ADLS Gen 2 for web server log data analysis

Elisabeta ZAGAN
Faculty of Electrical Engineering and Computer Science
Stefan cel Mare University
Suceava, Romania
elisabeta.b@gmail.com

Mirela DANUBIANU
Faculty of Electrical Engineering and Computer Science
Stefan cel Mare University
Suceava, Romania
mdanub@eed.usv.ro

Abstract— **Before the advent of Big Data, there were few possibilities for processing TB of data sets or more. As data generation capacity has grown, so has the need to store and process large volumes of data. The web server log files contain important and valuable information about events related to customer activities in the online environment, server activities and its response to customer requests, and much more. It is well known that web server log files are files that contain large volumes of data and grow very quickly in size, servers having to delete them once they reach a certain size. Thus, many valuable data are lost, data on which various analyzes can be made in order to obtain valuable information that will later lead to improvements on several levels. It is important to find cost-effective and easily accessible techniques for storing and analyzing these files, which are also considered to be of a high privacy level holding information classified as personal data. The main objective is the research for the development and implementation of an innovative solution for storing and analyzing large volumes of web server log files using cloud storage - ADLS Gen2.**

Keywords— Data Lake, web server log data, cloud Data Lake, ADLS Gen2.

I. INTRODUCTION

As Data Lake technology has gained more and more attention due to its great advantages in an information technology world where data is generated amazingly fast and is increasingly varied. Data Lake is a new technology with great potential, so it is expected that in the future companies will have to use it to make the most of the information that can be obtained from the variety of data they have. Web servers are accessed by millions of users every day. Users leave behind their so-called "visiting behavior" by recording their online activities in web log files. By analyzing the web server log files recorded on these servers, as well as on the proxy servers, it is possible to discover these "visiting behaviors", which helps to improve and customize the services offered to users [1]. A web log is a text document that contains all the activities of a server. It is automatically created and maintained by the server and can provide important information about its activity. Most servers generate Common Log Format (CLF) files or raw log files, which are quite crude and difficult to understand.

They can also take up a lot of storage space in a short period of time, which is why they are automatically deleted after a period of time. For example, some servers are configured to delete log files after two weeks, others delete files after they exceed a certain size.

A Data Lake architecture can be implemented in several ways, namely On-Premise, Cloud, Hybrid, and Multi-Cloud Data Lake. The choice of an architecture is made based on the requirements and the financial resources that a company have, being a technology that requires not negligible financial efforts.

The implementation of a Data Lake must take into account all the constituent processes that involve the use of both hardware and software resources. Data Lake On-Premises such as Hadoop is a physically implemented architecture locally within a company [2]. This implementation requires the local management of everything that involves building a Data Lake locally. Cloud Data Lake is the architecture that allows the implementation of a Data Lake in Cloud. The biggest cloud service providers have adapted to the new requirements in the field of Big Data and have implemented high-performance cloud services [3] for the successfully implementation of such an architecture. The Hybrid Data Lake architecture allows data lakes to be maintained both locally and in the Cloud [4]. The major advantage of this type of architecture is that less relevant data can be stored locally, in order to reduce storage costs, while still benefiting from the speed of Cloud services corresponding to important data. The Multi-Cloud Data Lake architecture offers the use of combined Cloud services from different providers [5]. Thus, there are large companies that use both AWS and Microsoft Azure to manage and maintain data gaps. This article addresses issues related to storing, structuring, and managing web server log files with Azure Data Lake Storage Gen2 (ADLS Gen2), while also highlights the importance of the data contained in these log files.

This article is structured as follows. After a brief introduction, Chapter II presents related work, and Chapter III describes the purposed solution and the advantages of storing web server log file into ADLS Gen2 Data Lake. Chapter IV describes the ADLS Gen2 main characteristics and in Chapter V it is presented a data structuring model in ADLS Gen2 that will serve the various ELT processes for managing web server log data. The article concludes with the conclusions of Chapter VI.

II. RELATED WORK

Data Lake is the latest technique for storing and analyzing large volumes of raw data in order to obtain sets of information that will improve decision-making processes in various fields

This work is supported by the project “119722/Centru pentru transferul de cunoștințe către întreprinderi din domeniul ICT – CENTRIC – Contract subsidiar 15682/02.09.2020/CRIODRIVE/Mechatronics Innovation Center”, contract no. 5/AXA 1/1.2.3/G/13.06.2018, cod SMIS 2014+ 119722 (ID P_40_305).

of activity. The information obtained can then be used to optimize processes, to increase the efficiency of a business or a system as we will see below. In [6] the authors present how local and global secondary information is obtained, organized, managed and used through a so-called aeronautical Data Lake. The Data Lake can be easily connected with advanced machine learning schemes and thus provides timely, context-aware values and predictions.

