HBase Storage Architecture

Subjects: Computer Science, Information Systems

Contributor: Muhammad Umair Hassan

HBase is the top option for storing huge data. HBase has been selected for several purposes, including its scalability, efficiency, strong consistency support, and the capacity to support a broad range of data models.

HBase big data storage architecture

1. Introduction

One of the leading technical problems confronted by today's companies is to ensure that a massive amount of data is stored, manipulated, and recovered efficiently. Online services and social media are the prime cause of data creation nowadays. Facebook (https://www.facebook.com/) users generate 4 million likes every minute and upload 450 billion photos and videos per day [1]. Together, other social media apps, the internet of things (IoT), and geographic, vector space, electric power, wireless sensor, and power grids produce an immense amount of data, exceeding the scale of petabytes daily [1][2]. These data may include valuable information that current technologies sometimes fail to investigate properly. Some existing traditional systems extract those parts of data that the system can store and process [3][4]. Moreover, all the remaining data are wasted—most of the data stored in an unstructured manner using different languages and tools are not compatible.

Big data evolution is disclosing a severe systemic problem. A systemic problem is when the amount of data is increasing day by day, but they are not stored and processed efficiently. The reason is that the standard tools and techniques are not designed to manage and handle big data's complexity [5][6]. Conventional tools and techniques are unable to handle these complexities because data are growing in a huge volume and in terms of variety and substantial value, and includes different data types [7]. It is necessary to handle such a huge amount of data and its significant problems efficiently. It is undoubtedly a crucial challenge for a database designer to design a system according to big data's immense problems. Inevitably, designing a scheme according to big data issues is a crucial task for database designers.

Large companies in the transportation, weather forecasting, IoT and power sectors, and social websites, e.g., Facebook, Instagram (https://www.instagram.com/), Yahoo (https://www.yahoo.com/), etc., produce petabytes of data per day that include different types and formats [8][9]. These data are used for future decision making, and this decision immediately impacts the company's future. These data have a wide range of distinct data formats and types that traditional schemes may not store and process. Many problems are faced while refining a big dataset; the main problem is to guarantee adequate storage, rather than just processing or analyzing.

There are specific tools that developed over time to store and process big data accurately. Hadoop is one of the tools that can process structured, unstructured, and semi-structured colossal datasets. It has two main paradigms —the Hadoop distributed file system and Hadoop storage area—and map-reduce: a powerful processing tool that processes the stored data [8]. HBase is another big data refining tool developed on Hadoop and ran on top of the Hadoop Distributed File System (HDFS). Because the architecture of HDFS is rigid, it cannot change according to the dataset's parameters. The first time Facebook selected HBase to implement its new messaging platform was in November 2010 [10]. In 2010, Apache HBase began as a project by the company powerset out of a need to process the massive amount of natural language search data. Now the Apache HBase is a top-level project.

1.1. Apache HBase

HBase is an open-source, distributed, multi-dimensional, and NoSQL database [10]. It is designed to achieve high throughput and low latency. It provides fast and random read/write functionality on substantial datasets. It provides the bloom filter data structure, which fulfills the requirement of fast and random read-write. The HBase runs on the top of HDFS and provides all capabilities of a large table to Hadoop. HBase stores files on HDFS. It has the capabilities to store and process a billion rows of data at a time. Due to the dynamic feature of HBase, its storage capacity can be increased at any time [11]. HBase leverages Hadoop infrastructure like HDFS and Zookeeper. Zookeeper co-ordinates with HMaster to handle the region server. A region server has a collection of regions. The region server is responsible for the handling, managing, and reading/writing functions for the region.

A region has data of all columns/qualifiers of column families. A HBase table can be divided into many regions where each region server handles a collection of regions. HBase became famous due to its features of fast and random read/write operation. Many companies adopted HBase because the current problem is handling big data with fast processing [11][12], and HBase is a good option.

The HBase framework offers several methods for extracting, manipulating, and storing big data [13]. This tool has progressed in recent years due to its dynamic architecture and promotes data integrity, scalability, fault-tolerance, and easy-to-use methods [13]. All of these variables have contributed to the popularity of Apache Hadoop, both in academia and in businesses. Most of the companies are using Apache HBase to store their dataset. They have changed the storage architecture according to the parameters of datasets.

