and other applications and the remaining storage is divided by Hadoop's default replication factor (rf) 3. Finally, for each 4TB worker node, 653 GB rough space is available to store data.

$$rspsn = \frac{tspn - 2 \times \frac{tspn}{100}}{rf} \quad ...(1)$$

The Spark documentation recommends to run Apache Spark on the same node as Hadoop if possible [32]. Either way, to get a proper idea of the exact Spark hardware requirements, it is necessary to load the data in the Spark system and use the Spark monitoring feature to see how much memory it consumes.
Another important point to note is that, according to the Cassandra's documentation, it is recommended to keep the utilization of each 1TB node to around 600GB [33]. Beyond that threshold, it is not uncommon to observe timeout rates and mean latencies explode and node crashes.

B. Kappa Architecture

The Kappa architecture was proposed to reduce the lambda architecture's overhead that came with handling two separate code bases for stream and batch processing. Its author, Jay Kreps, observed that the necessity of a batch processing system came from the need to reprocess previously streamed data again when the code changed. In Kappa architecture the batch layer was removed and the speed layer enhanced to offer reprocessing capabilities. By using specific stream processing tools such as Apache Kafka, it is henceforth possible to store streamed data over a period of time and create new stream processing jobs to reprocess that data when it's needed replacing batch processing jobs. The functioning process is depicted in Figure 2.

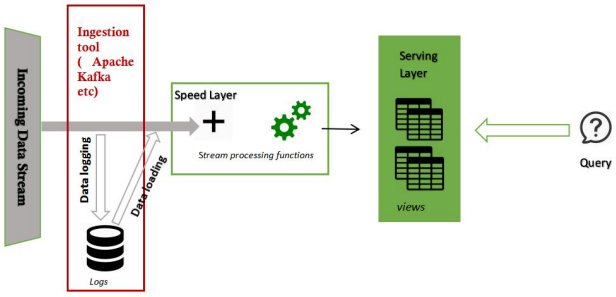

Fig. 2. Kappa Architecture

1) Advantages

Kappa architecture-based systems bring a lot of simplification to data processing. We get the best of both worlds by maintaining a single code base while still allowing historical data querying and analysis through replays. It has fewer moving parts than the Lambda architecture which allows for a simpler programming model as well. Also, it allows replaying streams and recomputing results in case the processing code changes. The incoming data can still be stored in HDFS but we don't rely on it to run reprocessing tasks on historical data.

2) Inconvenients

One of the challenges faced while using this architecture is that only analytical operations are possible not transactional ones. Also, it is not possible to implement the Kappa architecture with native cloud services because they do not support streams with a long Time to live (TTL). It is important to know that the data is not conserved for a long term. In a Kappa architecture-based system, data is kept for a limited predefined period of time after which it is discarded [11].

3) Use cases

The Kappa architecture is particularly suited for real-time applications because it focuses on the speed layer. The author of this architecture, Jay Kreps's company, Linkedin itself has already adopted it. Seyvet & Vielahave [16] presented a detailed implementation of a Kappa architecture for real time analytics of users, network and social data collected by a telco operator. We have inventoried two other use cases, a system for real time calculation of Key Performance Indicator (KPI) in telecommunication and another in the IOT field [17].

4) Software requirements

The software requirements for the Kappa architecture are quite similar to those of the Lambda Architecture minus the Hadoop platform used to implement the batch layer which is absent here.

The preferred ingestion tool is Apache Kafka because of its ability to retain ordered data logs allowing data reprocessing which is essential to the Kappa architecture.

Apache Flink is particularly suitable also for implementing Kappa architecture because it allows building time windows for computations. A popular alternative to it is Apache Samza.

5) Hardware requirements

Table 2 summarizes the hardware requirements for a Kappa architecture based system. IBM knowledge center published a sizing example, recommending the above reported hardware requirements to ingest 1 TB of data [35]. [34] specifies the minimal hardware requirement in a production environment to run Apache Storm in the speed layer.

