International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Efficient and Secure Transformation of Log Files using ADLS GEN2 Architecture

Sheik Syed Sulaiman M

Abstract: ***Web server log encompasses a document where all the activities transpiring on a web server are carefully recorded. It captures each request made to the server like request type, IP address of the device, what files where requested, date & time of request, name and location of requested file, file size etc. These files are stored in their raw form for a long time for analysis and taking high quality decisions which causes storage problem and security threat as data present in the log files are in raw format. This paper focuses on providing an economical, secure and high - performance solution for the storage of large amount of raw log files. The proposed system includes Azure Data Lake Storage Gen2 which allows large volumes of data to be stored in their raw form as well as they are subjected to transformation and advanced analysis processes without the need of a structured writing scheme. This paper mainly provides solution that is affordable and more accessible to perform web server log data ingestion, storage and transformation over Data Lake. This paper also proposes the use of Azure Trigger Function that transforms the log files into parquet files which reduces the storage space compared to their original size. A hierarchical data storage model has also been proposed for shared access to data over different layers in the Data Lake architecture, on top of which Data Lifecycle Management rules have been proposed for storage cost efficiency. The aim is to maintain this data in the long term to be used in future advanced analytics processes by cross - referencing with other organizational or external data which could bring important benefits.***

Keywords: Cloud Data Lake**,** Azure Data Lake Storage Gen2 (ADLS GEN2), Data Lake architecture, Web Server log, Azure Trigger Function

1. Introduction

The Internet of Things paradigm involves the development of diverse applications and solutions in various domains such as smart homes, digital surveillance, industrial processes etc. which needs to be constantly monitored and analyzed. Server log files are a raw and unfiltered information related to request made to the server. They're text files stored on your web server. Every time any browser or user - agent like Google, requests any resource pages, images, Java script file from the server, it adds a line in the log. High - scale analysis of these logs brings a great deal of added insight to the different business domain.

In [1] the authors have provided a systematic review of recent literature on the different types of log files that are used in research area. The importance of these log files is proven in all fields of activity. Nowadays, identifying a powerful storage and analysis environment is very imperative.

Web server log files store all events that occur as a result of requests issued by online users accessing information available on a server. Recording these events can lead to the identification of user visiting behavior [2] over certain periods of time, which can lead to a better understanding of customer requests. Companies can then make smart, personalized decisions to improve user experience. Servers keep these records in different log files, namely, web server access log files. By analyzing these log files, it is possible to discover "visiting behaviors", which can lead to major improvements and allow the services offered to users to be perfectly customized.

This paper proposes Data Lake Architecture to resolve the existing problem of storing, processing and analyzing the web server access log files at large scale. Analysis of web server access log files along with the Data Lake design methodology includes the procedure of transforming web server access log files into parquet files using the azure trigger function written in C#. In the current paper a hierarchical structuring model is also included for the organization of web server access log files data in order to serve the Extract Load Transform pipelines specific to the Data Lake architecture. Web servers are accessed by millions of users every day. Users leave behind their visiting behavior by recording their activities online in log files.

Servers keep these records in different files such as access logs, error logs, piped logs, script logs etc. Log files are often automatically deleted from servers due to their constant increase in volume. There are various techniques for analyzing web server access log files in the market [4], [5], the main problem lies in storing them in their raw form in order to allow various analysis processes to be run in the future enabling information to be extracted for different purposes. Transforming these logs into structured data for later storage, using traditional methods involve high storage costs. As a consequence, the explosion of these new data types led to emergence and development of new concepts, technologies and techniques for data management.

Data Lake is a revolutionary concept that has been in the spotlight recently. Data Lake, as the name suggests can be seen as a structure where absolutely all data can be stored in its raw form [6]. Data Lifecycle management techniques are used for cost optimization, data security mechanisms, data redundancy aspects. Several important technical aspects have been also highlighted and should be considered when building a data lake in the cloud. Thus, this paper presents the design and implementation of a Data Lake architecture in cloud for web server access log files in long - term storage.