1.2. Apache HBase Data Model

HBase is a NoSQL and column-oriented database. While it looks like a relational database that includes rows and columns, HBase is not a relational database. It is a column-oriented database, while the relational databases are row-oriented. Both databases store data differently on the hard disk [14]. Figure 1 is the illustration of the HBase table architecture while the components [13][14] of the HBase table are described below:

- **Table:** HBase tables are column-oriented, i.e., data are stored in column format.
- **Row Key:** It is the most crucial component of the HBase table. It is used for searching and retrieving data. It increases the speed of the searches.

- **Column Families:** The entire columns related to each other are combined and called column families, as shown in Figure 1.
- **Column Qualifiers:** Each column in the HBase table is known as the Column Qualifier.
- **Cell:** A cell is made up of row key, column family, and column qualifier. Actual data are stored in a cell. There are many versions of the cell.
- **Time Stamp:** A Time Stamp is made up of date and time. Whenever data are stored, they have a unique date and time. The timestamp is stored with the actual data, making it easy to search for a particular version of the data.

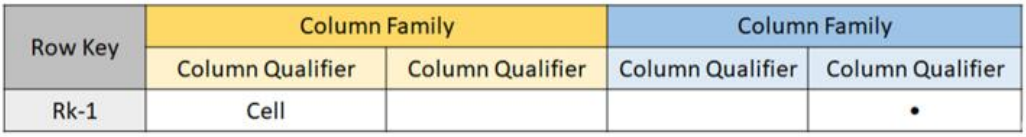

Figure 1. HBase table architecture.

1.3. Contributions

Over the past years, the researchers proposed several methods to store different datasets on HBase in different ways [15][16][17][18]. Most of the research was published worldwide in journals and conferences. Some of the storage technique research studies are dataset specific, and some cover all types of datasets. Many publications make it difficult for companies to find an accurate and efficient technique to store datasets. Thus, there is a need to organize and classify all the proposed storage techniques based on HBase storage architecture. Table 1 below presents the research questions and objectives which we tried to investigate in this work.

Table 1. Research questions and their objectives.

No.	Research Questions	Objectives
RQ1	What are the primary datasets and storage techniques used by the researchers?	The aim is to investigate the primary focus of the researcher in datasets and storage techniques.
RQ2	What are the primary factors of selecting HBase storage architecture in various domains?	The aim is to investigate the domain that is working on HBase storage architecture.

RQ3	Are the proposed approaches application/data-specific or generic?	The aim is to identify whether the proposed techniques are data-specific or generic.

2. Internet of Things (IoT)

The internet of things continually produces a large amount of data. There is a need to collect, process, store, analyze, and use these data. Humans produce big data while machines produce IoT-based data. IoT is the next stage of big data. Different NoSQL databases are used to manage such a tremendous amount of data. HBase is one such NoSQL database. The research proposed a broad range of storage models on HBase to store and process IoT, power, and sensor data. Table 2 shows the main features of selected studies in the IoT domain.

Table 2. Main features of selected studies for IoT domain.

No.	Storage Efficient	Scalable	Novel Technique	Indexing/Row Key	Improved Reading/Writing	Main Features	Experimental	Implementation
[P1]	a	a	a	Elastic based secondary	a	Managed heterogeneous remote sensing data.	a	a
[P2]	a	a	a	Implement row key in lexicographic order	a	Solved the issue of hotspot data scatter and high concurrent transactions.	Not mentioned (NM)	a
[P3]	a	a		r	r	Quickly stored massive sensor data.	a	a
[P4]	a	a	a	Event type and event	a	Novel virtual column family	a	a

				time as the row key			improved the join operation.		
[P5]	a	a	a	Single row key, complexed row key	a		Overcame the single point of failure. Reliable data storage. Met the real-time computing requirements.	a	a
[P6]	a	a	a	r	a		Redesigned the HBase table, solved the problem of the hotspot.	a	a
[P7]	a	a	a	Grid index and the Hibert curve	a		Stored imaged data. Effectively improved the data writing and query speed, and showed good scalability.	r	a
[P25]	a	a	r	a	a		Stored the traffic data.	a	a
[P30]	a	a	a	Hash prefix with a	a		Improved the HBase table	r	a

				timestamp		for time series data.			
[P31]	a	a	r	a	a	Improved query performance of time series data.	a	a	