TABLE II. KAPPA ARCHITECTURE HARDWARE REQUIREMENTS

INGESTION TOOLS	10 SERVERS HAVING EACH :12 PHYSICAL PROCESSORS, 16 GB RAM
SPEED LAYER (STORM)	Minimum one server having : 16 GB RAM, 6 core CPUs of 2 GHz (or more) each, 4 x 2 TB, 1 GB Ethernet
SERVING LAYER	IDEM. TO LAMBDA ARCHITECTURE

Apache Zookeeper is necessary for the functioning of Apache Kafka and can be installed on the primary Apache Kafka server

C. The Microservice Architecture

A system based on the microservice architecture is composed of a collection of loosely coupled services that are able to run independently and to communicate with each other via REST web services, remote calls or Push Messaging. Each service is implemented with the tools and in the language that are most suitable for the task it performs. Each service runs on a dedicated server and has a dedicated storage. The main difference between the microservice architecture and a simple SOA based system is that in the microservice architecture, each service focuses on accomplishing only one specific task and represents a standalone application on its own [20]. The microservice architecture is described in figure 3.

1) Advantages

As compared to monolithic systems, microservice based systems allow for faster development, faster tests and deployments because each service is small and independent from others thus, easier to understand. Thanks to that independence between services, fault tolerance is higher and each service can be developed or rewritten at any time using the newest technology stacks without compromising the other services.

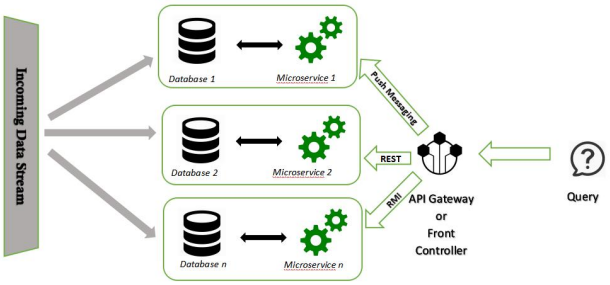

Fig. 3. Microservice architecture

Different teams can work more efficiently by being allocated specific services each. Moreover, services are reusable across a business and any function can be scaled independently from the others.

2) Inconvenients

On the other hand, an inter-service communication mechanism is required and its development is quite complex. There is a need for a strong team coordination and the network communication among the components has to be heavily secured. When two services using two different technological stacks need to communicate, the changes of format (marshalling and unmarshalling) also create an overhead. Though the deployments are faster, they are more complex to setup. Each service usually runs in its own container (possibly a JVM) thus the overall memory consumption is way higher than what is required for a monolithic application [18].

3) Use cases

The microservice architecture has provided a solution for many tech giants such as Amazon, Netflix and eBay as they have to handle a huge number of requests dayly [20].

Modularity can be achieved to a certain extent in monolith applications so some of the factors indicating we need to use a microservice architecture can : the need for decentralization, a high existing (or predictable) traffic. Before making the choice of a microservice architecture, it is also important to keep in mind the consequent investment in time and manpower required from the early stages of the development before the production stage [21]. In [21], the authors describe the implementation of a microservice based scalable mobility service that helps blind users find the most suitable paths for them throughout a city leveraging facilities like bus stops, stairs and audible traffic lights. A microservice is particularly adapted for that use case because some of the services required such as dynamic planner service, crowd-sensing service and travelling service already exist (or can be developed independently as standalone applications and reused). The only services that needed to be developed were the one that the user invokes and a high-level orchestrator service to fetch and provide to the user the useful information. [22] describes the application of microservices in a fraud detection system. Fraud detection systems are extremely time sensitive because, in a matter of seconds, a lot of processing has to be done in order to determine whether or not a transaction is genuine to prevent a potential fraudster to get away with a customer's money. Several microservices leveraging different databases (user past activity database, blacklist database, white list activities database etc.) can quickly and simultaneously perform the necessary checking required and their results are further evaluated by another service to decide if the user should be allowed to proceed or not.

4) Software requirements

Each microservice is technologically independent. It can be developed using any language or technology. Many types of components intervene in the development of microservices and we will be listing giving examples of corresponding tools as used in [31] where the Spring boot framework was used to develop Java based microservices.

Container. Every microservice runs within a container. One tool that can be used to build, deploy and manage containers is OpenShift (particularly suitable for Docker based containers). It orchestrates how and when container-based applications run and allows developers to fix and scale those applications seamlessly. The container management service Docker is used to create containers in which the applications will be developed, shifted and run anywhere.

Distributed version control system. Microservice architecture projects generally imply the existence of several independent teams working on separate microservices. In order to facilitate the collaboration between teams, Git is generally leveraged as source code repository.

