Big Data Ecosystem: Review on Architectural Evolution

Kamakhya Narain Singh, Rajat Kumar Behera and Jibendu Kumar Mantri

Abstract Big data is the collection of large datasets of different varieties, generated at an alarming speed with noises and abnormality. It is primarily popular for its five Vs namely volume, variety, velocity, value, and veracity. In an extremely disruptive world of open source systems that are playing a dominant role, big data can be referred to as the technology that can address the storage, processing, and visualization of such data which is too diverse and fast-changing. It has led to a complex ecosystem of new frameworks, tools, and libraries that are being released almost every day, which creates confusion as the technologists tussle with a swamp. This survey paper discusses the big data architecture and its subtleties that might help in applying the appropriate technology as a use case.

Keywords Big data · Big data ecosystem architecture
Big data processing and big data storage

1 Introduction

The concept big data has evolved due to an outburst of data from various sources like data centers, cloud, internet, Internet of things (IoT), mobile, sensors and other spheres [1]. Data have entered into every industry, all business operations and currently thought about a major factor in production [2]. Big data is broad and encompasses new technology developments and trends [3]. It represents the enormous volume of structured, semi-structured and unstructured data and usually in terms

K. N. Singh (✉) · J. K. Mantri
Department of Computer Application, North Odisha University, Baripada 757003, India
e-mail: kamakhya.vphcu@gmail.com

J. K. Mantri
e-mail: jkmantri@gmail.com

R. K. Behera
Kalinga Institute of Industrial Technology, Deemed to be University, Bhubaneswar 751024, India
e-mail: rajat_behera@yahoo.com

A. Abraham et al. (eds.), *Emerging Technologies in Data Mining and Information Security*, Advances in Intelligent Systems and Computing 813,
https://doi.org/10.1007/978-981-13-1498-8_30

of petabytes (PB) or exabytes (EB). The data are collected at an unparalleled scale which creates difficulty in making intelligent decisions. For instance, in the process of data acquirement, when the sourced data require decisions on cleanness, i.e., what data to discard and what data to keep, remains a challenging task and how to store the reliable data with the right metadata becomes a major point of concern. Though the decisions can be based on the data itself, greatly data are still not in a structured format. Blog content, Tweeter, and Instagram feeds are imperceptibly structured pieces of text while machine-generated data, such as satellite images, photographs, and videos are structured well for storage and visualization but not for semantic content and search. Transforming such content into a structured format for data analysis tends to be a problem. Undoubtedly, big data has the potential to help the industries in improving their operations and to make intelligent decisions.

1.1 The Five Vs of Big Data

Every day, 2500 petabytes (PB) of data are created from digital pictures, videos, transaction records, posts from social media websites, intelligent sensors, etc. [4]. Thus, big data is described as massive and complex data sets which are unfeasible to manage with traditional software tools, statistical analysis, and machine learning algorithms. Big data can be therefore characterized by the creation of data and its storage, analysis, and retrieval as defined by 5 V [5].

1. Volume: It denotes the enormous quantity generated in no time and determines the value and potential of it under consideration and requires special computational platforms in order to analyze it.
2. Velocity: It refers to the speed at which data is created and processed to meet the challenges and demands that lie in the path of development and growth.
3. Variety: It can be defined as the type of the content of data analysis. Big data is not just the acquisition of strings, dates, and numbers. It is also the data collected from various sources like sensors, audio, video, structured, semi-structured, and unstructured texts.
4. Veracity: It is added by some organizations which focus on the quality of the variability in the captured data. It refers to the trustworthiness of the data and the reputation of the data sources.
5. Value: It refers to the significance of the data being collated for analysis. The proposition of the value is easy to access and produces various quality analytics like descriptive, diagnosis, and prescriptive to produce insightful action in time-bound manner.

Additionally, two more Vs which represent visualization and variability (i.e. constantly changing data) are commonly used to make it to 7Vs but it fails to address additional requirements such as usability and privacy which are equally important.

1.2 Big Data Ecosystem Characteristics

A big data ecosystem must perform vigorously and be resource-efficient. Following are the desired characteristics of the big data ecosystem.