The proposed system is based on the implementation of a DL architecture in cloud using the Microsoft technology, namely Azure Data Lake Storage Gen2. It is based on Apache Hadoop and Apache YARN. It is a solution which does not require installation of any hardware or software systems on the user's side. It is a cost - effective cloud service where you pay only

for the used storage space. These costs can be optimized as needed through data lifecycle management. An ADLS Gen2 architecture with different levels of data processing is presented.

Most servers generate web server access log files that can be stored in different formats, such as Common Log Format or Combined Log Format. A study on the content of these files has led to the identification of the following categories of data [7], [8], [9] such as requested resource file location, name, and size; request time and date; request method; identity of client device that made the request to the server; client browser information; page from which the client left the site; server response to client request; number of pages visited by client; what search engine was used and search terms used; whether it was a direct access by a user or a redirect from another site, or part of an advertising campaign.

WSAL files are raw data files that are difficult to understand without a proper log parsing tool. Another major problem is storing these files, because they need a lot of storage space. In a short period of time, depending on the frequency with which the servers are accessed, huge numbers of data can be generated and stored on them, which is why they are automatically deleted after a period or after they exceed a certain size [10]. It is estimated that 80% [11] of all companies' data is lost, because organizations cannot keep up with the volume, speed, and variety of data. Inside companies' it is being estimated that 80% of data is unstructured. On average, between 60% and 73% of all this data is never analyzed [12]. This is so - called auxiliary data or dark data, collected from various sources like data from sensors, statistics on processes, social media, monitoring services, log files, raw survey data, audio, and video, etc. [13]. There is a large percentage of lost unstructured data not being analyzed inside a corporation, data from which valuable information, that could lead to multi - level improvements through advanced analysis processes, could be extracted.

The main advantages of the long - term analysis of WSAL files are improved website security and structure; improved web server performance; increased website traffic; elimination of navigation errors on website; high - quality marketing strategies based on customer preferences. Thus, it becomes a necessity for most large companies to be able to store and analyze these log files to ensure reliability, security, and performance, especially for the financial benefits. For small sites, there are already a variety of online or open - source services on the market [14] that can extract various reports based on standard analyzing techniques, but with several limitations, either related to file size or to the type of analyzing processes, which usually return specific reports and statistics. As mentioned above, the major problem lies in storing the log files on long term. One ideal candidate for storing and analyzing large volumes of unstructured data in their raw form is the latest technology: a DL with unlimited storage space with auto scaling at low costs.

2. Proposed System

The examination of log files even after longer periods of time is necessary. Companies do not store these logs as it requires significant amount of storage capacities. Therefore, an optimal solution for archiving these log files for long periods of time may be to use on - premise or cloud Data Lakes.

The main advantages of saving log files in a Data Lake:

- As web server access log files are locally stored on servers for limited periods of time, the use of a Data Lake offers maximum flexibility in storing data for long periods of time at affordable costs
- Control over hierarchically structured data within the DL storage layer can be successfully managed by assigning different access roles to each area;
- Existing web analytics tools rely on Java script codes or cookies, which can cause delays in loading a web page and are more susceptible to various technical problems. If the user has scripts and cookies blocked in their personal browser, then reports from services such as Google Analytics may not be conclusive. Thus, storing log files in a DL offers the possibility to implement dedicated analysis processes to extract customized information and reports without influencing the loading of a web page and without depending on the client browser settings.

Hence, the implementation proposed in this paper based on Data Lake technology in the cloud is a low - cost solution with remarkable performance in terms of storing and analyzing web server access log data to obtain valuable information from web servers.