The Data Lake prototype, presented in [7], ingests data from several sources, including FAA sources such as SFDPS, TFMData, TBFM, STDDS, ITWS, and AEDT data sources, and stores them in raw, processed, and refined format. The prototype provides an illustration of how users can perform powerful air traffic data analysis using structured, unstructured, and semi-structured data using open source tools to perform queries, searches, stream processing, and view data. The purpose of a Data Lake is to present an unrefined view of the data to only the most qualified analysts, to help them explore their data refining and analysis techniques independently of any of the compromises of the recording system that may exist in an analysis traditional data repository (such as a March date or a data repository). Data Lake is an architectural design model that allows businesses to store a huge amount of data in its native format as needed. For a Data Lake, typical Big Data challenges, such as Variety, Speed, and Data Volume, must be resolved, as specified in [8]. Due to the nature of the project and its data, however, there are also new atypical challenges. From the literature, it is common practice to define multiple layers during the Data Lake design phase. For example, in [9], the authors introduced a three-layer Data Lake, namely assimilation, maintenance, and querying. In [10] a basic conceptual design of a connected data system based on micro cloud storage was proposed. Thus, a connected Data Lake system based on distributed cloud storage has been designed and implemented, which securely provides data stored in multiple Edge Clouds to micro cloud storage. In addition, the system performs real-time error recovery so that the transferred data can be restored to an abnormal point when an anomaly occurs during transmission.

Log files can provide a lot of information about events related to client activities, server activities, and so on. Researchers face the need to analyze large web log files to ensure reliable, secure and high-performance services. In [11] are presented log file analysis algorithms in order to obtain patterns by identified users along with their navigation behaviors, clustering similar users based on different interesting log file content from different websites.

III. Web server Log Files in Data Lake

Web log files are files in which each line represents a request. This data stored in the log files can provide several information such as: the IP address, the identity of the device that placed a request on the server, the name, location and size of the requested file, time and date of the request, respectively the method of request, client browser information, the page from which the user left the site, server response, HTTP status code, duration and number of pages visited by the user, the type of search engine used and the search term entered, if it is a direct access to a user or a redirect from another site, or is part of an advertising campaign, etc.

All this data can provide, a series of information that will help to improve the security and the structure of the website, the performance of the web server, the traffic to the website, detects errors on the website, detects attacks that can occur spontaneously on certain segments and without being noticed immediately and is an important tool that can help in improving the SEO of the website. An analysis of log files can provide information on how search engine robots access and index sites so that they can be optimized for the best possible ranking, and so on. Because log files are generated automatically, the amount of data can quickly become very large relative to the number of visitors. As log data grows, so does the need for long-term file storage, forcing companies to find an optimal solution for large-scale indexing and analysis of log files. Thus, researchers face the need to analyze large web log files to ensure reliable, secure and high-performance services. For small sites, log file analysis can be easily done either directly by reading them or by using various online or open-source services [12], which provide solid support for log file analysis. However, these services come with a number of limitations, either in the size of the files to be analyzed or in the process of analyzing them, so another approach must be found for the analysis of large log files. Also, log files are files with sensitive data for which the use of online analysis services is often not a desired solution. As we have seen above, one of the problems with large logs is their storage. An optimal solution can be the implementation of a Data Lake that offers data storage services in their natural form with automatic auto-scaling that allows unlimited data storage. This research aims to present an optimal solution for storing large web server log files with no storage limit at low cost, so that they can later be subjected to advanced analysis processes, relying on Azure cloud resources in order to implement a successful ADLS Gen2 architecture. Saving log files into a Data Lake brings maximum flexibility in storing and analyzing long-term data at reasonable costs and allows full control over all data held in the log files. Nowadays, the requirement to examine these log files even after long periods is significantly increased.

IV. Azure Data Lake Storage gen 2 – main characteristics

In designing a Data Lake, it is necessary to take into account a number of important factors related to the complexity of the implementation. Thus, there is no single structuring method for building a Data Lake, each implementation being unique starting from the ingestion layer to the analysis layer.