- Robustness and fault tolerance: Robustness is the ability of the system to cope with the error and during the execution and also to cope with erroneous input. Systems need to work correctly and efficiently despite the machine failures. Fault tolerance is the ability of the system to continue operating correctly in the event of the failure of some of its components. The system must be robust enough to handle machine failures and human errors. The systems must be human fault tolerance [6].
- Low-latency reads and updates: It is the measurement of delay time or waiting time experienced by a system. As far as possible, the big data system has to deliver low read time and low update time [7].
- Scalability: It is the ability of a system to manage a growing amount of work or its potential to be enlarged to accommodate the growth. The big data system has to promise for the highly scalable, i.e., in the event of increasing data and load, computing resources should be plugged-in easily.
- Generalization: The big data system to support a wide spectrum of applications with the operational functions of all dataset [6].
- Extensibility: When needed, the big data system provision to add functionalities with minimized cost.
- Ad hoc queries: Big data system should facilitate for ad hoc queries. As the need arises, the ad hoc queries can be created to obtain required information.
- Minimal Maintenance: Maintenance is defined as the work required in keeping the system runs smoothly. Big data system with modest complexity should be prioritized [6], i.e., the maintenance of the system should be kept as minimal as possible.
- Debuggability: Debuggability is defined as the capability of being easily debugged. When required, a big data ecosystem must present the necessary granular information to debug [6] and also facilitate for the required extent to which something can be debugged.

2 Big Data Ecosystem Architecture

The architecture presented below is representing the evolution of big data architecture.

1. Lambda (λ) architecture: In the earlier days, big data systems were constructed to handle three Vs of big data, namely volume, velocity, and variety to discover insights and make timely better business decisions. Nathan Marz coined lambda architecture (LA) to describe fault-tolerant (both against hardware failures and human mistakes), scalable, and generic data processing architecture. LA aims to

Table 1 λ architecture open source technology stack

Area	Technology stack
Data ingestion	Apache kafka, apache flume and apache samza
Batch layer	Apache hadoop, apache MapReduce, apache spark and apache pig
Batch views	Apache HBase, ElephantDB, and apache impala
Speed layer	Apache storm, apache spark streaming
Real-time view	Apache cassandra, apache HBase
Manual merge	Apache impala
Query	Apache hive, apache pig, apache spark SQL and apache impala

satisfy the needs for a strong, robust, and healthy system by serving a wide range of workloads and use cases with low-latency reads and updates. LA also aims that resulting system should be linearly scalable, and should scale out rather than up [8]. The architecture uses a combination of batch and real-time processing paradigm in parallel and runs on a real-time computational system [9]. λ has three layers namely:

Batch Layer: The batch layer is aimed to serve twofold purpose. The first purpose is to store the constantly growing immutable dataset into data sink and the second is to pre-compute batch views from the in-housed dataset. Computing the views is an ongoing operation, i.e., when new data arrives, it will be combined into the earlier existing views. These views may be computed from the entire dataset and therefore this layer is not expected to update the views frequently. Depending on the size of the in-housed dataset and cluster configuration, pre-computation could take longer time [10]. Batch layer produces a set of flat files containing the pre-computed views.

Speed Layer: While the batch layer continuously recompute the batch views, the speed layer uses an incremental approach whereby the real-time views are incremented as and when new data is sourced [10], i.e., it manages only the recent data and computes the real-time views.

Serving Layer: Performs indexing and exposes views for querying.

Incoming data are transmitted to both batch and speed layers for processing. The batch layer manages the immutable append-only raw data and then pre-computes the batch views [9]. The speed layer deals with recent data and computes the real-time views. At the other end, queries are answered by assimilation of both batch and real-time views. Both layers execute the same processing logic and output results in a service layer. Batch views are batch write and random read wherein real-time views are random write and random read [8]. Queries from back-end systems are executed based on the data in the service layer, reconciling the results produced by the batch and real-time views. The three-layer architecture is outlined in Fig. 1.

Open source technology stacks for λ architecture are presented in Table 1.