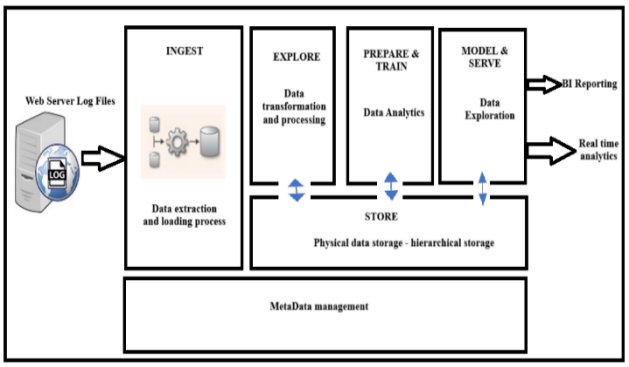

Figure 1: Proposed DL architecture and various Extract Load Transform processes serving Web Server Access Log files.

3. Implementation of Data Lake Architecture

This paper addresses the usage of technique, cloud Data Lake storage of Web Server Log data which is more accessible without the need for hardware, software or engineering resources that are not always affordable and available to everyone and can be expensive as time consuming to implement. Data Lake in the Cloud is a leading technology for managing large data sets with advanced real - time data analytics. With this new technology, the huge potential of previously unanalyzed data is revealed. We present and validate an original project implemented in Azure Cloud for storing Web Server Log files without storage limits at low costs, so that they can then undergo advanced analysis processes. Thus, we propose a solution using Azure Data Lake Servera Gen2 cloud storage, a service provided by Microsoft that allows the customer to create and own a Data Lake where large volumes of data can be stored, transformed, and analyzed.

Azure Data Lake Server Gen2 is a hyper - scalable system which relies on archiving service to keep the cost of storing unstructured Big Data as low as possible. It includes the capabilities like hierarchical directory structure, ideal for big data analytics workloads, optimized cost and performance, storage Tiers, finer grain security model, massive scalability. A strength of Azure Data Lake Server Gen2 concept is that it allows data to be organized in a hierarchy of directories and subdirectories for efficient data access (hierarchical namespace). Until Azure Data Lake Server Gen2, data was stored in flat namespace mode, this hierarchy is new and comes with multiple advantages. As data volumes grow, the hierarchy allows data to be maintained in a more organized way and provides better performance for data analysis tasks.

The proposed system contains a Data Lake architecture for the implementation of different processes within the ELT pipeline, storage of Azure Data Lake Server files, transformation and preparation of data for consumption, once they have been submitted to different analysis processes. As a result, we were able to design and propose a stable, reliable and economical architecture shown in Figure 1.

The proposed architecture consists of 5 different layers. The Store layer is the core level of the entire architecture. Above this layer there are the Explore, Prepare & Train, and Model & Serve layers, which serve different transformation, enrichment, and processing steps that data undergo for valuable information to be extracted with different advanced analysis processes. Over the ingestion layer, fast processes are running to load different types of raw data from various external sources into the Data Lake. There is no data alteration within this layer. Raw data can be ingested in real time or in batch mode. Following ingestion, the data are saved into the Data Lake storage layer to a dedicated raw data area. Thus, the data are stored in their natural form within this area of the Data Lake. In the exploration layer, transformation processes are performed over the stored data, thus the data are cleaned and standardized into proper data types. This is where raw data are being transformed into more structured datasets that are being placed into well - organized directories and subdirectories into a different area in the storage layer.

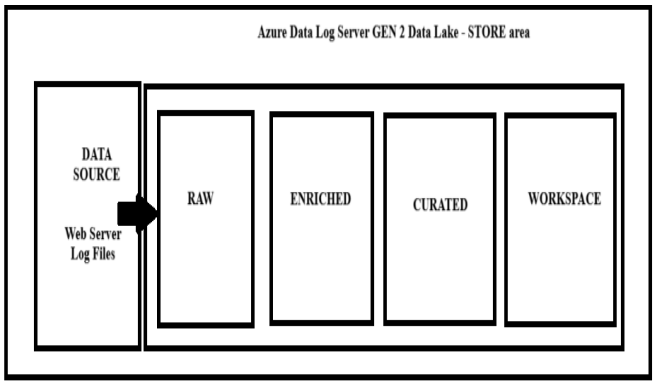

Figure 2: ADLS Gen2 hierarchical structuring of web server log data.

