

Data Lake Architecture for Storing and Transforming Web Server Access Log Files

*Report submitted to the SASTRA Deemed to be University
as the requirement for the course*

CSE300 - MINI PROJECT

Submitted by

SREE RENGABALAN S
Reg No.:125003334, B.Tech CSE
RAGHURAM SHRINATH R
Reg No.:125003244, B.Tech CSE
SAKTHI GANESH M
Reg No.:125003279, B.Tech CSE

May 2024



SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

T H A N J A V U R | K U M B A K O N A M | C H E N N A I



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401



SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled “**Data Lake Architecture for Storing and Transforming PROJECT**” for B.Tech. is bonafide record of the work done by **Mr. Sree Renga Balan (125003334, B. Tech CSE), Mr. Raghuram Srinath R(125003244, B. Tech CSE), Mr. Sakthi Ganesh(125003279, B. Tech CSE)** during the academic year 2023-24, in the School Of Computing, under my supervision.

Signature of the Project Supervisor:

Name with Affiliation: Dr.Saravanan N

Date:22.04.2024

Mini Project *Viva voce* held on _____

Examiner 1

Examiner 2

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S.Vaidhyasubramaniam** and **Dr.S.Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R.Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr.V.S.Shankar Sriram**, Dean, School of Computing, **Dr.R.Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr.D.Manivannan**, Associate Dean, Infrastructure, **Dr.R.Algeswaran**, Associate Dean, Students Welfare

Our guide **Saravanan N,Asst. Professor – III** , School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through project.

List of Figures

Figure No.	Title	Page No.
1	proposed DL architecture and various ELT processes serving WSAL Files	2
4.1	Creating IAM user using cli and setting password	9
4.2	Listing IAM users present in group using cli	9
4.3	Splitting log files into smaller chunks and viewing them	10
4.4	Viewing no.of lines present in each log files which are seperated to verify whether it matches with the dataset	10
4.5	Results from the base paper	10
4.6	Log files stored in the projectminilog-1 S3 bucket	11
4.7	Transformed Parquet files stored in outputbucketmini S3 bucket	11
4.8	Error text files stored in erroroutputmini S3 bucket	11
4.9	Datalife cycle management policy	12
4.10	Storage space reduction achieved by transforming log files into parquet files	13
4.11	Comparing expected and recorded outputs	14

Abbreviations

AWS	Amazon Web Services
S3	Simple Storage Service
IAM	Identity & Access Management
IP	Internet Protocol
HTTP	Hypertext Transfer Protocol
WSAL	Web Server Access Logfiles

Abstract

Web server log file is a record of all requests that are made to web server. This file contains informations such as IP address of client, HTTP method, timestamp and user-agent etc. The main issue is storing these files for long time so that we can analyze them whenever need. Azure Data Lake Storage Gen2 (ADLS Gen2), a cloud data lake architecture, is employed for storing large amounts of log files in their raw form. This ADLS integrates with ELT processes for further advanced analysis. This provides an affordable and more accessible solution for performing web server access log data ingestion, storage, and transformation using the latest technology, Data Lake. Additionally we also utilize Azure Blob Trigger Function to execute the algorithm for transforming log files into parquet files. This implementation results in a 90% reduction in storage space compared to the original size, which leads to much lower storage costs. A hierarchical data storage model has also been proposed for shared access to data over different layers in the DL architecture, on top of which Data Lifecycle Management (DLM) rules have been proposed for storage cost efficiency. The proposal suggests the ingestion of log files into a cloud-deployed Data Lake for the ease of deployment and cost-effectiveness. The primary goal is to store this data for the long term, facilitating future advanced analytics processes through cross-referencing with other organizational or external data. The anticipated benefits include valuable insights that can be derived from the analysis of this integrated data. While the proposed solution is explicitly based on ADLS Gen2, it represents an important benchmark in approaching a cloud data lake solution offered by any other vendor, such as AWS, Google Cloud, and so on.

Keywords: Blob Trigger Function, Parquet Files, Log Files, S3 Bucket.