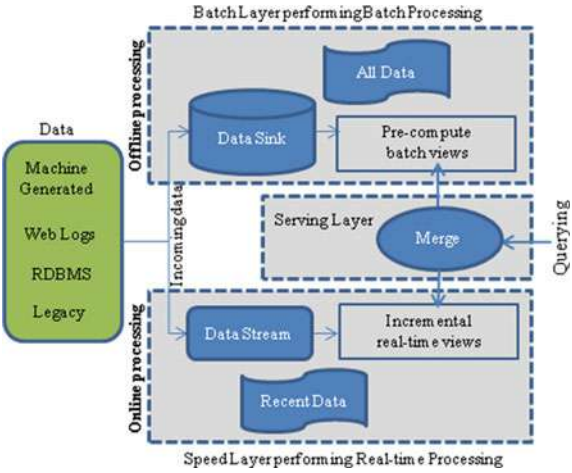

Fig. 1 Three layers of λ architecture, namely batch, speed, and serving layer

Error rectification of λ architecture is performed by allowing the views to be recomputed [11]. If error rectification is time consuming, the solution is to revert to the non-corrupted previous version of the data. This leads to a human fault-tolerant system where toxic data can be completely removed and recomputation can be done easily. The disadvantage of λ architecture is its complexity and its limiting influence. The batch and streaming processing pipeline requires a different codebase that must be properly versioned and kept in sync so that processed data produces the same result from both paths [12]. Keeping in sync the two complex distributed processing pipeline is quite maintenance and implementation, it would bring the same benefits and handle the problem. So the essence is to develop λ architectures using a "unified" framework which makes the same codebase available to both the speed and batch layer and combines the results of both layers transparently, which leads to the development of unified λ architecture.

2. Unified λ architecture: It combines both batch and real-time pipeline, which runs concurrently and the results merged automatically [13]. From processing paradigms, the architecture integrates batch and real-time processing pipeline by offering a single API. The architecture is outlined in Fig. 2.
   With a unified framework, there would be only one codebase to maintain. Open source technology stacks for unified λ architecture are the replica of λ architecture except the "Auto Merge" area. Spring "XD", and Summingbird which are the commonly used open source technology tool.

3. Kappa architecture: It is the simplification of λ architecture [14]. Kappa architecture is similar to λ architecture with the removal of batch processing paradigm. In summer 2014, Jay Kreps posted an article addressing pitfalls associated with λ architecture [15]. Kappa architecture is avoiding maintaining two separate codebases of batch layer and speed layer. It is handling real-time data processing and continuous data reprocessing using a single stream processing computation

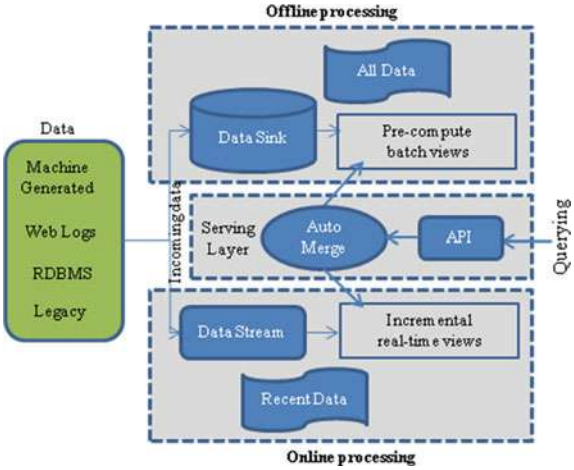

Fig. 2 Unified λ architecture

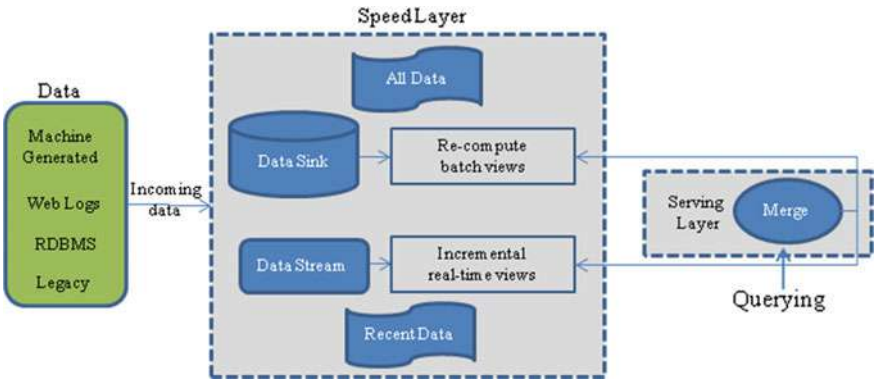

Fig. 3 Kappa architecture

model. Hence, it consists of two layers, namely speed and serving layer. The speed processing layer runs the stream/online processing jobs. Usually, a single online processing job is run to enable real-time data processing. Data reprocessing is done when some code of the online processing job needs to be tweaked. This is accomplished by running another changed online processing job and replaying all previous housed data. Finally, the serving layer is used for querying [15]. The architecture is outlined in Fig. 3.