Continuous integration tool. Continuous integration/continuous delivery (CI/CD) pipelines are built to facilitate the deployment of services by automating their deployment after they have passed a test suite. Its main objective is to allow early detection of integration bugs. The most popular framework used for that purpose is Jenkins. Other popular options include GitLab CI, Buildbot, Drone and Concourse.

5) Hardware requirements

Table 3 summarizes the hardware requirements for the microservice architecture. The hardware requirements for a multi-node cluster deployment of Docker, as specified in the IBM Cloud Private documentation, are described in [36]. The recommended hardware configuration for Jenkins in small teams has been specified in their documentation [37].

TABLE III . MICROSERVICE ARCHITECTURE HARDWARE REQUIREMENTS

Container management system	1 boot node (1+ core, 4 GB RAM, 100+ GB storage) 1, 3 or 5 master nodes (2+ cores, 4+ GB RAM, 151+ GB storage) 1, 3 or 5 proxy nodes (2+ cores, 4 GB RAM, 40+ GB storage) 1+ worker nodes (1+ cores, 4GB RAM, 100+GB storage) 1+ optional management node (4+ cores, 8+ GB RAM, 100+ GB storage) 2.4 GHz cores recommended
CI/CD	1 node (1+ GB RAM, 50+ GB storage)

D. The Zeta Architecture

The Zeta architecture proposes a novel approach in which the technological solution of a company is directly integrated with the business/enterprise architecture. Any application required by a business can be "plugged in" this architecture. It provides containers which are isolated environments in which softwares can be run and made to interact together independently of the platform incompatibilities. Due to that,

many types of applications can be accommodated and run in a zeta architecture.

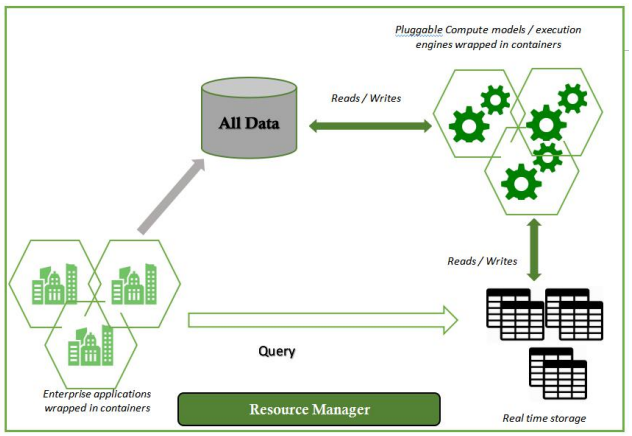

Fig. 4. Zeta architecture

1) Advantages

Since the hardware is not specifically dedicated to any set of services in particular but is common to the entire system, it is better utilized and it can be allocated to serve the most pressing need at any moment. The near real time backups also help avoid over extended recovery periods from failures. The architecture helps to discover issues more quickly too. It facilitates the testing and deployment phases by allowing the creation of binaries that can be deployed seamlessly in any environment without the need to modify them. The example of an advertising platform based on the zeta architecture is presented in [24]. It shows how intermediaries are suppressed by logs directly being saved, read and processed from the same Distributed File System.

2) Use cases

The zeta architecture is suitable for organizations handling real-time data processing as part of their internal business operations. For instance, the example of dynamic allocation of parking lots based on data coming from sensors is a good use case that has been evoked in [23]. It is the architecture leveraged by Google for systems such as Gmail. The zeta architecture is also particularly suitable for complex data-centric web applications, machine learning based systems and for Big data analytics solutions [24].

3) Software requirements

There are many components in the zeta architecture playing different roles that can each fulfilled by several existing tools. We list next some of the tools that can be useful to build a decent zeta architecture-based system.

The Distributed File System hosting the master data is generally implemented using Hadoop Distributed File System while the real-time storage can be implemented using NoSQL or NewSQL databases (HBase, MongoDB, VoldDB etc.). The enterprise applications on the diagram generally consist in web servers or any other business application (varying from one business to the other).

The compute model/execution engine is destined to perform all analytics operation. Any data processing tool that is pluggable can be used for that purpose: MapReduce, Apache Spark and even Apache Drill. Apache Mesos or Apache YARN can serve as global resource manager. The container management system can be chosen among Docker, Kubernetes or Mesos.

4) Hardware requirements

The requirements for each of the software components of this architecture have been described in the previous architectures' sections already. The reader can refer to hardware requirements section for the Lambda, Kappa and microservice for more details.