Table of Contents

Title	Page No.
Bonafide Certificate	ii
Acknowledgements	iii
List of Figures	iv
List of Tables	iv
Abbreviations	v
Abstract	vi
1. Summary of the base paper	1
2. Merits and Demerits of the base paper	4
3.Source Code	5
4. Snapshots	8
5. Conclusion and Future Plans	15
6. References	16
7. Appendix -Base Paper	17

CHAPTER 1

SUMMARY OF THE BASE PAPER

Title	:	Data Lake Architecture For Storing and Transforming Web Server Access Log Files
Publisher	:	Elisabeta Zagan And Mirela Danubianu
Year	:	25 April 2023
Journal	:	IEEE
DOI	:	10.1109/ACCESS.2023.3270368
URL	:	https://ieeexplore.ieee.org/document/10107911

1.1 Introduction:

Web server access log files are records generated by web servers, capturing details of each request made to the server. Analyzing these records can help understand how users visit the website over time, providing insights into what customers want. These log files are difficult to store, process and analyze at large scale, so we proposed storing them in a Data lake architecture.

Moreover, current analytics tools depend on JavaScript code and cookies, potentially leading to extended loading times and various technical issues. If the user had blocked the cookies in their personal device the results we are generating from google analytics is not possible. Hence storing log files in data lake for long period helps us to analyse about all the users needs.

We utilize S3 buckets, which are storage containers offered by AWS, to store our log files.

1.1.1 S3 BUCKETS:

S3 buckets are storage containers for storing data in the AWS cloud. It has fine grained access control mechanisms. Policies can be attached to a bucket to determine who can access them. It also has multiple storage tiers like Standard, Standard-IA, Intelligent-tiering, Glacier and Glacier Deep Archive. Each of these storage tiers has its own functionalities to store the files. We can also apply data lifecycle policies to S3 buckets, defining how the bucket should behave over time.

1.1.2 WEB SERVER ACCESS LOG FILES:

Web server access log files store various pieces of information about each request made to the server. those are as follows

- IP address of the client making the request
- Request time and date
- Information about the client browser
- What search engine was used and search terms used
- HTTP Method used
- Number of bytes transferred
- Page from which the client left the site
- Response sendd by the server

1.2 Methodology

Proposed architecture:

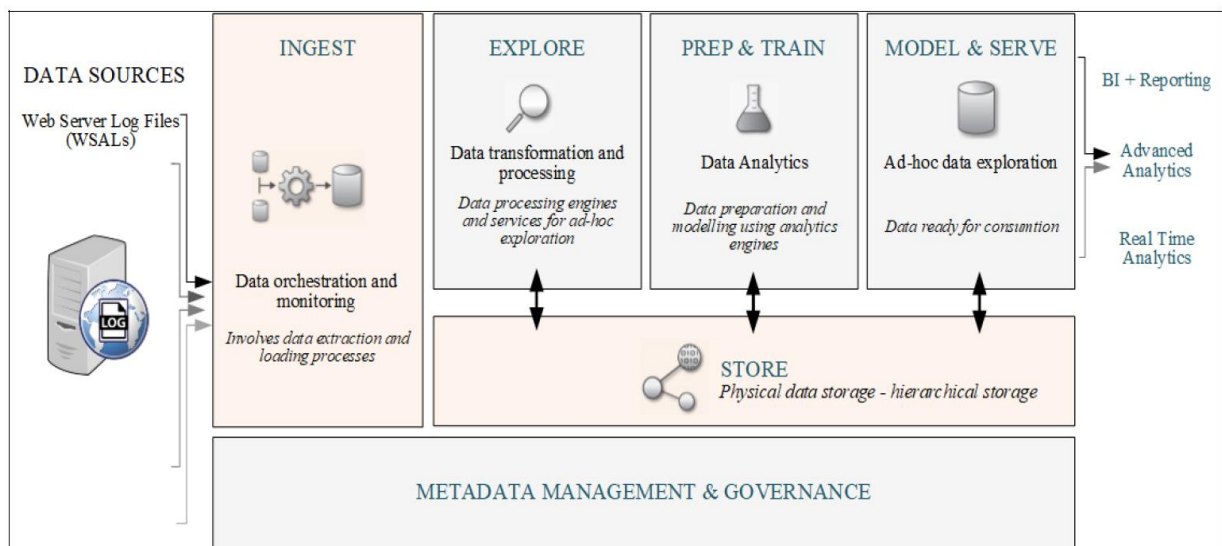


Fig 1. proposed DL architecture and various ELT processes serving WSAL Files

1.2.1 Steps Involved:

Uploading Log files into S3 bucket: consider a scenario where a web server is generating log files for our website. Once the log file size exceeds the specified threshold, which is 1GB in our case, we trigger the upload process to transfer the log file to the S3 bucket. This task can be accomplished using various methods such as a cron job, a scheduled task, or a custom script running on the web server. However, as we don't have a live running server, we are utilizing a dataset obtained from the internet, splitting it into smaller chunks, and uploading them into the S3 bucket.