Open source technology stacks for Kappa architecture are presented in Table 2.

4. Microservices architecture: It divides big data system into many undersized services called microservices that can run independently. This allows every service to run its own process and communicate in a self-ruling way without having

Table 2 Kappa architecture open source technology stack

Area	Technology stack
Data ingestion	Similar to λ architecture open source technology stack
Batch views	Apache HBase, ElephantDB
Speed layer	Apache storm, apache spark streaming
Real-time view	Apache cassandra
Queries	Apache hive, apache pig and apache spark SQL

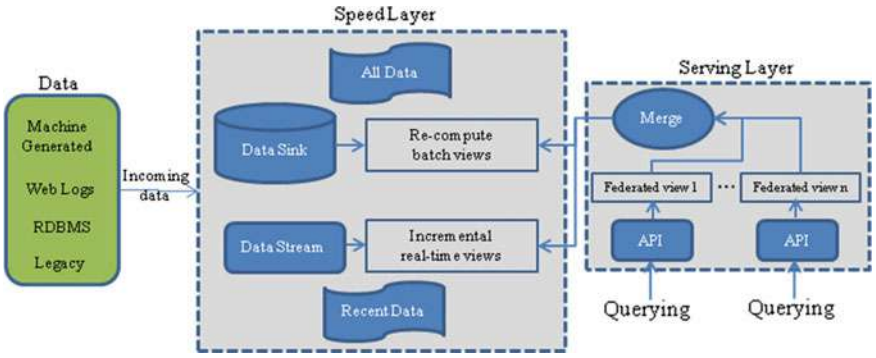

Fig. 4 Microservices architecture

to depend on other services or the application as a whole [16]. Martin Fowler defines that the services must adhere to the common architectural principles, including single responsibility, separation of concerns, do not repeat yourself (DRY), composability, encapsulation, loose coupling, use of consistent, standardized interfaces, etc. [17]. The architecture is outlined in Fig. 4.

Open source technology stacks for microservice architecture are identical to the λ architecture except for real-time view and federated view. Apache Cassandra is the commonly used open source technology tool for real-time view and Apache Phoenix is the commonly used open source technology tool for federated view.

5. Mu architecture: It ingests data into both batch and streaming process, but not for reliability viewpoint, because some work is better done in batches and some are in streaming paradigm [18]. The architecture is outlined in Fig. 5.

Open source technology stacks for Mu architecture remains same as Kappa architecture.

6. Zeta architecture: It is built on pluggable components and all together, it produces a holistic architecture [19]. Zeta is characterized by seven components, namely:

   - Distributed File System (DFS): It is the common data location for all needs and is reliable and scalable.

