Big Data Architectures: A Detailed and Application Oriented Analysis

Godson KoffiKalipe, Rajat Kumar Behera

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 data which are internal and external to organizations. 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. The purpose of this body of work is to equip Big Data architects with the necessary resources to make better informed choices to design optimal Big Data systems.***

Index Terms: Big Data Architecture , Big Data Architectural Patterns, Big Data Use Cases

I. INTRODUCTION

The investment in Big Data has been growing rapidly these past years and will continue to in 2019, according to Gartner [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 is obvious that a careful planning has to 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 instead of focusing on business requirements. Having a clear list of current and future needs in order to take scalability considerations into account from the earliest stages of the design of the Big data system is of the utmost importance for a company to choose the most suitable Big Data architecture 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 palliate the limitations of the lambda architecture. 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 structure of this paper is described as follows. Section 2 reviews the work that has been done in the Big Data field to survey the domain, propose architectures and eventually compare them. Section 3 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 software configuration required to set up an environment based on that architecture. An overall comparison of the architectures discussed is presented in Section 4 and Section 5 concludes the paper.

II. LITERATURE SURVEY

Most Big Data reviews 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]. The majority focuses on technical challenges, and the latest advances but also on analysis methods and tools [57]. 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 [39, 40, 47, 49, 50]. Most of those architectures were derived from existing Big Data projects whose components were grouped into modules. Industry specific architectural solutions have also 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 reused 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.

Revised Manuscript Received on July 05, 2019

Godson KoffiKalipe, School of computer Engineering, Kalinga Institute of Industrial Technology, Odisha, India.

Rajat Kumar Behera, School of computer Engineering, Kalinga Institute of Industrial Technology, Odisha, India.

Layers' components are generally defined by a set of technological tools or features. 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 [5, 6]. 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]. Yet, none of the existing addressed detailed hardware requirements or attempted to classify use cases and target problems architecture wise. There does not yet exist 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.

III. 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. We also refer to it to define how to transform structured, unstructured and semi-structured data for analysis and reporting. 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. 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 (LA) is an approach to big data processing that aims to achieve low latency updates while maintaining the highest possible accuracy. It has 3 layers (Fig. 1.) : "The batch layer" is a distributed file system storing the entirety of the collected data and predefined functions which produce batch views from the dataset . "The speed layer" computes incremental functions on the new data as it arrives in the system between two consecutive batch views re-computation producing real-time views. "The serving layer" stores those views for interactive querying by the user.

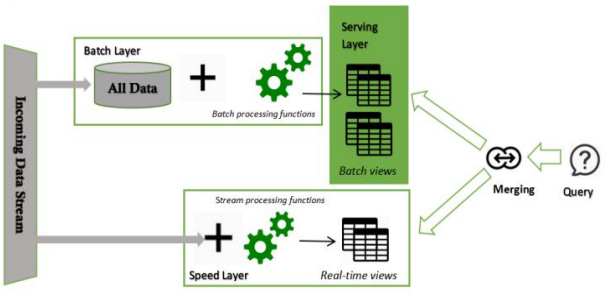

Fig.1. Lambda Architecture.

Advantages & Drawbacks

Nathan Marz proposed the LA primarily to palliate operational complexities (online compaction for example), handling of eventual consistency in highly available systems and human fault tolerance issues, encountered while using fully incremental systems. On the contrary, a LA-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 and reliable 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). 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]. The main challenge with the Lambda Architecture is maintaining the synchronization of the batch and speed layers by regularly discarding the recent data from the speed layer once 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. We also need to maintain two similar code bases for the speed layer and for the batch layer to perform the same computation on different sets of data. It leads to redundancy and calls for two different sets of skills in order to write the logic for both needs [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.
Log messages being immutable and often generated at a high speed in highly available systems, their ingestion and analysis suits particularly the LA [12]. LA 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 as in social media and especially tweets analysis [12]. The system in [13] keeps track of users' subscription to an online meet-up 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 web services, 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. The LA 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] where the overall processing time per claim from a user's point of view can be considerably reduced.

Software requirements

Table 1 summarizes the software requirements for the Lambda Architecture. Spark Streaming can also be used in the streaming 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]. Cassandra is particularly preferred for the serving layer because of the write-fast option that it provides. A queuing system is necessary to ensure asynchronous and fault-tolerant transmission of the real-time data to the batch and speed layer.