Triggering Function: Once the log files are uploaded into the S3 bucket, a triggering function will be activated to convert them into Parquet files and store them in a separate bucket. This triggering function will only be enabled when it detects a new file uploaded into the S3 bucket.

Data life cycle policy: Finally, we will implement a data lifecycle policy for the S3 buckets to optimize costs. For instance, lifecycle policies can be utilized to automatically transition data to lower-cost storage tiers as it ages, remove outdated data, or archive data to Glacier for long-term storage.

These are the Steps involved in the base paper. They utilized Microsoft Azure ADLS Gen2 for their data lake, whereas we opted for AWS S3, which offers equivalent features.

CHAPTER 2

MERITS AND DEMERITS

The paper discusses various existing techniques for storing webserver access log files:

- "ADLS Gen2 for web server log data analysis" by Elisabeta Zagan, Mirela Danubianu. In this paper they used ALDS Gen2 for storing log files but there is no data reduction transforming process involved
- "Web Access Log Processing Method Based on Storm" by Zhang Khan, the authors detail a methodology comprising four stages: log collection, log reception, log processing, and log storage. However, the paper notes that storing log files solely in a database is ineffective for analysis purposes

Merits:

- **Data lake architecture:** The proposed data lake architecture provides an economical, secure, and cost-effective solution for storing large quantities of raw log files
- **Size reduction:** The triggering function results in a 90% reduction in storage space compared to the original size of log files, leading to lower costs
- **Cost reduction:** The proposed model, with its data lifecycle policy, contributes to cost reduction by transitioning data to the suitable storage tiers

Demerits:

- **Complexity:** Implementing and managing a data lake can be complex, requiring expertise in data integration and maintenance. The sheer volume and variety of data stored in a data lake can lead to challenges in organization and management
- **Lambda:** Lambda triggering function is suitable for small files. As our dataset is large we cannot use lambda function for triggering S3 bucket. Instead we used third party Google colab for triggering the S3 bucket. It may arise some security concerns.
- **Cost considerations:** While data lakes offer potential cost savings compared to traditional data warehousing approaches, they can also incur significant costs in terms of infrastructure, storage, and maintenance. If we don't maintain the architecture correctly, it may result in significant financial losses

CHAPTER 3

Source Code

#AWS

#Creating IAM user in aws using linux cli

```
aws iam create-user --user-name sakthi
```

#Setting password for user

```
aws iam create-login-profile --user-name sakthi --password Saks4321#
```

#adding user to group in aws

```
aws iam add-user-to-group --user-name-sakthi --group-name mini-project
```

#listing users in group

```
aws iam get-group --group-name mini-project
```

#IAM JSON Policy attached to the users on aws

#ADMINISTRATOR ACCESS POLICY

#User have administator access and can control all other users

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

#S3 READ ONLY ACCESS POLICY

#User can only able to read the S3 buckets

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
        "s3:Get*",
        "s3:List*",
        "s3:Describe*",
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
    ],
    "Resource": "*"
}
]
}

```

#SPLITTING LOG FILES

#splitting log files into smaller chunks using linux cli
split -b 1048576000 access.log --additional-suffix=.log

#to view the splitted files
ll

#to view the no.of.lines the splitted files
wc -l xaa.log xab.log xac.log xad.log

#Triggering function for converting log files into parquet files

```

!pip install boto3
import boto3
import pandas as pd
import re
import io

```

```

regex_pattern = r'^(?P<ClientIP>\S+) (?P<RemoteLogName>\S+) (?P<AuthUserName>\S+)
\[ (?P<TimeStamp>[^\]]+)\] "(?P<AccessMethod>[A-Z]+) (?P<AccessRequest>.+?) HTTP/[0-
9.]+)" (?P<ResultStatus>[0-9]{3}) (?P<SizeBytes>[0-9]+|-) "(?P<ReferrerURL>[^\"]*)"
"(?P<UserAgent>[^\"]*)"'
columns = ['ClientIP', 'RemoteLogName', 'AuthUserName', 'TimeStamp', 'AccessMethod',
'AccessRequest', 'ResultStatus', 'SizeBytes', 'ReferrerURL', 'UserAgent']

```