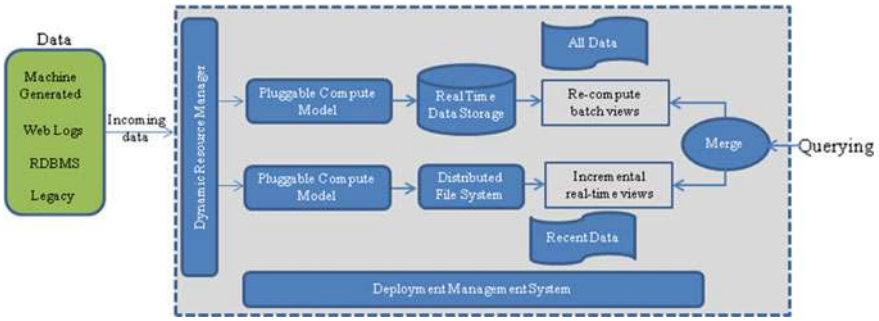

Fig. 5 Mu architecture

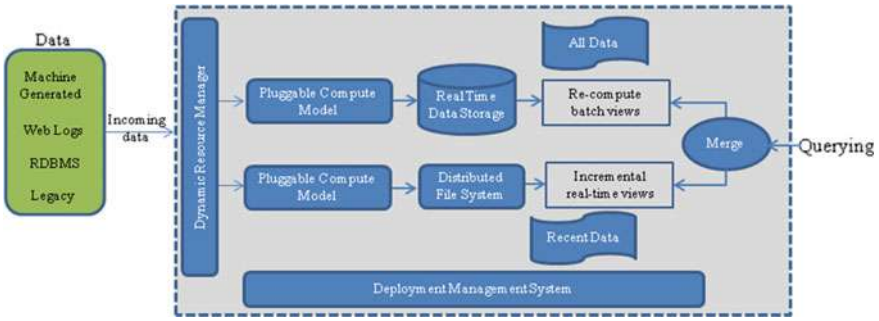

Fig. 6 Zeta architecture

- Real-time Data Storage: It is based on real-time distributed technologies, especially NoSQL solutions and is meant for delivering user supplied responses promptly and quickly.
- Enterprise Applications (EA): EA focuses to comprehend all business goals of the system. The examples of this layer are web servers or business applications.
- Solution Architecture (SA): SA spotlight is on a specific business problem. Unlike EA, it concerns a more specific problem. Different solutions can be combined to construct the solution for the more global problem.
- Pluggable Compute Model (PCM): It implements all analytic computations and are pluggable in nature as it has to cater to different needs.
- Dynamic Global Resource Management (DGRM): It allows dynamic allocation of resources that enables business to easily accommodate for priority tasks.
- Deployment/Container Management System: This guarantees a single, standardized method of deployment and implies that deployed resources do not concern about any environment changing, i.e., deployment in the local environment is identical with prod environment.

The architecture is outlined in Fig. 6.

Table 3 Zeta architecture open source technology stack

Area	Technology stack
Data ingestion	Similar to λ architecture open source technology stack
DGRM	Apache mesos and apache YARN
PCM	Apache spark and apache drill
Real-time data storage	Apache HBase and couchbase
DFS	HDFS
Batch view	Similar to kappa architecture open source technology stack
Real-time view	Apache cassandra
Queries	Similar to kappa architecture open source technology stack

Open source technology stacks for Zeta architecture are presented in Table 3.

7. IoT architecture (IoT-a): The Internet of Things (IoT) is an ecosystem of connected devices and possibly human, accessible through Internet. The IP address is assigned to the devices and it collates, and transfers data over the Internet without manual and human intervention. Examples of such devices are RFID sensors and sophisticated devices like smartphones. Data from such IoT devices transmitted to big data ecosystem to produce a continuous stream of information. In simplicity, IoT devices generate the data and are delivered to big data ecosystem to generate insight in time-bound manner [20]. The big data ecosystem uses scaling-out approach on commodity hardware to overcome the challenges posed by such devices. IoT-a is composed of three primary building blocks namely:

   - Ad hoc queries: Message Queue/Stream Processing (MQ/SP): It receives data from upstream system and depending on the business mandate, and performs buffering, filtering and complex online operations.
   - Database: It receives data from MQ/SP and provides structured and low-latency access to the data points. Typically, the database is a NoSQL solution with auto-shading and horizontal scale-out properties. The database output is of interactive nature, with an interface provided either through a store-specific API or through the standard interface SQL [20].
   - Distributed File System (DFS): In general, it receives data from either the DB and performs batch jobs over the entire dataset. If required, it can also receive data directly from MQ/SP block. This might include merging data from IoT devices with other data sources [20].

   IoT-a architecture is outlined in Fig. 7.
   Open source technology stacks for IoT-a architecture are presented in Table 4.