[P1] proposed a distributed data management architecture. This architecture handles remote sensing data in a heterogeneous environment. There are different file formats that HBase supports. Remote sensing data have a different characteristic, based on these characteristics; data are generated in different data types and formats. The authors selected a NetCDF file format to store the remote sensing in the proposed architecture. First, they converted all types of data in a NetCDF format then loaded it into the proposed architecture of HBase. An elastic search-based secondary index was built on HBase that provided the ability to search for data based on metadata content. This architecture is mainly divided into three components: the data integration component, data organization and storage component, and data index and search component. The first component extracts, transforms and loads remote sensing data automatically. The second component organizes the data remote sensing data in HBase, while the third component indexes the data and prepares it for search queries. [P2] introduced a lexicographical order for solving hotspot data center issues and high concurrent data transactions. A two-layer distributed storage structure was designed that includes two systems such as (1) a distributed database to store metadata and (2) MySQL to store sensor data. However, they have not mentioned the experimental evaluations in their work. [P3] worked on IoT-based sensors data. They proposed the HBase architecture for wireless sensors' data to design a real-time storage model. The dynamics of HBase storage architecture are updated and optimized in their work.

[P4] proposed a versatile Event-Driven data model on HBase for multi typed data of the power grid. Event-driven models defined each row as a unique record in the power grid and stored each single row in the HBase table structure. A novel virtual column family is used to resolve the compatibility issue between various data formats. HBase does not support the join operation. The join operation is integrated with the proposed model that improved the reading performance. A unique row key is designed to improve query performance. The row key is a combination of a timestamp and a novel virtual column. Because all the data are stored in a single column, there is no need to use the join operation.

[P5] proposed a storage system on HBase for handling the IoT. IoT based system architecture consists of three primary levels: the gateway layer, HBase managing layer, and the last one is global managing layer. The proposed system has two layers. One is the distributed database HBase to store the metadata of sensor data, and the second is MySQL to store the sensor data.

[P6] proposed a novel HBase data storage for wireless sensor networks. The storage model is designed for efficient real-time data processing. The main reason behind this storage model is that a real-time processing system can only fulfill the need for different users' queries one at a time. Ethnics primitives were used to integrate heterogeneous datasets stored in the HBase database. A real-time flow of data, stored in the cluster database, is used to satisfy the users' multiple needs that required data storage performance. Besides, text data of the historical stream is also migrated in a proposed storage model. A line keyword is used as a unique row key to store the data in HBase. In the last section, a discussion is conducted on when the storage space is insufficient.

The power grid usually generates enormous amounts of data from multi-sources with hundreds of complicated data types. Different complex data types are needed to store accurate results efficiently.

Sensor technologies generate a tremendous amount of data. How to efficiently store and process this sensor data is a hot research topic. [P6] proposed a data processing middleware to tackle the issue related to sensor data. This middleware was named massive sensor data processing (MSDB). The hotspot is a big issue while managing big data; the authors used a pre-splitting technique to solve the hotspot issue. To store sensor data, they redesigned the HBase table according to the characteristics of sensor data.

[P7] proposed an improved distributed storage and query for remote sensing data. The proposed system works in three steps. First, the storage model, according to the characteristics of remote sensing data, redesigned the HBase table. Second, a grid index and Hibert curve were combined to establish the index for the image data. The third is the last step in which MapReduce parallel processing was used to read and write remote sensing data.

The work [P25] used the MapReduce framework component of Hadoop for processing and statistical analysis of urban traffic data. The monitoring system generates a massive amount of time series monitoring data in real-time. It is an issue to store and process data efficiently. To tackle the issue of time series data in the cloud computing environment, [P30] proposed a scheme on Hadoop. This schema used HBase as a database to store the time-series data and MapReduce for improving the processing efficiency. The authors changed the HBase table structure according to the need for traffic data and their requirements. Furthermore, the authors designed a framework for storing or processing the traffic data. This program has two main modules: the HBase storage module and the MapReduce processing module.

The concept of the Internet-of-vehicles is derived from the Internet of Things. Primarily, the IoV's problem is to collect and process enormous amounts of massive information. [P5] originally designed a distributed HBase IoV data storage system. The proposed system has three main layers: the application layer, which provides the interface; the data processing layer, which processes the data according to the design format of HBase; and the third layer is the data storage layer, which stores the actual data. Moreover, three types of row keys are designed, such as single row key, complex row key, and secondary row key. This system ensured to solve the main three issues of IoV data. First, it can overcome a single point of failure; secondly, it ensures the real-time computing requirement; third, it ensures reliable data storage. In the same way, [P31] proposed a massive traffic data

querying solution using HBase. The authors designed a schema for the HBase table, and, in particular, a row key structure was designed to store data effectively.