The Prepare & Train layer of the architecture is where previously transformed data are prepared and modelled using advanced analytics engines. A new area within the DL storage layer is dedicated to the data derived out of these processes. At the Model & Serve layer, data are accessed by users who know the data and the needs of an organization. New modelling processes are carried out using appropriate analysis techniques in order to obtain information to be used for final statistics and reports. Each of these layers is served by different processes that can be implemented using either integrated technologies provided by Microsoft, such as Azure Data Factory, Azure Databricks or through ad - hoc developed processes written in different programming languages that can be easily integrated using Azure Logics or Azure Functions.

This research paper focuses mainly on the ingestion layer, the storage layer and the data transformation process, within the Explore layer, leading to standardization and preparation of data for further advanced analysis processes. The data ingestion process in Azure Data Log Server Gen2 consists in uploading the log files holding the raw data into the Data Lake by implementing a batch process on the server that automatically uploads the log files to the Data Lake at regular intervals or using various built - in technologies provided by Microsoft. Once data are saved in their raw form in the cloud Data Lake, they undergo a transformation and standardization process that transforms log files into Parquet files. This process is fully automated with the use of Azure Blob Trigger, which automatically launches into execution a function that transforms log files into Parquet files. Once transformed, data can be subjected to various cleaning and enrichment processes to be prepared for the final analysis process that serves the ultimate interests.

3.1. Hierarchical Data Structuring in the Data Lake

The data that are ingested into the Data Lake must be stored in a well - organized way so that they can be easily managed. The data should not be randomly thrown into the Data Lake Store layer, but should be organized in a structured way. Each area of this structure serves different processing layers. In Figure 2, we show the design of a data structuring model to serve the entire Azure Data Log Server Gen2 architecture throughout the various layers. Thus, into the Store layer, we have the following areas:

The **Raw** area is the area where raw data are loaded directly from the server source, with restricted access. This area provides storage for raw data obtained during the ingestion process. This area can be organized using a hierarchy of directories based on the frequency with which log files are generated. All ingestion processes dedicated to different data sources, in case the data comes from different servers, have write - only access to the directory associated with the data source. In our case we have only one source so we designed under Raw folder only one dedicated directory, labeled weblog.

The **Enriched** area stores data that have undergone a transformation and standardization process and have subsequently been saved in Enriched area as Parquet files. Thus, original raw data remain in the Raw area, which, to minimize storage costs, can be transferred to a cooler or archive tier for long - term archiving. In addition, in the Enriched area, there is a folder named logerrparsing, where text files containing error log lines that could not be successfully transformed are uploaded. This is the area

serving the Explore layer within the proposed DL architecture.

The **Curated** area stores data from the enriched area that have been cleaned to make them ready for exploration. This area serves the processes running on Prep & Train layer in the proposed architecture. Different cleaning methods can be applied to the same dataset to serve different analysis techniques, depending on the purpose. In this area, data undergo a change in the storage hierarchy, which can serve different purposes. In our case study we move from a hierarchy organized according to the date on which the logs were generated to a hierarchy dedicated to different segments of interest, like clients, search engine scans, system errors, etc.

The **Workspace** area is the data exploration area serving the Model & Serve layer of the proposed architecture. The data in this area is organized by project and can be accessed by data engineers, data scientists, and data analysts who understand the data and business needs. Here, the data are structured into projects with dedicated access for different teams. Data organized hierarchically into directories and sub - directories are given different access rights as needed to maintain their integrity as recommended in Figure 2.