A successful Data Lake must take into account a number of characteristics that belong to each level of architecture, such as data ingestion, storage, transformation and analysis, as well as the level of data consumption. (Figure 1). ADLS Gen2 is a highly scalable, cost-effective and productive Data Lake for large-scale data analysis, structured data extraction from unstructured data using transformations, advanced analysis, machine learning or for real-time data analysis in order to obtain useful information.

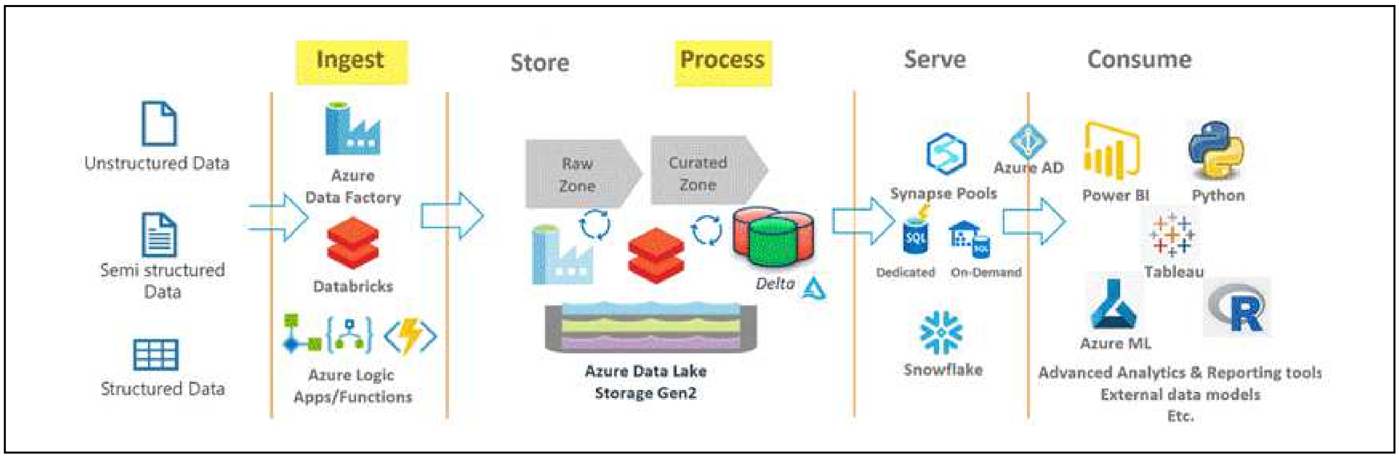


Fig. 1. The main processes related to the Azure Data Lake Storage Gen2 architecture [13]

It is a completely cloud-based solution that does not require the installation of any hardware or server systems, it scales automatically, and allows data to be ingested from different sources and at different speeds. In terms of Data Lake deployment costs, ADLS Gen2 comes with the same storage price as blob storage, which is already known to be the most economical Big Data storage system. In ADLS Gen2 only the used storage space is paid, and there is no concept of reserving a certain storage space at a fixed cost. Storage costs can be optimized as needed through Data Lifecycle Management (DLM). It should be noted, however, that the costs for the various transactions are somewhat higher in the case of hierarchical storage [14].

The ingestion level is the first level in the whole architecture. Data assimilation usually involves two basic steps, namely extracting data and uploading data to Data Lake. There are a variety of platforms and frameworks that not only automate the loading of data into Data Lake, but can also perform a number of data quality checks. Data ingestion frameworks can collect data from multiple sources, where it detects and captures changes and reproduces them in Data Lake. Most data ingestion frameworks consist of two main components, namely Data Collector and Data Integrator. The Data Collector component collects data, while the function of the Data Integrator module is to ingest and integrate data into Data Lake. The storage level in ADLS Gen2 is a hierarchical storage that allows data to be organized into containers, directories, and subdirectories for better organization of different areas of Data Lake. ADLS Gen2 takes advantage of the hierarchical file system without sacrificing the scalability and cost-effectiveness of blob storage. The structuring of data in a Data Lake differs from case to case and most of the time cannot be predicted from the beginning. Thus, it is recommended to implement several areas in the basic structure, in order to allow easy extension and subsequent modeling. These areas will be shared areas that will delimit the data in their transformation process (Figure 2). This existing delimitations also allow the assignment of personalized access to the data to strengthen their security, as we will see below.