3. GeoScience

A traditional database is a good option for transactional operations but not suitable for large scale data analysis and processing. For large scale data analysis and processing, like geospatial data, an efficient storage model is required. [P8] proposed a storage model based on HBase and MapReduce to tackle geospatial data storage and processing. There are two important parts of the proposed storage model. First, to improve the efficiency of I/O, an advanced method for geospatial data was proposed on MapReduce. Secondly, the authors redesigned the table structure of HBase according to the requirement of Geospatial data. A unique row key was designed based on the Geohash string to access the stored geospatial data efficiently. Table 3 shows that the main feature of selected studies of the GeoScience domain.

Table 3**.** Main features of selected-studies of GeoScience domain.

No.	Storage Efficient	Scalable	Novel Technique	Indexing/Row Key	Improved Reading/Writing	Main Features	Experimental	Implementation
[P8]	a	a	a	Geo Hash	a	Improved I/O efficiency.	a	a
[P9]	a	a	a	a	a	Improved query processing time.	a	r
[P10]	a	a	a	Z curve or Geometry object identifier	a	Improved query performance.	a	r
[P11]	a	a	a	a	a	Improved write operation, and a log layer was	a	a

						used instead of HDFS.			
[P12]	a	a	a	Combination of B-order value and keyword	a	Improved query efficiency and reduced the time consumption of spatial queries.	a	a	
[P13]	a	a	a	Index structure using quadtree over HBase	a	Multi-dimensional data storage. Thousands of location updates per second	r	a	
[P26]	a	r	a	Geohash	a	Decreased the spatial query time.	a	a	

[P9] stores spatial-temporal information in HBase and mainly focuses on the classification and abstraction of it. The HBase storage model is built by constructing a spatial-temporal data table and designing the meta semantic object (MSO). MSO is an abstraction of geographic space. To support spatial queries for geographical databases, Hong Van Le et al. [P26] proposed a novel partitioning method.

The spatial vector data of the geographic information system (GIS) is increasing dramatically. Nowadays, storing and processing spatial vector data is a key step for GIS. [P10] proposed a NoSQL storage schema for spatial vector data on HBase. Two storage schemas were designed: one, Z schema, is a row key-based Z curve, and the second, ID schema, is a row key based on geometry object schema. In the Z curve, the schema row key is based on the Z curve. First, it divides the regions into two dimensions, then fills the gap by Z curve. Now, this Z curve is used as a row key. In the second schema, the polygon ID is used as a row key. In this schema, each column family has only two column qualifiers. The first column qualifier is used to store the spatial index, and the other column is

used to store the co-ordinate information of the polygon. A series of experiments were performed to evaluate the proposed schema design.

In the advancement of cloud computing, its dynamic nature, powerful storage capacity, and other features create incentives to migrate the traditional system to cloud computing. This leads to the opening of a new research gate for researchers to solve the deployment issues from a traditional system to the cloud computing environment. [P11] designed a geographical information query system based on HBase. The query system made users retrieve the information of sea wind and satellite images by the graphical interface. This system has three layers: the operation and view layer, data processing layer, and storage layer. Data processing is improved by designing a row key as combined by the B-order value and keyword. The system was implemented on the National Geographic Public Welfare project. The scale of maritime traffic data is rapidly increasing. The AIS has emerged as a safe means of navigation for ships. [P12] proposed AIS Data management based on HBase and Spark. This distributed system provides efficient storage and near real-time queries to massive AIS data.

AIS HBase and spark store all data of a single ship in one region of HBase. This work is done with the help of HMaster. This way, it improves query performance and data access time. A secondary index was constructed on spatial-temporal attributes of massive AIS data, and co-location was created between HBase regions and Spark RDD to ensure efficient spatial-temporal query. The AISHS consists of main three-components. First, the data storage module; this module stores massive AIS data on HBase. It is also responsible for storing single ship data in a single region of the region server of HBase. The second component is the secondary index module; it creates the secondary index based on AIS's spatial-temporal attributes. The third module is a query processing module responsible for ensuring efficient query processing with a low cost and a high performance.