3.2. Log Data Transformation Process

After loading the data into the DL, a process of transforming raw, unstructured data into structured data is automatically triggered. Thus, log files, once loaded into the DL, undergo an automatic process of transformation into Parquet files. The transformation process is achieved by developing an Azure Function Trigger written in C# using the Visual Studio Code development environment. Azure Function is a serverless compute service that enables user to build and debug functions locally, deploy and operate at scale in the cloud, and integrate services using triggers and bindings.

A collection of directories and files are automatically generated locally. Before creating a new project, it is important that in VS Code the connection to the Azure account to be already established, so that when the files for the new project are being generated, the Azure storage connection parameters are automatically inherited and therefore the access to the cloud resources will be granted. The function. json file defines the inputs and outputs for the main function that performs the data transformation process, therefore there are defined an input blob Trigger and two output blobs. The main function expects three parameters: one input blob Trigger and two output blobs.

Algorithm: Blob Trigger Function to Transform Log Files into Parquet Files
Input: inputvalue
Output: outputvalue; textoutvalue
Begin main function
regex =<define regular expression>
columns =<define parquet columns>
inputvalue=<read inputvalue bytes>
mytxt_obj =<convert bytes to Unicode object in order to perform line by line reading>
array_log_lines =<initialize array containing valid log lines>
For each log line
Begin
If (check for regex format)
Append valid log lines to array_log_lines else
Append non valid log lines into ASCII text error file
End for
df_log_line=<put array_log_lines into a data frame>
call to_parquet function to convert data frame into parquet file
(Pushing the transformed data to the output blobs)
outputvalue =<set outputvalue with the new parquet file>
textoutvalue =<set textoutvalue with the text file containing the transformation errors>
End main function

The main transformation function automatically runs when a new file is uploaded to DL in the raw/ directory which is being monitored by the trigger. This function performs the transformation of log files into Parquet files. This Algorithm conceptually shows how to transform log files into parquet files.

4. Experimental Results

Once the trigger function is created, it can be run locally and then upload a new log file to the Azure DL in the folder monitored by the trigger. Once the upload to the server is successfully completed, a sequence of messages appears in the VS Code terminal confirming that the trigger has detected a new file that has successfully uploaded to the DL, which triggers the execution of the function that performs the transformation process. The log file underwent a line - by - line read process for a regex check on each line.

The conversion time varies greatly depending on the machine on which the process is performed. In this case, for a 1.0 GB log file, the transformation time was about 76s from the time the trigger found a load at the selected directory level. In the transformation process, a delay time of about 8 ÷ 10 min can be encountered from the moment the upload finishes until the trigger confirms that a successfully uploaded file has been detected, because trigger functions rely on logs to scan for new/changed blobs; this can cause delays, which can be improved, according to [17]. After the transformation process, a 108.39 MiB access - weblog1. parquet Parquet file and a 102.02 KiB erroraccess - weblog1. txt text file was obtained from the 1.0 GB log file. The file was reduced from 1.0 GB to 108.39 MiB, that is, 10.8 % of its initial size. Following the tests carried out in Azure Portal, we obtained in each of the two areas, raw and enriched, the four files corresponding to each area.

The Parquet files containing the data transformed and saved in columnar format were greatly reduced in size compared with the initial size. In the text files, we captured log lines that could not be transformed because they did not respect the constraints imposed by the regex. The nature of the errors should be evaluated, and the regex should be adapted in order to obtain as few error lines as possible. At a first analysis of the lines returned as errors by the regex process, they appear to be lines that, from an analytics point of view, are irrelevant and can easily be ignored. If, however, they are still of interest and must exist in the Parquet files for further analysis, then

the regex in the main function must be adjusted. Thus, in this paper, the results of the Blob Trigger Function and the related results of the DL transformation are presented.