The areas in the basic Data Lake architecture for structuring data in ADLS Gen 2 are as follows:

- Landing Zone: This zone is optional and transitional that allows data to be passed from source to Data Lake. Thus, it is an area where valid data can be distinguished from invalid data. Valid data is automatically processed in the Raw Zone, and the rest will be stored in a quarantine area for further manual verification.
- Raw Zone: This zone is where the raw data will be stored, with a restricted access. This level of data is controlled by Data engineers and is rarely offered to other consumers. This zone can be organized using a directory associated with each source, each ingestion process having write access only to the associated directory. The directory structure is generally given by the frequency with which data is transferred or can be configured as needed. The data, as needed, can be structured as follows: Data Source > Year > Month > Day > Hour.
- Enriched, Cleansed or Standardized Zone: This zone will have the data subjected to a rough process of cleaning and improvement, a so-called standardization process of data suitable for the next zone called Curated zone. The main data processing that can take place here are those for defining data types (eg .jpg images, etc.), deleting unnecessary data, or applying data cleaning rules. Data improvement processes can also take place in this area by combining data sets in order to obtain more valuable information. The data structure of this area is similar to that of the Raw Zone. This area can be considered as a parking area on which write permissions are granted for certain automatic jobs. Data permissions can also be assigned to read data scientists.
- Curated Zone: This is the area with valuable data, which has undergone transformation and cleaning processes in order to be ready for consumption. The data resources located at this level are very well governed and documented. This area can be accessed by Data analysts, Data scientists but also by Data engineers for advanced analysis processes, ad-hoc queries, and more. At this level the structure of the

existing data will be rather customized according to the areas of interest, such as: Subject Areas > Year > Month; Subject Areas > Region; Subject Areas > Sales.

- Sensitive zones: This is the area with sensitive data that is characterized by stricter access permissions. This area allows you to define a separate Data Lifecycle Management (DLM) policy using prefix matching rules. It is an optional area, as not all companies will need it. The structure of the directories is given as needed, it will largely depend on the type of data access rules of this area.
- Sandbox, Laboratory or Workspace Zone: This is the exploration and experimentation zone accessed by Data scientist, Data engineers, Data analysts, who understand the company's data and needs. Here everyone has the opportunity to create their own models, their own data sets with the ultimate goal of obtaining important information for analysis. The data is structured on projects, so that personalized access can be granted to work teams.