E. The IoT Architecture

The Internet of Things domain is so vast that no uniform architecture has been defined so far in the field. Nevertheless, several architectures have been proposed by scholars over time [25]. Michael Hausenblas has made an attempt to propose a high abstraction architecture for all IOT projects based on the requirements of an IOT data processing system [26]. The architecture is called iot-a and it is the one we discuss here. It is represented by Figure 5.

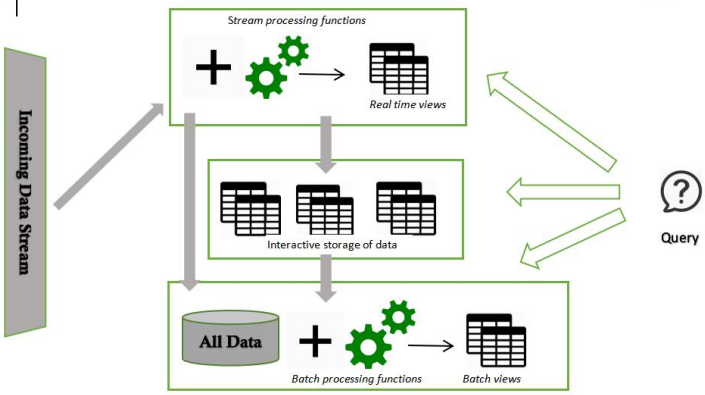

Fig. 5. iot-a architecture

1) Advantages and Inconvenients

There has not yet been enough feedback on projects done using this architecture to provide any thorough evaluation of its performance and eventually of its flaws.

2) Use cases

The discussed architecture is a solution designed to be a good fit for use cases such as smart homes and smart cities [27]. A specific example in the automotive sector describes how the Message Queue/Stream Processing layer helps alert in real-time a car user about failures thus preventing eventual accidents [28]. The Database layer is used here to query the system and obtain information about the status of a car for checkup or in order to develop a repair strategy. Finally, the Distributed File System layer can allow the owner of a car to weekly or monthly assess the overall metrics and performance of his car and possibly identify problems. [28] also lists three other potential use cases of this architectures respectively in biometric database creation (the example of the Aadhaar system in India), financial Services and waste collection and recycling. Each of those use cases requires real-time processing of the data whether to trigger instantaneous notifications or fraud detection alerts. But the interactive aspect is also important in order to generate better routes for trucks or to help target specific companies with banking offers for instance.

TABLE IV. COMPARISON OF THE

Architectures / Features	Lambda	Kappa	Iot-a	Microservices	Zeta
Analysis type	Batch/Real-time	Real-time	Batch / Real-time	Batch/ Real-time	Batch/ Real-time
Processing methodology	Query and reporting	Query and reporting	Query and reporting/ Analytical/ Predictive analysis	Query and reporting/ Analytical	Query and reporting
Data frequency	Real-time feeds	Continuous feeds	On-demand feeds	On-demand feeds	On- demand feeds
Data type	Master data	Transactional	Master data	Transactional data	Transactional data
Content format	Structured, Semi-structured & Unstructured	Structured, Semi-structured & Unstructured	Structured, Semi-structured & Unstructured	Structured, Semi-structured & Unstructured	Structured,Semi-structured & Unstructured
Data sources	Human & Machine generated, web or social media	Machine & Human generated, Web or social media	Machine generated	Internal data sources, machine generated	Web and social media, Internal Data sources
Data consumers	Human	Human	Human/ Other data repositories	Business process	Enterprise applications

Again, the Distributed File System layer can be leveraged with aggregation-like operations to investigate fraud cases that have been flagged over a certain period of time to build a better detection model. The same layer can help municipalities to generate useful reports on local waste recycling activities on a monthly basis.

3) Software requirements

The MQ/SP (Message queuing and Stream processing) layer can be implementing Apache Kafka or fluentd for data collection and Apache Spark or Storm for its processing. The interactive storage layer can be implemented using any NoSQL database along with tools like Apache Drill to interact with it. The DFS layer can use HDFS along with Hive and Apache Mahout for machine learning over the master dataset.

4) Hardware requirements

The requirements for each of the components of this architecture have been described in the previous architectures already. The reader can refer to hardware requirements section for the Lambda, Kappa and microservice for more details.

V. ARCHITECTURES COMPARISON

Table 4 summarizes the discussion about the 5 architectures into a simple format where it can be referred to help Big Data architects make the right choice during the design of a Big Data ecosystem, depending on their needs and requirements.