Once the log files from the raw area have been converted to Parquet files, one can then move the log files to a cooler tier for long - term archiving and for significantly reducing storage costs. During the creation phase of a standard Azure account, one can choose between the three different types of tiers in order to minimize the costs applied to data storage, processing, and consumption. Hot tier is optimized for storing data that are accessed or changed frequently. It has the highest storage costs but the lowest access costs.

Data Lake datasets have different lifecycles. For example, at the beginning of the data lifecycle, the data are accessed frequently, but the need for access often decreases drastically as the data age. Some data are accessed frequently while other data remain inactive in the cloud and are rarely accessed. Some datasets expire within days or months of creation, while other datasets are actively read and modified throughout their lifetime.

Azure Data Lake Server Gen2 is a cost - effective storage environment with no upfront costs. Data lifecycle management in the cloud is an important process that can be achieved using Data Lifecycle Management policies. Azure storage offers rules - based management policies that can be used to manage the right types of data access.

We proposed a hierarchical storage model to serve the different levels within the DL architecture. Thus, the raw data area, following the ingestion process, stores the data in its raw form. The enriched data area stores data from the raw area that has undergone an automatic transformation and standardization process, whereby the raw unstructured data is transformed into structured data and stored in parquet files. In this area are also stored in a separate directory called logerrparsing the log lines that could not be transformed according to the required criteria. The curated data area stores preprocessed data ready for consumption according to purpose. On the last level we have the workspace data area with fully transformed and aggregated data that can be accessed by the various work teams who understand the needs of the organization.

Following the ingestion process the data is stored in its raw form without transformation. Since WSAL in its raw form is data that is difficult to analyze and interpret, it was proposed to transform it into parquet files.

To automatically transform data from log files to parquet files, we made an Azure Blob Trigger function written in Python. This trigger function monitors the raw area of the data lake. When it detects that a new log file has been successfully loaded, the trigger launches the transform function into execution. Various tests have been performed on loading the raw data into the DL with the aim of adjusting the transformation criteria so that we have as few non - conforming log lines as possible ending up in the logerrparsing directory.

After transforming the log files to parquet files, a reduction in storage space of about 90% from the original size was observed. This transformation thus comes with 2 major advantages: a significant reduction in storage costs and the transformation of the data into structured data that is easy to retrieve.

By achieving the objectives, we have demonstrated the capability and performance of a DL [21], which in the near future will become one of the most widely used storage media for diverse data types. DLs can store all types of data within an organization, in one central location, without the need to impose a schema upfront as with Data Warehouses. Unlike most databases and data warehouses, DLs can process all types of data that are essential for advanced analysis processes [22], [23]. The main purpose of a DL is to make all organizational data, from different sources, accessible to end users: Business Analysts, Data Engineers, Data Scientists, Product Managers, Executives, etc. Their access to these varied sources of data can maximize the information gained from them in a cost - effective way in order to improve an organization's performance [24], [25].

5. Conclusion

This research project demonstrates the importance of using the new Data Lake technology in the cloud to store large volumes of unstructured data. The new trend is to move and deploy a Data Lake in the cloud, which offers comprehensive services for the entire process of data ingestion, storage, processing, and analysis with a high level of security services that are being constantly improved by cloud service providers.

In this paper, we present a detailed theoretical description of the new cloud technology and report the practical implementation of an economical and reliable cloud Data Lake architecture in order to provide an optimal implementation for storing Web Server Access Log data that can then be subjected to various advanced analysis processes. A major advantage of storing in Azure Data Lake Server Gen2 is that it enables hierarchical data storage. Azure Data Lake Server Gen2, through its hierarchical namespace system, supports atomic operations which eliminates the risk of data loss encountered with flat storage. There are a number of important differences between flat storage and hierarchical storage in terms of performance and security, among which we can list:

- query performance, with a hierarchical file system it is possible to scan only certain partitions to obtain the data searched that improves the data query processes.
- performance moving data, in a hierarchical system renaming or moving files from one directory to another is done almost instantaneously.
- data consistency and atomic operations, which implicitly eliminates the risk of data loss if an error occurs during a move or rename operation, as with object - based storage.