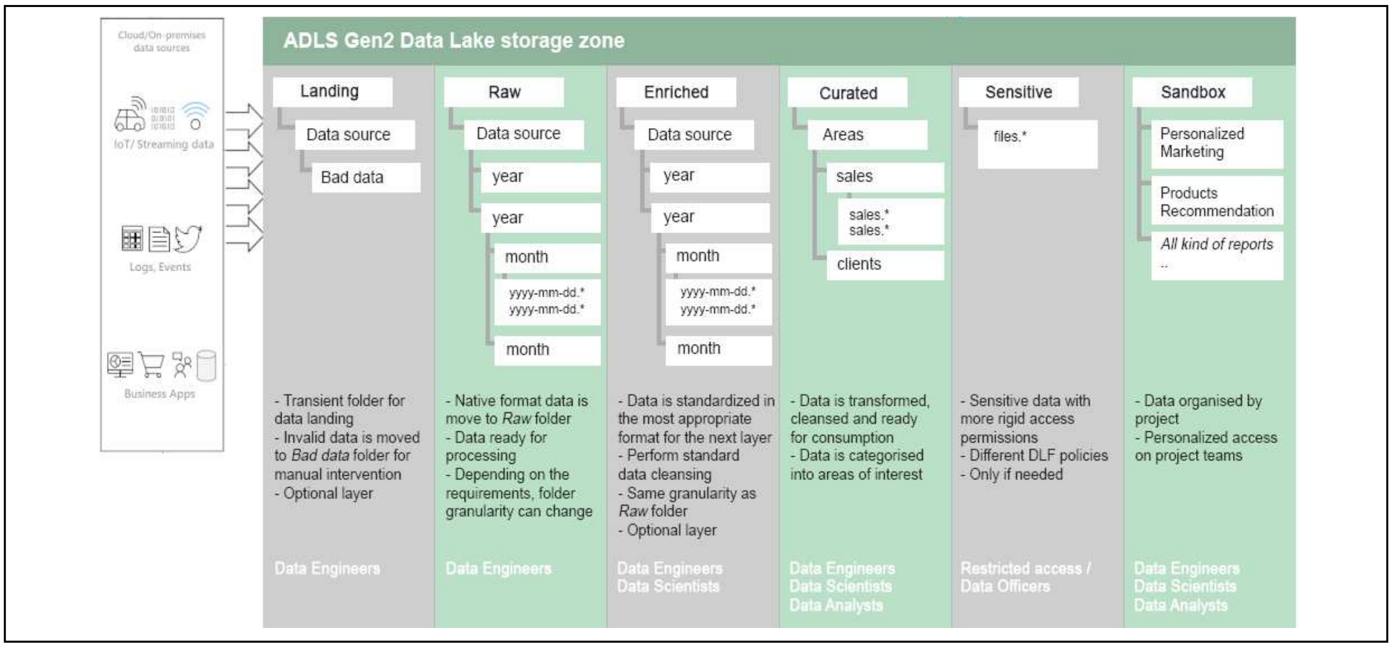

Cloud/On-premises data sources
IoT/ Streaming data
Logs, Events
Business Apps
ADLS Gen2 Data Lake storage zone
Landing
Data source
Bad data
Raw
Data source
year
year
month
yyyy-mm-dd.*
yyyy-mm-dd.*
month
Enriched
Data source
year
year
month
yyyy-mm-dd.*
yyyy-mm-dd.*
month
Curated
Areas
sales
sales.*
sales.*
clients
Sensitive
files.*
Sandbox
Personalized Marketing
Products Recommendation
All kind of reports
..
- Transient folder for data landing
- Invalid data is moved to Bad data folder for manual intervention
- Optional layer
- Native format data is move to Raw folder
- Data ready for processing
- Depending on the requirements, folder granularity can change
- Data is standardized in the most appropriate format for the next layer
- Perform standard data cleansing
- Same granularity as Raw folder
- Optional layer
- Data is transformed, cleansed and ready for consumption
- Data is categorised into areas of interest
- Sensitive data with more rigid access permissions
- Different DLF policies
- Only if needed
- Data organised by project
- Personalized access on project teams
Data Engineers
Data Engineers
Data Engineers
Data Scientists
Data Engineers
Data Scientists
Data Analysts
Restricted access /
Data Officers
Data Engineers
Data Scientists
Data Analysts

Fig. 2. Data Lake architecture storage levels.

When deciding the Data Lake structure, both the semantics of the data and the type of consumers accessing the data, as specified above, must be considered. Thus, the elimination of one of these areas and the introduction of a new area are not excluded. Also, not all areas explicitly need to be physically located in the same Data Lake [15] so they can be deployed on different storage media. The implementation of a Data Lake will be adapted to the needs depending on the data source, the frequency with which the data is generated, the requirements for structuring and accessing the data, and more. The most used structure in a Data Lake consists of the *Raw, Enriched, Curated* and *Sandbox* areas. ADLS Gen2 is part of a wide variety of architectures implemented for advanced analysis of large volumes of data. Once the data is brought into the desired shape and structure, they are to be subjected to various analysis processes in order to be ready for consumption.

V. ADLS GEN2 STRUCTURE FOR WEB SERVER LOG FILES

To create an Azure Data Lake Storage Gen 2, you first need three resources that define the main features of the new Data Lake:

- Subscription: depending on the selected subscription, you may have access or limitations to ADLS Gen2 resources.
- Resource group: a logical container that stores metadata about different resources needed for an Azure solution to be managed together as a group.
- Storage account: It is an Azure resource that contains all Azure Storage data objects, such as blobs, files, queues, tables, and disks.

The ADLS Gen2 resources used for this research is an Azure Blob Storage account with the hierarchical namespace enabled. We will present the proposed data structuring model in ADLS Gen2 that will serve the varius ELT processes for managing web server log data. It is recommended to implement several areas in the basic structure for easy extension and further modeling. An *adlswebserver1* master container has been created that will serve the data came from, namely *server1* source. For data comming from different servers, new master containers will be created under dedicated resource groups, for separate management and for security sharing by source group. In our data storage model using Data Lake we have the raw data area (*Raw*), the transformed data area –