[P13] presented a multi-dimensional data storage model for location-based applications on HBase. The authors introduced a new index structured on HBase using the quadtree. The multi-dimensional data were based on the location that was inserted first and nearest neighbor queries were performed. In the end, response time was compared with the traditional database management systems.

References

1. Coughlin, 2019. Available online: https://www.seagate.com/in/en/our-story/data-age-2025/ (accessed on 13 February 2020).
2. Morris, 2019. Available online: https://www.business2community.com/big-data/19-data-and-analytics-predictions-through-2025-02178668 (accessed on 21 March 2020).
3. Zheng, ; Fu, Y. Research on vector spatial data storage schema based on Hadoop platform. Int. J. Database Theory Appl. 2013, 6, 85–94.

4. Um, H.; Lee, S.; Kim, T.H.; Jeong, C.H.; Song, S.K.; Jung, H. Distributed RDF store for efficient searching billions of triples based on Hadoop. J. Supercomput. 2016, 72, 1825–1840.
5. Zhang, ; Wu, G.; Hu, X.; Wu, X. A distributed cache for hadoop distributed file system in real-time cloud services. In Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, Beijing, China, 20–23 September 2012; pp. 12–21.
6. Li, ; Zhu, Z.; Chen, G. A scalable and high-efficiency discovery service using a new storage. In Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference, Kyoto, Japan, 22–26 July 2013; pp. 754–759.
7. Kim, ; Choi, J.; Yoon, J. Development of the big data management system on national virtual power plant. In Proceedings of the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, Poland, 4–6 November 2015; pp. 100–107.
8. Rathore, M.; Son, H.; Ahmad, A.; Paul, A.; Jeon, G. Real-time big data stream processing using GPU with spark over hadoop ecosystem. Int. J. Parallel Program. 2018, 46, 630–646.
9. Smith, K. Available online: https://www.brandwatch.com/blog/facebook-statistics/ (accessed on 20 December 2019).
10. George, HBase: The Definitive Guide: Random Access to Your Planet-Size Data; O'Reilly Media, Inc.: Sebastopol, California, USA, 2011.
11. Huang, ; Wang, L.; Yan, J.; Deng, Z.; Wang, S.; Ma, Y. Towards Building a Distributed Data Management Architecture to Integrate Multi-Sources Remote Sensing Big Data. In Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, UK, 28–30 June 2018; pp. 83–90.
12. Wang, Y.; Li, C.; Li, M.; Liu, Z. HBase storage schemas for massive spatial vector data. Clust. Comput. 2017, 20, 3657–3666.
13. Taylor, C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinform. 2010, 11, S1.
14. Sinha, HBase Tutorial: HBase Introduction and FaceBook Case Study. 2018. Available online: https://www.edureka.co/blog/hbase-tutorial (accessed on 28 September 2020).
15. Chen, ; Chen, S.; Feng, X. A design of distributed storage and processing system for internet of vehicles. In 2016 8th International Conference on Wireless Communications & Signal Processing (WCSP); IEEE: Yangzhou, China, 13-15 October 2016. pp. 1–5.
16. Liu, ; Huang, R.; Huang, T.; Yan, Y. MSDB: A massive sensor data processing middleware for HBase. In Proceedings of the 2017 IEEE Second International Conference on Data Science in

Cyberspace (DSC), Shenzhen, China, 26–29 June 2017; pp. 450–456.

17. Jing, ; Tian, D. An improved distributed storage and query for remote sensing data. Procedia Comput. Sci. 2018, 129, 238–247.
18. Gao, ; Yue, P.; Wu, Z.; Zhang, M. Geospatial data storage based on HBase and MapReduce. In Proceedings of the 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, USA, 7–10 August 2017; pp. 1–4.
19. Liu, ; Huang, R.; Huang, T.; Yan, Y. MSDB: A massive sensor data processing middleware for HBase. In Proceedings of the 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), Shenzhen, China, 26–29 June 2017; pp. 450–456.
20. Jing, ; Tian, D. An improved distributed storage and query for remote sensing data. Procedia Comput. Sci. 2018, 129, 238–247.
21. Gao, ; Yue, P.; Wu, Z.; Zhang, M. Geospatial data storage based on HBase and MapReduce. In Proceedings of the 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, USA, 7–10 August 2017; pp. 1–4.
22. Wang, Y.; Li, C.; Li, M.; Liu, Z. HBase storage schemas for massive spatial vector data. Clust. Comput. 2017, 20, 3657–3666.