VI. CONCLUSION

This paper presents a review of the different prominent existing Big Data architectures. We present an overall assessment of the recent review work done in the field of Big Data as a whole. Although there is a plethora of work concerning the characteristics of Big Data itself, its application domains, opportunities, challenges and technologies, there is a lack of comprehensive review work concerning its architecting process. We present reviews of several architectures that have been proposed in industry and in academics in the past years in an attempt to solve real world problems or to serve as roadmap for Big Data architects for a wide range of problems. Then, we focus on five architectures: the lambda architecture, the kappa architecture, the iot-a architecture, the micro service architecture and the zeta architecture. We describe and compare them in terms of the type of processing they can perform and the type of use cases they are suitable for. We

list and briefly explain, for each architecture, a list of real world reliable use cases that can be referred to get guidelines concerning how to make the most out of each architecture during its implementation. We also present known advantages and inconvenient of each architecture as well as a hardware requirements assessment which can help stakeholders plan wisely in terms of budget and infrastructures before going for a Big Data solution.

Big Data architecting is still in its early age and there is still a lack of reliable information about the technical aspects of how the industry is leveraging it. There will need to be a lot more experimentation and applications in order to establish standards and performance statistics to refine the choice of an appropriate architecture. Our work can be further extended with more architectures provided the data concerning their performance and requirements in real world use cases is made available. Nevertheless, even at this stage, we hope it will of great contribution to efficient Big Data ecosystem builders.

REFERENCES

[1] Gartner Says Global IT Spending to Reach $3.7 Trillion in 2018. (2018, January 16). Retrieved from https://www.gartner.com/newsroom/id/3845563

[2] Press, G. (2017, January 20). 6 Predictions For The $203 Billion Big Data Analytics Market.Retrieved fromhttps://www.forbes.com/sites/gilpress/2017/01/20/6-predictions-for-the-203-billion-big-data-analytics-market/#599b23752083

[3] Kreps, J. (2014, July 2). *Questioning the Lambda Architecture.* Retrieved May 26, 2018, *https://www.oreilly.com/ideas/questioning-the-lambda-architecture*

[4] Marz, N., & Warren J. (2015). *Big Data : Principles and best practices of scalable realtime data systems*. Retrieved from https://www.manning.com/books/big-data

[5] Zhelev, S.& Rozeva, A.(2017, December).*Big Data Processing in the Cloud - Challenges and Platforms*. Paper presented atthe 43rd international conference applications of mathematics in engineering and economics, Sozopol, Bulgaria. http://dx.doi.org/10.1063/1.5014007

[6] Ounacer S., Talhaoui M. A., Ardchir S., Daif A.& Azouazi M. (2017). A New Architecture for Real Time Data Stream Processing. *International Journal of Advanced Computer Science and Applications,8*(11), 44-51. http://dx.doi.org/10.14569/IJACSA.2017.081106

[7] Singh, K. , Behera R. J. &Mantri, J. K.(2018, February).*Big Data Ecosystem - Review On Architectural Evolution*. Paper presented at the International Conference on Emerging Technologies in Data Mining and Information Security, Kolkata, India. Retrieved from https://www.researchgate.net/publication/323387483_Big_Data_Ecosystem_-_Review_on_Architectural_Evolution

[8] Kambatla, K., Kollias, G., Kumar,V. &Grama, A. (2014). Trends in Big Data Analytics. *Journal of Parallel and Distributed Computing, 74*(7),2561-2573. https://doi.org/10.1016/j.jpdc.2014.01.003

[9] Chen, M., Mao, S. & Liu, Y.(2014). Big Data : A Survey . *Mobile Networks and Applications, 19*(2),171-209.https://doi.org/10.1007/s11036-013-0489-0

[10] Latinović, T. S., Preradović, D. M., Barz, C. R., Latinović, M. T., Petrica, P. P. & Pop-Vadean A. (2015, November).*Big Data in Industry*. Paper presented at theInternational Conference on Innovative Ideas in Science (IIS2015) , Baia Mare, Romania.https://doi.org/10.1088/1757-899X/144/1/012006

[11] Buckley-Salmon, O. (2017). Using Hazelcast as the Serving Layer in the Kappa Architecture [PowerPoint slides]. Retrieved from https://fr.slideshare.net/OliverBuckleySalmon/using-hazelcast-in-the-kappa-architecture