References

[1] L. Korzeniowski and K. Goczyla, ''Landscape of automated log analysis: A systematic literature review and mapping study, '' IEEE Access, vol.10, pp.21892–21913, 2022, doi: 10.1109/ACCESS.2022.3152549.

[2] S. Hernandez, P. Alvarez, J. Fabra, and J. Ezpeleta, ‘‘Analysis of users behavior in structured e - commerce websites, ’’ IEEE Access, vol.5, pp.11941–11958, 2017, doi: 10.1109/ACCESS.2017.2707600.
[3] E. Zagan and M. Danubianu, ‘‘ADLS Gen 2 for web server log data analysis, ’’ in Proc. Int. Conf. Develop. Appl. Syst. (DAS), Suceava, Romania, 2022, pp.161–166, doi: 10.1109/DAS54948.2022.9786071.
[4] J. M. P. Jeba, M. S. Bhuvaneswari, and K. Muneeswaran, ‘‘Extracting usage patterns from web server log, ’’ in Proc.2nd Int. Conf. Green High Perform. Comput. (ICGHPC), Nagercoil, India, Feb.2016, pp.1–7, doi: 10.1109/ICGHPC.2016.7508074.
[5] P. Ghavare and P. Ahire, ‘‘Big data classification of users navigation and behavior using web server logs, ’’ in Proc.4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA), Pune, India, Aug.2018, pp.1–6, doi: 10.1109/ICCUBEA.2018.8697606.
[6] Nambiar and D. Mundra, ‘‘An overview of data warehouse and data lake in modern enterprise data management, ’’ Big Data Cognit. Comput., vol.6, no.4, p.132, Nov.2022, doi: 10.3390/bdcc6040132.
[7] G. Turkington and G. Modena, ‘‘Big data con Hadoop, ’’ Apogeo, 2015. [Online]. Available: https: //www.unilibro. it/libro/turkington - garry - modena - gabriele/big - data - con - hadoop/9788850333431
[8] Load Data Into Azure Data Lake Storage Gen2 With Azure Data Factory. Accessed: Jan.2022. [Online]. Available: https: //docs. microsoft. com/en - us/azure/data - factory/load - azure - data - lake - storage - gen2
[9] D. Sarramia, A. Claude, F. Ogereau, J. Mezhoud, and G. Mailhot, ‘‘CEBA: A data lake for data sharing and environmental monitoring, ’’ Sensors, vol.22, no.7, p.2733, Apr.2022, doi: 10.3390/s22072733.
[10] S. Kundu and L. Garg, ‘‘Web log analyzer tools: A comparative study to analyze user behavior, ’’ in Proc.7th Int. Conf. Cloud Comput., Data Sci. Eng., Jan.2017, pp.17–24, doi: 10.1109/CONFLUENCE.2017.7943117.
[11] B. Plejic, B. Vujnovic, and R. Penco, ‘‘Transforming unstructured data from scattered sources into knowledge, ’’ in Proc. IEEE Int. Symp. Knowl. Acquisition Model. Workshop, Wuhan, China, Dec.2008, pp.924–927, doi: 10.1109/KAMW.2008.4810643.
[12] Hadoop is Data’s Darling for a Reason. Accessed: Nov.2022. [Online]. Available: https: //www.forrester. com/blogs/hadoop - is - datas - darling - for - a - reason/
[13] The Unseen Data Conundrum. Accessed: Nov.2022. [Online]. Available: https: //www.forbes. com/sites/forbestechcouncil/2022/02/03/the - unseen - data - conundrum/?sh=4a07b7ac7fcc
[14] J. Sun, G. Gui, H. Sari, H. Gacanin, and F. Adachi, ‘‘Aviation data lake: Using side information to enhance future air - ground vehicle networks, ’’ IEEE Veh. Technol. Mag., vol.16, no.1, pp.40–48, Mar.2021, doi: 10.1109/MVT.2020.3014598.