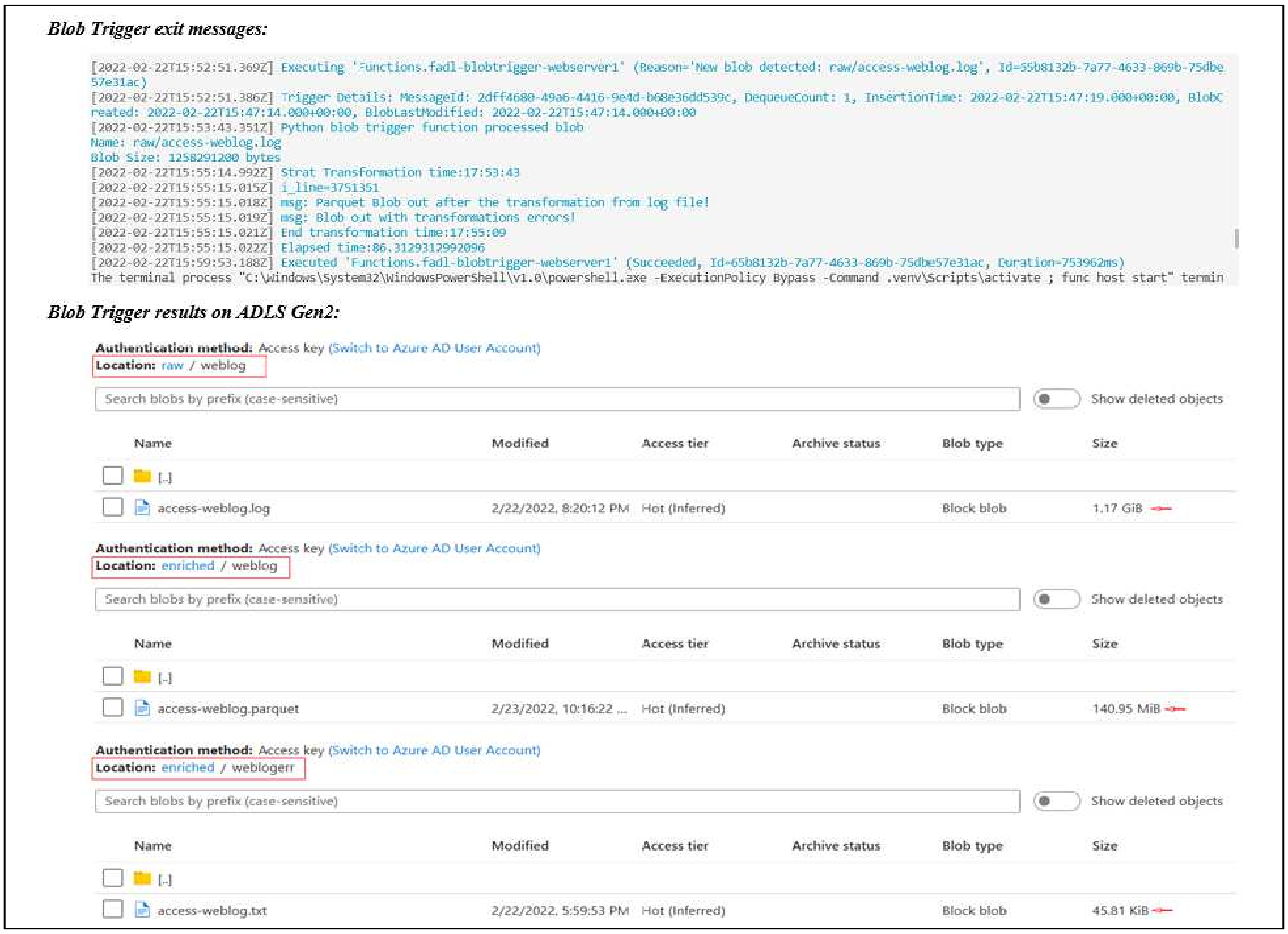
(*Enriched*), the ready-to-consume data area (*Curated*), and the data exploration and experimentation area (*Workspace*). In the *Enriched* area there is a *weblogerr* folder that will contain the files with the error lines resulting from the transformation process.

The ingestion process consists of transferring data from *server1* to ADLS Gen2 into the *raw/weblog/* directory. The ingestion of the *access-weblogs.log* file of 1.17G size from local to Cloud Data Lake was performed using the Azure CLI, which is a command-line cross-platform tool that allows the execution of administrative commands on resources in the Azure Storage account.

```
#Azure CLI – data ingestion sequence code
$adls_acct_name = "adlswebserver1"
$logFileToUpload = $source_dir + "access-weblogs.log"
$adls_dir_name = "raw/weblog/"
az storage blob upload --account-name $adls_acct_name --
container-name $adls_dir_name --name 'access-weblogs.log'
--file $logFileToUpload --auth-mode login
```

Over the processing level, transformations take place on raw data to obtain structured data that can be easily analyzed. Raw data loaded in the *raw/weblog/* directory will then be processed and uploaded to the *enriched/weblog/* directory. Thus, the errors resulting from the transformation process will be uploaded to the *enriched/weblogerr/* directory for further analysis. Raw log files will be converted to .parquet format files, that are organized in columns, so data can be read individually significantly improving system performance.