Table 1. Software requirements for the Lambda Architecture

Batch layer	HDFS (Storage) + MapReduce/PIG/Hive (Functions)
Speed Layer	Storm / S4 / Spark Streaming
Serving Layer	NoSQL database (Cassandra/ Hbase / CouchDB/ Voldemort/ MongoDB)
Queuing System	Apache Kafka / Flume

Hardware requirements

The hardware requirements presented here are estimated for 1 TB of data. For the calculation, we use the 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. Each worker node's raw storage per node(rpsn) is 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.

$$\square\square\square\square\square = \frac{\square\square\square\square - 2 \times \frac{\square\square\square\square}{100}}{\square\square} \ldots(1)$$

Table 2. Hardware requirements for the lambda architecture

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)

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. Also, according to the Cassandra's documentation, it is recommended to keep the utilization of each 1TB node to around 600GB [33]. Beyond that threshold, timeout rates and mean latencies generally explode and node crashes. All the estimated requirements are presented in Table 2.

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 is 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.

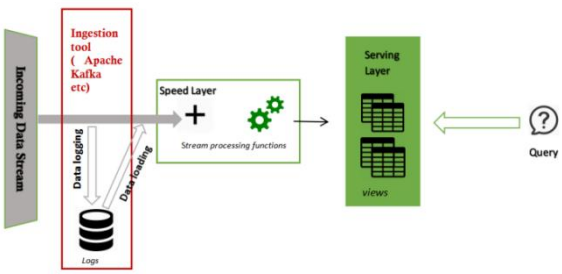

Fig 2. Kappa Architecture

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Advantages & Drawbacks

Kappa architecture simplifies data processing. It combines the best of both worlds by maintaining a single code base while still allowing historical data querying and analysis through stream replays when code changes. It has fewer moving parts than the Lambda architecture which allows for a simpler programming model as well. The incoming data can still be stored in HDFS but we don't rely on it to run reprocessing tasks on historical data.

On the other hand, only analytical operations are possible not transactional ones. It cannot be implemented with native cloud services because they do not support streams with a long Time to live (TTL). Furthermore, the data is kept for a limited predefined period of time after which it is discarded [11].

Use cases

The Kappa architecture is particularly suited for real-time applications because it focuses on the speed layer. The author of this architecture'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].

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. Because it can retain ordered data logs allowing for data reprocessing, Apache Kafka is preferred for ingestion. Apache Flink is particularly suitable for processing as it allows building time windows for computations. A popular alternative to it is Apache Samza.

Hardware requirements

Table 3 summarizes the hardware requirements for the Kappa architecture as recommended by IBM knowledge center [34, 35]. Apache Zookeeper is necessary for the functioning of Apache Kafka and can be installed on the primary Apache Kafka server.

Table 3. 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

C. **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 communicate with each other via REST web services, remote calls or Push Messaging. Each service is implemented with the tools and language that are most suitable for its purpose and runs on a dedicated server having a dedicated storage. The main difference between the microservice architecture and a simple Service Oriented Architecture based system is that here, each service focuses on accomplishing only one specific task and represents a standalone application[20]. The microservice architecture is described in Figure 3.

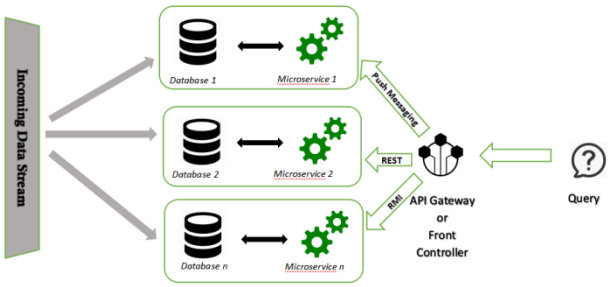

Fig. 3.Microservice architecture

Advantages & Drawbacks

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. Fault tolerance is higher and a service can be (re)written at any time using the newest technology stacks without compromising the other services. 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.On the other hand, the development is complex and there is a need for a strong team coordination, an inter-service communication mechanism and security measures for network communication among the components. When two services using two different technological stacks need to communicate, the changes of format (marshalling and unmarshalling) also create an overhead. 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].

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 daily [20]. Some of the factors indicating a need of a microservice architecture are : the need for decentralization, a high existing (or predictable) traffic. 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.

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. 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 [22]. 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.