[15] Ingestion and Processing Layers in Azure Data Lakehouse. Accessed: Dec.2021. [Online]. Available: https: //www.mssqltips. com/ sqlservertip/7037/azure - data - lakehouse - ingestion - processing - options/
[16] Z. Farzin. (2019). Online Shopping Store—Web Server Logs. Accessed: Sep.2021. [Online]. Available: https: //dataverse. harvard. edu/ dataset. xhtml?persistentId=doi: 10.7910/DVN/3QBYB5
[17] Azure Blob Storage Bindings for Azure Functions Overview. Accessed: Jan.2022. [Online]. Available: https: //learn. microsoft. com/en - us/azure/azure - functions/functions - bindings - storage - blob?tabs=in - process%2C functionsv2%2Cextensionv3&pivots=programming - language - python# trigger—polling
[18] Azure Data Lake Storage Pricing. Accessed: Feb.2023. [Online]. Available: https: //azure. microsoft. com/en - us/pricing/details/storage/data - lake/
[19] Configure a Lifecycle Management PolicyAccessed: Oct.2021. [Online]. Available: https: //docs. microsoft. com/en - us/azure/storage/blobs/lifecycle - management - policy - configure?tabs=azure - portal
[20] Optimize Costs by Automatically Managing the Data Lifecycle. Accessed: Oct.2021. [Online]. Available: https: //docs. microsoft. com/en - us/azure/storage/blobs/lifecycle - management - overview
[21] S. Villarroya, J. R. R. Viqueira, J. M. Cotos, and J. A. Taboada, ‘‘Enabling efficient distributed spatial join on large scale vector - raster data lakes, ’’ IEEE Access, vol.10, pp.29406–29418, 2022, doi: 10.1109/ACCESS.2022.3157405.
[22] Cuzzocrea, ‘‘Big data lakes: Models, frameworks, and techniques, ’’ in Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp), Jeju Island, Korea (South), Jan.2021, pp.1–4, doi: 10.1109/Big - Comp51126.2021.00010.
[23] H. Fang, ‘‘Managing data lakes in big data era: What’s a data lake and why has it became popular in data management ecosystem, ’’ in Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER), Shenyang, China, Jun.2015, pp.820–824, doi: 10.1109/CYBER.2015.7288049.
[24] J. C. Couto and D. D. Ruiz, ‘‘An overview about data integration in data lakes, ’’ in Proc.17th Iberian Conf. Inf. Syst. Technol. (CISTI), Jun.2022, pp.1–7, doi: 10.23919/CISTI54924.2022.9820576.
[25] F. Nargesian, K. Pu, B. Ghadiri - Bashardoost, E. Zhu, and R. J. Miller, ‘‘Data lake organization, ’’ IEEE Trans. Knowl. Data Eng., vol.35, no.1, pp.237–250, Jan.2023, doi: 10.1109/TKDE.2021.3091101.
[26] T. A. Al - Asadi and A. J. Obaid, ‘‘Discovering similar user navigation behavior in web log data, ’’ Int. J. Appl. Eng. Res., vol.11, no.16, pp.8797–8805, 2016.
[27] R. Hai, S. Geisler, and C. Quix, ‘‘Constance: An intelligent data lake system, ’’ in Proc. Int. Conf. Manag. Data, Jun.2016, pp.2097–2100.
[28] Gorelik, The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science. Sebastopol, CA, USA: O’Reilly, 2019.
[29] S. Park, B. Cha, and J. Kim, ‘‘Design and implementation of con - nected DataLake system for reliable data transmission, ’’ in Proc.23rd Int. Comput. Sci. Eng. Conf. (ICSEC), Oct.2019, pp.141–144, doi: 10.1109/ICSEC47112.2019.8974823.