```

aws_access_key_id = "AKIAV4XM5JQIET3AR77T"
aws_secret_access_key = "ccZUhmV4KFC5Ljfd1O3imy4kOKdqMmnsyIleScYxz"

```

```

input_bucket = "projectminilog-1"
output_bucket_parquet = "outputbucketmini"
output_bucket_error = "erroroutputmini"

s3 = boto3.client('s3', aws_access_key_id=aws_access_key_id,
aws_secret_access_key=aws_secret_access_key)

def process_log_files(log_files):
    successful_files = []
    unsuccessful_files = []

    for log_file in log_files:
        try:
            obj = s3.get_object(Bucket=input_bucket, Key=log_file)
            log_data = obj['Body'].read().decode('utf-8')

            if log_data.strip():
                matches = re.findall(regex_pattern, log_data)
                if matches:
                    df = pd.DataFrame(matches, columns=columns)
                    parquet_buffer = io.BytesIO()
                    df.to_parquet(parquet_buffer, index=False)
                    output_key = log_file.replace('.log', '.parquet').replace("logfiles/", "")
                    s3.put_object(Body=parquet_buffer.getvalue(), Bucket=output_bucket_parquet,
Key=output_key)
                    successful_files.append(log_file)
                else:
                    unsuccessful_files.append(log_file)
            else:
                unsuccessful_files.append(log_file)
                print(f"Empty file: {log_file}")
        except Exception as e:
            unsuccessful_files.append(log_file)
            print(f"Error processing file {log_file}: {str(e)}")

    return successful_files, unsuccessful_files

def main():
    response = s3.list_objects_v2(Bucket=input_bucket, Prefix="logfiles/")
    log_files = [obj['Key'] for obj in response.get('Contents', []) if obj['Key'].endswith('.log')]
    successful_files, unsuccessful_files = process_log_files(log_files)

```

```
if unsuccessful_files:
    error_file_content = '\n'.join(unsuccessful_files)
    s3.put_object(Body=error_file_content.encode(), Bucket=output_bucket_error,
Key='error_files.txt')

print("parquet uploaded successfully")

# Execute the main function
main()
```

CHAPTER 4

Snapshots

```
jessepinkmann@jessepinkmann:~$  
jessepinkmann@jessepinkmann:~$ aws iam create-user --user-name sakthi  
{  
  "User": {  
    "Path": "/",  
    "UserName": "sakthi",  
    "UserId": "AIDAV4XM5JQIOUS32XZHC",  
    "Arn": "arn:aws:iam::405297515536:user/sakthi",  
    "CreateDate": "2024-03-04T10:58:58+00:00"  
  }  
}  
jessepinkmann@jessepinkmann:~$ aws iam create-login-profile --user-name sakthi --password Sakthi2004#  
{  
  "LoginProfile": {  
    "UserName": "sakthi",  
    "CreateDate": "2024-03-04T11:00:17+00:00",  
    "PasswordResetRequired": false  
  }  
}  
jessepinkmann@jessepinkmann:~$ aws iam add-user-to-group --user-name sakthi --group-name mini-project  
jessepinkmann@jessepinkmann:~$ aws iam add-user-to-group --user-name sakthi --group-name mini-project  
jessepinkmann@jessepinkmann:~$
```

Fig 4.1 Creating IAM user using cli and setting password

```
jessepinkmann@jessepinkmann:~$ aws iam get-group --group-name mini-project  
{  
  "Users": [  
    {  
      "Path": "/",  
      "UserName": "renga",  
      "UserId": "AIDAV4XM5JQIKF3ZNDK70",  
      "Arn": "arn:aws:iam::405297515536:user/renga",  
      "CreateDate": "2023-12-16T05:14:12+00:00",  
      "PasswordLastUsed": "2024-03-04T10:50:00+00:00"  
    },  
    {  
      "Path": "/",  
      "UserName": "raghuram",  
      "UserId": "AIDAV4XM5JQINBDZBCZLW",  
      "Arn": "arn:aws:iam::405297515536:user/raghuram",  
      "CreateDate": "2024-03-04T10:16:17+00:00"  
    },  
    {  
      "Path": "/",  
      "UserName": "sakthi",  
      "UserId": "AIDAV4XM5JQIOUS32XZHC",  
      "Arn": "arn:aws:iam::405297515536:user/sakthi",  
      "CreateDate": "2024-03-04T10:58:58+00:00"  
    }  
  ],  
  "Group": {  
    "Path": "/",  
    "GroupName": "mini-project",  
    "GroupId": "AGPAV4XM5JQID35DCFOHY",  
    "Arn": "arn:aws:iam::405297515536:group/mini-project",  
    "CreateDate": "2024-03-04T10:16:03+00:00"  
  }  
}
```

Fig 4.2 Listing IAM users present in group using cli