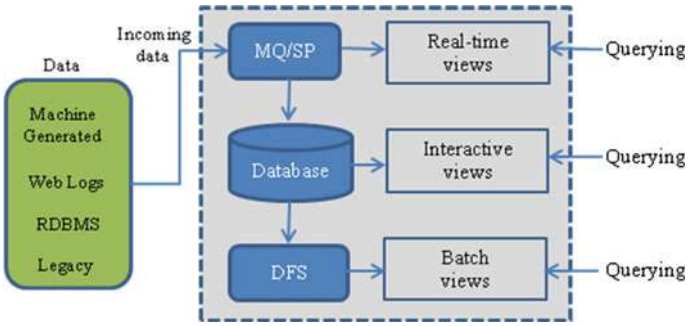

Fig. 7 IoT-a architecture

Table 4 IoT-a architecture open source technology stack

Area	Technology stack
Data Ingestion	Similar to λ architecture open source technology stack
Real-time views	Apache cassandra
Interactive processing	Apache spark
Interactive views	Apache drill
Batch processing	Apache mahout
Batch views	Apache hive
Queries	Apache hive, apache cassandra and apache drill

3 Discussion

This paper briefly reviews big data ecosystem architecture to the best of the knowledge, for discussions and usages in research, academia, and industry. The information presented discusses some research papers in the literature and a bunch of systems, but when it comes to the discussion of a small fraction of the existing big data technology and architecture, there are many different attributes that carry equal importance, weight and a rationale for comparison.

4 Conclusion

A theoretical study or a survey of various tools, libraries, languages, file systems, resource managers, schedulers, search engines, SQL and NoSQL frameworks, operational and monitoring frameworks had been highlighted in order to provide the researcher with the information for understanding big data ecosystem which are on a rapid growth raising concerns in terms of business intelligence and scalable management that includes fault tolerance and optimal performance. In this paper, big data

architecture has been discussed though not in an elaborate manner, but hope this paper will serve as a helpful introduction to readers interested in big data technologies.

References

1. Borodo, S.M., Shamsuddin, S.M., Hasan, S.: Big data platforms and techniques. **17**(1), 191–200 (2016)
2. James, M., Michael, C., Brad, B., Jacques, B., Richard, D., Charles, R.: Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Glob Inst. (2011)
3. Emerging technologies for Big Data—TechRepublic (2012). http://www.techrepublic.com/blog/big-data-analytics/10-emerging-technologies-for-big-data/
4. Every Day Big Data Statistics—2.5 Quintillion Bytes of Data Created Daily. http://www.vcloudnews.com/every-day-big-data-statistics-2-5-quintillion-bytes-of-data-created-daily/ (2015)
5. Big Data: The 5 Vs Everyone Must Know. https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know (2014)
6. Notes from Marz' Big Data—principles and best practices of scalable real-time data systems—chapter 1. https://markobigdata.com/2017/01/08/notes-from-marz-big-data-principles-and-best-practices-of-scalable-real-time-data-systems-chapter-1/ (2017)
7. The Secrets of Building Realtime Big Data Systems. https://www.slideshare.net/nathanmarz/the-secrets-of-building-realtime-big-data-systems/15-2_Low_latency_reads_and (2011)
8. Lambda architecture. http://lambda-architecture.net/ (2017)
9. Jay, K.: Questioning the lambda architecture. www.radar.oreilly.com 2014
10. The Lambda architecture: principles for architecting realtime Big Data systems. http://jameskinley.tumblr.com/post/37398560534/the-lambda-architecture-principles-for
11. Big Data Using Lambda Architecture. http://www.talentica.com/pdf/Big-Data-Using-Lambda-Architecture.pdf (2015)
12. Wikipedia lambda architecture. https://en.wikipedia.org/wiki/Lambda_architecture
13. Lambda Architecture for Big Data by Tony Siciliani. https://dzone.com/articles/lambda-architecture-big-data (2015)
14. Kappa architecture. http://milinda.pathirage.org/kappa-architecture.com/
15. Data processing architectures—Lambda and Kappa. https://www.ericsson.com/research-blog/data-processing-architectures-lambda-and-kappa/ (2015)
16. Microservices Architecture: An Introduction to Microservices. http://www.bmc.com/blogs/microservices-architecture-introduction-microservices/ (2017)
17. Data Integration Design Patterns With Microservices by Mike Davison. https://blogs.technet.microsoft.com/cansql/2016/12/05/data-integration-design-patterns-with-microservices/ (2016)
18. Real Time Big Data #TD3PI, 2015, http://jtonedm.com/2015/06/04/real-time-big-data-td3pi/
19. Zeta architecture. http://www.waitingforcode.com/general-big-data/zeta-architecture/read (2017)
20. Iot-a: the ınternet of thıngs archıtecture. http://iot-a.info