[12] Kumar, N. (2017, January 31). Twitter's tweets analysis using Lambda Architecture [Blog post]. Retrieved fromhttps://blog.knoldus.com/2017/01/31/twitters-tweets-analysis-using-lambda-architecture/

[13] Dorokhov, V. (2017, March 23). Applying Lambda Architecture on Azure. Retrieved fromhttps://www.codeproject.com/Articles/1171443/Applying-Lambda-Architecture-on-Azure

[14] Katkar, J. (2015).*Study of Big Data Architecture Lambda Architecture* (Master's thesis). Retrieved from http://scholarworks.sjsu.edu/etd_projects/458/

[15] Lakhe, B. (2016). Case Study : implementing Lambda Architecture. In R. Hutchinson, M. Moodie & C. Collins (Eds.)*Practical Hadoop Migration* (pp. 209-251). https://doi.org/10.1007/978-1-4842-1287-5

[16] Seyvet, N. & Viela, I. M. (2016, May 19). Applying the Kappa Architecture in the telco industry. Retrieved fromhttps://www.oreilly.com/ideas/applying-the-kappa-architecture-in-the-telco-industry

[17] Garcia, J. (2015). Kappa Architecture [PowerPoint slides]. Retrieved from https://fr.slideshare.net/juantomas/aspgems-kappa-architecture

[18] Richardson, C. (n.d.). Pattern : Microservice architecture. Retrieved from *http://microservices.io/patterns/microservices.html*

[19] Huston, T. (n.d.).What is microservice architecture?. Retrieved fromhttps://smartbear.com/learn/api-design/what-are-microservices/

[20] Kumar, M. (2016, January 5).Microservices Architecture : What, When, And How?. Retrieved from*https://dzone.com/articles/microservices-architecture-what-when-how*

[21] Melis, A., Mirri, S., Prandi, C., Prandini, M., Salomoni, P. & Callegati, F. (2016, November). *A Microservice Architecture Use Case for Persons with disabilities*.Paper presented at Smart Objects and Technologies for Social Good : Second International Conference, GOODTECHS 2016, Venice, Italy. https://doi.org/10.1007/978-3-319-61949-1_5

[22] Scott, J. (2017,February 21).Using microservices to evolve beyond the data lake. Retrieved fromhttps://www.oreilly.com/ideas/using-microservices-to-evolve-beyond-the-data-lake

[23] Pal, K. (2015, September 28). What can the zeta Architecture do for Enterprise?. Retrieved fromhttps://www.techopedia.com/2/31357/technology-trends/what-can-the-zeta-architecture-do-for-enterprise

[24] Konieczny, B. (2017, April 9). General Big Data. Retrieved from http://www.waitingforcode.com/general-big-data/zeta-architecture/read

[25] Madakam, S., Ramaswamy, R. & Tripathi, S. (2015). Internet of Things (IoT) : A Literature Review. *Journal of Computer Science& Communications, 3*(5), 164-173. http://dx.doi.org/10.4236/jcc.2015.35021

[26] Hausenblas, M. (2015, January 19). Key Requirements for an IOT data platform. Retrieved from*https://mapr.com/blog/key-requirements-iot-data-platform/*

[27] Hausenblas, M. (2014, September 9). iot-a : the internet of things architecture. Retrieved from https://github.com/mhausenblas/iot-a.info

[28] Hausenblas, M. (2015, April 4). A Modern IoT data processing toolbox [PowerPoint slides].Retrieved from https://fr.slideshare.net/Hadoop_Summit/a-modern-iot-data-processing-toolbox

[29] Hausenblas, M. & Bijnens, N. (2014, July 1). Lambda Architecture. Retrieved from http://lambda-architecture.net/

[30] Chu, A. (2016, March 28). Implementing Lambda Architecture to track real-time updates. Retrieved from https://blog.insightdatascience.com/implementing-lambda-architecture-to-track-real-time-updates-f99f03e0c53

[31] Eudy, K. (2018, March 7). A healthcare use case for Business Rules in a Microservices Architecture. Retrieved from https://blog.vizuri.com/business-rules-in-a-microservices-architecture

[32] Hardware provisioning - Spark 2.3.1 documentation (n.d.) . Retrieved from https://spark.apache.org/docs/latest/hardware-provisioning.html