Software requirements

Each microservice is technologically independent and can be developed using any language or technology [31]. A microservice runs within a container which orchestrates how and when container-based applications run and allows developers to fix and scale those applications seamlessly. The container management service helps to create containers in which the applications will be developed, shifted and run anywhere. In order to facilitate the collaboration between teams, Git is generally leveraged as source code repository. 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. Table 4 groups together the tools required to set up a microservice architecture based system.

Table 4.Microservervice architecture software requirements

Container	OpenShift (Docker based containers)
Distributed Version Control System	Git
Continuous Integration tool	Jenkins / GitLab CI, Buildbot, Drone, Concourse

Hardware requirements

Table 5 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 5.Microservice architecture hardware requirements

Container management system	**1 node (1+ GB RAM, 10+ GB storage) 4+ GB RAM, 150+ GB storage) 4 GB RAM, 40+ GB storage) 4GB RAM, 100+ GB storage) 4GB 2.4 GHz cores recommended**
CI/CD	1 node (1+ GB RAM, 50+ GB storage)

D. 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 software can be run and made to interact together independently of the platform incompatibilities. It is described in Figure 4.

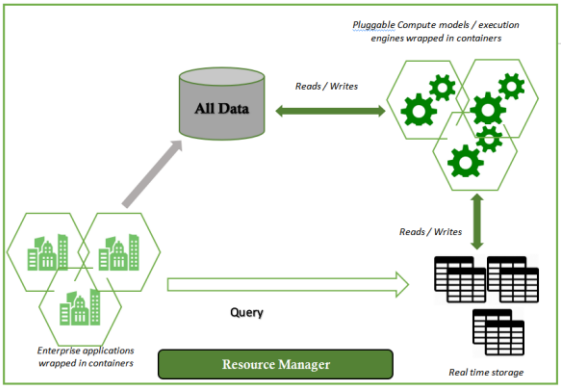

Fig. 4. Zeta architecture

Advantages & Drawbacks

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 help avoid over extended recovery periods from failures and issues diagnosis is quicker. The creation of binaries that can be deployed seamlessly in any environment without the need to modify them, makes testing and deployment easier. The advertising platform based on the zeta architecture presented in [24] shows how intermediaries are suppressed by logs directly being saved, read and processed from the same Distributed File System.

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].

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 in Table 6. The enterprise applications on the diagram generally consist in web servers or any other business application (varying from one business to the other). Table 6.summarizes the available data.

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.

Table 6. Zeta architecture software requirements

Distributed File System	Hadoop DFS
Real-time storage	NoSQL/NewSQL (HBase, MongoDB…)
Compute model engine	MapReduce, Spark, Apache Drill
Resource Manager	Apache Mesos, YARN
Container Management System	Docker, Kubernetes, Mesos

E. IOT Architecture

The Internet of Things domain is so vast that no uniform architecture has been defined so far for the field. Nevertheless, several architectures have been proposed by scholars over the course of 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 in Figure 5.

Fig. 5.iot-a architecture

Advantages and Drawbacks

The scarcity of feedback on projects done using this architecture has prevented us from being able to provide a thorough evaluation of its performance and eventually of its flaws.

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. 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.

Software & Hardware requirements

The MQ/SP (Message queuing and Stream processing) layer can be implemented using 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. The hardware 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.

IV. ARCHITECTURES COMPARISON

Table 7 summarizes the discussion about the 5 architectures presented in this paper.

V. CONCLUSION

In this paper, we have presented 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 architecture design process. Therefore, we have presented reviews of several architectures that have been proposed in industry and in academics in the past years in an attempt to provide a roadmap for Big Data architects. Big Data architecture design 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.

Table 7. Trade-off 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 data	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

REFERENCE

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 from https://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).MicroservicesArchitecture : 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. [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. (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 (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. (2014). Big Data: Survey, Technologies, Opportunities, and Challenges. *The Scientific World Journal 2014*(2014). 1-19. doi :http://dx.doi.org/10.1155/2014/71282
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. 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

Authors Profile

Godson KALIPE is a software engineering graduate currently pursuing his Mtech in KIIT Deemed to beUniversity, Bhubaneswar, Odisha, India where he has been conducting research in the fields of Machine Learning and Big Data for the past two years

Rajat Kumar Behera is working as associate professor in School of Computer Science & Engineering, KIIT Deemed to be University, Bhubaneswar, Odisha, India. He is a PMP, ITIL, Six Sigma certified and holds 15 years of industry experience & two years of teaching experience. His area of interest includes Data Science & Software Engineering and can be reached at rajat_behera@yahoo.com