Using parquet files compresses data by greatly reducing the initial file size and thus significantly reducing storage costs. Log files under *raw/weblog/* directory that have already been transformed from log to parquet files, can be transferred to a cooler tire for archiving in order to reduce storage costs for raw data. The transformation of .log files into .parquet files was done by implementing an Azure serverless Blob Trigger function that detects when a new log file is loaded and starts its transformation automatically into a Parquet file. The Azure Functions tool is an extension for Visual Studio Code that allows you to perform serverless functions locally, which can then be published online directly in Azure. To implement the function that underlies the process of transforming log files to parquet files, Python libraries were used because they facilitate this process at the procedural procedural level.



Blob Trigger exit messages:

```
[2022-02-22T15:52:51.369Z] Executing 'Functions.fadl-blobtrigger-webserver1' (Reason='New blob detected: raw/access-weblog.log', Id=65b8132b-7a77-4633-869b-75dbe
57e31ac)
[2022-02-22T15:52:51.386Z] Trigger Details: MessageId: 2dff4680-49a6-4416-9e4d-b68e36dd539c, DequeueCount: 1, InsertionTime: 2022-02-22T15:47:19.000+00:00, BlobC
reated: 2022-02-22T15:47:14.000+00:00, BlobLastModified: 2022-02-22T15:47:14.000+00:00
[2022-02-22T15:53:43.351Z] Python blob trigger function processed blob
Name: raw/access-weblog.log
Blob Size: 1258291200 bytes
[2022-02-22T15:55:14.992Z] Strat Transformation time:17:53:43
[2022-02-22T15:55:15.015Z] i_line=3751351
[2022-02-22T15:55:15.018Z] msg: Parquet Blob out after the transformation from log file!
[2022-02-22T15:55:15.019Z] msg: Blob out with transformations errors!
[2022-02-22T15:55:15.021Z] End transformation time:17:55:09
[2022-02-22T15:55:15.022Z] Elapsed time:86.3129312992096
[2022-02-22T15:59:53.188Z] Executed 'Functions.fadl-blobtrigger-webserver1' (Succeeded, Id=65b8132b-7a77-4633-869b-75dbe57e31ac, Duration=753962ms)
The terminal process "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -Command .venv\Scripts\activate ; func host start" termin
```

Blob Trigger results on ADLS Gen2:

Authentication method: Access key (Switch to Azure AD User Account)
Location: raw / weblog

Search blobs by prefix (case-sensitive) | Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
access-weblog.log	2/22/2022, 8:20:12 PM	Hot (Inferred)		Block blob	1.17 GiB

Authentication method: Access key (Switch to Azure AD User Account)
Location: enriched / weblog

Search blobs by prefix (case-sensitive) | Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
access-weblog.parquet	2/23/2022, 10:16:22 ...	Hot (Inferred)		Block blob	140.95 MiB

Authentication method: Access key (Switch to Azure AD User Account)
Location: enriched / weblogerr

Search blobs by prefix (case-sensitive) | Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
access-weblog.txt	2/22/2022, 5:59:53 PM	Hot (Inferred)		Block blob	45.81 KiB

Fig. 3. ADLS Gen2 – Azure Blob Trigger function results.

Once new .log files are uploaded to the raw/weblog/ directory, the Blob Trigger launches the Python function in execution starting the transformation process automatically. In this case, the transformation time was about 12 minutes, respectively 753962 ms from the moment the trigger detected a successfully completed upload into the monitored directory. Figure 2 shows the results of the Blob Trigger function and the results of data ingestion and transformation into Data Lake. After the transformation, from the original file of 1.17GB of log data it was obtained one parquet file access-weblog.parquet of only 140MB and one text file access-weblog.txt with 45 KB transformation errors. The file has been reduced from 1.17GB to 140MB, which is 11.9% of the original size. As for the 45KB text file containing the log lines that couldn't be converted, it will undergo further evaluation and may be needed to adjust the python regex in order to reduce to zero the log lines losts as a result of the transformations.

Once the transformation process is adjusted, if the transformation is successful, you can move on to the next step, which is to improve and clean the parquet file, so that it can be transferred in an optimal shape to the curated area. In the curated area the data will undergo significant transformations that will serve the final purpose of obtaining specific information. Oance having the curated data, different analysis processes can be applied like:

- Static aggregation and Statistics – most common, that provides fast data visualization by extracting simple reports without applying complex analysis processes;
- Session Analysis based on Heuristic analysis that analyzes groups behavior;
- Web usage mining which is a data mining techniques to discover usage patterns from web log data.