[33] Cassandra/Hardware (2017, May 12). Retrieved from https://wikitech.wikimedia.org/wiki/Cassandra/Hardware

[34] Simplilearn (n.d.). Apache Storm - Installation and Configuration Tutorial. Retrieved from https://www.simplilearn.com/apache-storm-installation-and-configuration-tutorial-video

[35] Example sizing (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSPFMY_1.3.5/com.ibm.scala.doc/config/iwa_cnf_scldc_hw_exmple_c.html

[36] Hardware requirements and recommendations (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSBS6K_2.1.0/supported_system_config/hardware_reqs.html

[37] Installing Jenkins (n.d.). Retrieved from https://jenkins.io/doc/book/installing/

[38] Blumberg, G., Bossert, O., Grabenhorst, H. & Soller, H. (2017, November). Why you need a digital data architecture to build a sustainable digital business. Retrieved from https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/why-you-need-a-digital-data-architecture

[39] Pekka , P., & Daniel, P. (2015). Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems.

Big Data Research 2(4). 166-186. doi : https://doi.org/10.1016/j.bdr.2015.01.001
[40] Mert, O. G., & al. (2017). Big-Data Analytics Architecture for Businesses: a comprehensive review on new open-source big-data tools. *Cambridge Service Alliance*. Retrieved from https://cambridgeservicealliance.eng.cam.ac.uk/news/2017OctPaper
[41] Peter, M., Ján, Š. & Iveta Z. (2014). Concept Definition for Big Data Architecture in the Education System. Paper presented at the 12th International Symposium on Applied Machine Intelligence and Informatics, Herl'any, Slovakia, 2014. https://doi.org/10.1109/SAMI.2014.6822433
[42] Xing, H., Qi & al. (2017). A Big Data Architecture Design for Smart Grids based on Random Matrix Theory. *IEEE Transactions on Smart Grid 8*(2). 674-686. Doi : https://doi.org/10.1109/TSG.2015.2445828
[43] Samuel, M., Xiuyan, J., Radu, S. & Thomas, E. (2014). A Big Data architecture for Large Scale Security Monitoring. Paper presented at IEEE International Congress of Big Data, Anchorage, AK, USA, 2014. https://doi.org/10.1109/BigData.Congress.2014.18
[44] Yichuan, W., LeeAnn, K. & Terry, A., B. (2016). Big Data Analytics : Understanding its capabilities and potential benefits for healthcare organizations. *Technological forecasting and social change 126*. 3-13. doi : https://doi.org/10.1016/j.techfore.2015.12.019
[45] Fei, S., Yi, P., Xu, M., Xinzhou, C., & Weiwei, C. (2016). The research of Big Data on Telecom industry. Paper presented at 16th International Symposium on Communications and Information Technologies (ISCIT), QingDao, China, 2016. https://doi.org/10.1109/ISCIT.2016.7751636
[46] Guilherme, G., Paulo, F., Ricardo, S., Ruben, C. & Ricardo, J. (2016). An Architecture for Big Data Processing on Intelligent Transportation Systems. Paper presented at IEEE 8th International Conference on Intelligent Systems, Sofia, Bulgaria, 2016. https://doi.org/10.1109/IS.2016.7737393
[47] Go, M. S., Lai, X., & Paul, V. (2016). A reference Architecture for Big Data Systems. Paper presented at 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), Chengdu, China, 2016. Doi : 10.1109/SKIMA.2016.7916249
[48] Sanjib, B. & Jaydip, S. (2017). A Proposed Architecture for Big Data Driven Supply Chain Analytics. *ICFAI University Press (IUP) Journal of Supply Chain Management 13*(3). 7-34. Doi : https://arxiv.org/ct?url=https%3A%2F%2Fdx.doi.org%2F10.2139%252Fssrn.2795906&v=b6e0857f
[49] Julio, M., Manuel A. S., Eduardo, F. & Eduardo, B. F. (2018). Towards a Security Reference Architecture for Big Data. Paper presented at 21st International Conference on Extending Database Technology and 21st International Conference on Database Theory joint conference, Vienna, Austria, 2018. Retrieved from https://www.semanticscholar.org/paper/Towards-a-Security-Reference-Architecture-for-Big-Moreno-Serrano/3966ce99e50c741dcb401707c6c8cacd8420d27e