VI. Conclusions

In terms of Data Lake application areas, the implementation proposed in this article is a low-cost, forward-looking solution with outstanding performance in terms of log data analysis based on Azure Data Lake Storage Gen2 in order to gain the maximum insights from web server log data. It can be a powerful tool to improve SEO strategies, to improve the online environment security and site structure, increasing the performance of web servers and troubleshootings.

In this article the authors highlights the importance of web server log files and the valuable information they contain, and proposed a technique for storing large volumes of web server log data using Data Lake architecture. Thus, this research proposes a solution for storing and transformig web server log data in the cloud using ADLS Gen2. In order to reduce the storage costs by physically reducing the data storage, it was proposed the usage of an Blob Trigger function in Python that transforms the log files into parquet files. A hierarchy has also been proposed for structuring data into the Data Lake in order to allow controlled data management and also to allow cost management through DLF. This is a low-cost, high-performance solution that can be expanded to store and process large volumes of web server log files.

Acknowledgment

This research was funded by the project "119722/Centru pentru transferul de cunoștințe către întreprinderi din domeniul ICT – CENTRIC – Contract subsidiar 15682/02.09.2020/CRIODRIVE/Mechatronics Innovation Center", contract no. 5/AXA 1/1.2.3/G/13.06.2018, cod SMIS 2014+ 119722 (ID P_40_305), using the infrastructure from the project "Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control", contract no. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness co-funded from the European Regional Development Fund.

References

[1] S. Kundu and L. Garg, "Web log analyzer tools: A comparative study to analyze user behavior," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 17-24, doi: 10.1109/CONFLUENCE.2017.7943117

[2] Garry Turkington, Gabriele Modena, "Big data con Hadoop", May 2015, ISBN: 9788850333431.

[3] https://docs.microsoft.com/en-us/azure/data-factory/load-azure-data-lake-storage-gen2 (Accessed Jan. 2022).

[4] https://www.datanami.com/2021/06/07/back-to-basics-big-data-management-in-the-hybrid-multi-cloud-world/ (Accessed Apr. 2021)

[5] https://www.datanami.com/2021/06/07/back-to-basics-big-data-management-in-the-hybrid-multi-cloud-world/ (Accessed Apr. 2021)

[6] Sun, Jinlong; Gui, Guan; Sari, Hikmet; Gacanin, Haris; Adachi, Fumiyuki. "Aviation Data Lake: Using Side Information to Enhance Future Air-Ground Vehicle Networks" IEEE Vehicular Technology Magazine, pp. 40-48, September 2020. doi: 10.1109/MVT.2020.3014598.

[7] Raju, Ramakrishna; Mital, Rohit; Finkelsztein, Daniel. "Data Lake Architecture for Air Traffic Management", 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), pp. 1-6, September 2018. doi: 10.1109/DASC.2018.8569361.

[8] Ben Hamadou, Hamdi; Pedersen, Torben; Thomsen, Christian. "The Danish National Energy Data Lake: Requirements, Technical Architecture, and Tool Selection", 2020 IEEE International Conference on Big Data (Big Data), pp. 1523-1532, December 2020. doi:10.1109/BigData50022.2020.9378368.

[9] R. Hai, S. Geisler, and C. Quix, "Constance: An intelligent data lake system," in SIGMOD, 2016.

[10] Park, Sun; Cha, Byung; Kim, Jongwon. "Design and Implementation of Connected DataLake System for Reliable Data Transmission", 2019 23rd International Computer Science and Engineering Conference (ICSEC), pp. 141-144, October 2019. doi:10.1109/ICSEC47112.2019.8974823.

[11] S. Kundu and L. Garg, "Web log analyzer tools: A comparative study to analyze user behavior," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 17-24, doi: 10.1109/CONFLUENCE.2017.7943117

[12] https://en.ryte.com/magazine/ultimate-guide-log-file-analysis (Accessed Oct. 2021)

[13] https://www.mssqltips.com/sqlservertip/7037/azure-data-lakehouse-ingestion-processing-options/ (Accessed Dic. 2021)

[14] https://azure.microsoft.com/en-us/pricing/details/storage/data-lake/ (Accessed Jan. 2022)

[15] https://azure.github.io/Storage/docs/analytics/hitchhikers-guide-to-the-datalake/?WT.mc_id=helloworld-17228-cxa#key-considerations-in-designing-your-data-lake (Accessed Oct. 2021)