[50] Yuri, D., Canh, N. & Peter, M. (2013). Architecture Framework and Components for the Big Data Ecosystem. Paper presented at International Conference on Collaboration Technologies and Systems (CTS), Minneapolis, MN, USA, 2014. Doi : https://doi.org/10.1109/CTS.2014.6867550
[51] Doug, C., Oracle. (2014). Information Management and Big Data : A Reference Architecture [White paper]. Retrieved from https://www.oracle.com/technetwork/topics/entarch/articles/info-mgmt-big-data-ref-arch-1902853.pdf
[52] Microsoft. (2014). Microsoft Big Data : Solution Brief. Retrieved from http://download.microsoft.com/download/f/a/1/fa126d6d-841b-4565bb26d2add4a28f24/microsoft_big_data_solution_brief.pdf
[53] IBM Corporation. (2014). IBM Big Data & Analytics Reference Architecture v1. Retrieved from https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/e747a4bd-614d-4c5d-a411-856255c9ddc4/document/bbc80340-3bf4-4e0a-8caf-a43f64a22f05/media
[54] NIST NBD-WG. (2017). Draft NIST Big Data Interoperability Framework : Volume 6, Reference Architecture. Retrieved https://bigdatawg.nist.gov/_uploadfiles/M0639_v1_9796711131.docx
[55] Nawsher, K. & al. (2014). Big Data: Survey, Technologies, Opportunities, and Challenges. *The Scientific World Journal 2014*(2014). 1-19. doi : http://dx.doi.org/10.1155/2014/712826
[56] Seref, S. & Duygu, S., (2013). Big Data : A Review. Paper presented at International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, USA. Doi : https://doi.org/10.1109/CTS.2013.6567202
[57] Andrea, M., Marco, G., & Michele, G. (2015). What is Big Data? A Consensual Definition and a Review of Key Research Topics. Paper presented at 4th International Conference on Integrated Information, Madrid, Spain, 2014. Doi : https://doi.org/10.1063/1.4907823
[58] Amir, G. & Murtaza, H. (2014). Beyond the hype : Big data concepts, methods and analytics. *International Journal of Information Management 35*(2). 137–144. Doi : https://doi.org/10.1016/j.ijinfomgt.2014.10.007
[59] Chen, M., Mao, S. & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications 19*(2). 171-209. Doi : https://doi.org/10.1007/s11036-013-0489-0
[60] Elgendy N. & Elragal A. (2014). Big Data Analytics: A Literature Review Paper. Paper presented at Industrial Conference on Data Mining, St. Petersburg, Russia, 2014. doi : https://doi.org/10.1007/978-3-319-08976-8_16
[61] Bernard M. (2018, May 21). How Much Data Do We Create Everyday? The Mind-Blowing Stats Everyone Should Read. Retrieved from https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#4c87a72b60ba
[62] Tom, H. (2017, July 26). How much data does the world generate every minute? Retrieved from https://www.iflscience.com/technology/how-much-data-does-the-world-generate-every-minute/
[63] Josh J. (DOMO) , (2018, June 5). Data Never Sleeps 6.0. Retrieved from https://www.domo.com/blog/data-never-sleeps-6/
[64] Mary, L. (WordStream) (2018, October 2017). 33 Mind-Boggling Instagram Stats & Facts for 2018. Retrieved from https://www.wordstream.com/blog/ws/2017/04/20/instagram-statistics
[65] International Data Corporation (IDC), Intel. A Guide to the Internet of Things. Retrieved from https://www.intel.in/content/www/in/en/internet-of-things/infographics/guide-to-iot.html
[66] Nasser, T., & Tariq, R. S. (2015). Big Data Challenges. *Journal of Computer Engineering and Informatiion Technology 4*(3). 1-10. http://dx.doi.org/10.4172/2324-9307.1000135
[67] Chaowei, Y., Qunying, H., Zhenlong, L., Kai, L. & Fei H. (2017). Big Data and Cloud Computing : Innovation Opportunities and Cloud Computing. *International Journal of Digital Earth 10*(1). 13-53. https://doi.org/10.1080/17538947.2016.1239771
[68] Uthayasankar, S., Muhammad, M. K., Zahir, I. & Vishanth, W. (2016). Critical analysis of Big Data Challenges and Analytical Methods. *Journal of Business Research 70*. 263-286. https://doi.org/10.1016/j.jbusres.2016.08.001
[69] Zoiner, T., Mike, W. (2018, March 31). Big Data architectures. Retrieved from https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/