```
jessepinkmann@jessepinkmann:~/Desktop/mini project/log files$ split -b 1048576000 access.log --additional-suffix=.log
jessepinkmann@jessepinkmann:~/Desktop/mini project/log files$ ll
total 6840736
drwxrwxr-x 2 jessepinkmann jessepinkmann 4096 Mar 4 21:53 ./
drwxrwxr-x 3 jessepinkmann jessepinkmann 4096 Mar 4 21:39 ../
-rw-rw-r-- 1 jessepinkmann jessepinkmann 3502440823 Feb 13 2021 access.log
-rw-rw-r-- 1 jessepinkmann jessepinkmann 1048576000 Mar 4 21:53 xaa.log
-rw-rw-r-- 1 jessepinkmann jessepinkmann 1048576000 Mar 4 21:53 xab.log
-rw-rw-r-- 1 jessepinkmann jessepinkmann 1048576000 Mar 4 21:53 xac.log
-rw-rw-r-- 1 jessepinkmann jessepinkmann 356712823 Mar 4 21:53 xad.log
```

Fig 4.3 Splitting log files into smaller chunks and viewing them

```
jessepinkmann@jessepinkmann:~/Desktop/mini project/log files$ wc -l access.log
10365152 access.log
jessepinkmann@jessepinkmann:~/Desktop/mini project/log files$ wc -l xaa.log xab.log xac.log xad.log
3170023 xaa.log
3037404 xab.log
3203637 xac.log
954088 xad.log
10365152 total
```

Fig 4.4 Viewing no.of lines present in each log files which are seperated to verify whether it matches with the dataset

Log File (Name)	LOG (BYTES)	LOG LINES (NUMBER OF LINES)	Transformation Time (s)	DL log (MiB)	DL parquet (MiB)	DL err txt (KiB)
access-weblog1.log	1048576000	3170024	76.95	1000.00	108.39	102.02
access-weblog2.log	1048576000	3037405	75.51	1000.00	111.2	21.99
access-weblog3.log	1048576000	3203638	107.60	1000.00	116.06	18.78
access-weblog4.log	356712823	954088	17.86	340.19	31.26	8.35
Total	3502440823	10365155	277.92	3340.19	366.91	151.14

Fig 4.5 Results from the base paper





<input type="checkbox"/>	Name ▾	Type ▾	Size ▾	Storage class ▾
<input type="checkbox"/>	 xaa.log	log	1000.0 MB	Standard
<input type="checkbox"/>	 xab.log	log	1000.0 MB	Standard
<input type="checkbox"/>	 xac.log	log	1000.0 MB	Standard
<input type="checkbox"/>	 xad.log	log	340.2 MB	Standard

Fig 4.6 Log files stored in the projectminilog-1 S3 bucket





<input type="checkbox"/>	Name ▲	Type ▾	Size ▾
<input type="checkbox"/>	 xaa.parquet	parquet	90.8 MB
<input type="checkbox"/>	 xab.parquet	parquet	91.3 MB
<input type="checkbox"/>	 xac.parquet	parquet	92.6 MB
<input type="checkbox"/>	 xad.parquet	parquet	31.3 MB

Fig 4.7 Transformed Parquet files stored in outputbucketmini S3 bucket





<input type="checkbox"/>	Name ▲	Type ▾	Size ▾
<input type="checkbox"/>	 xaa.txt	txt	101.1 KB
<input type="checkbox"/>	 xab.txt	txt	21.2 KB
<input type="checkbox"/>	 xac.txt	txt	18.4 KB
<input type="checkbox"/>	 xad.txt	txt	8.3 KB

Fig 4.8 Error text files stored in erroroutputmini S3 bucket

Current version actions

Day 0

- Objects uploaded



Day 30

- Objects move to Standard-IA



Day 60

- Objects move to One Zone-IA



Day 90

- Objects move to Glacier Flexible Retrieval (formerly Glacier)



Day 180

- Objects move to Glacier Deep Archive

Fig 4.9 Datalife cycle management policy

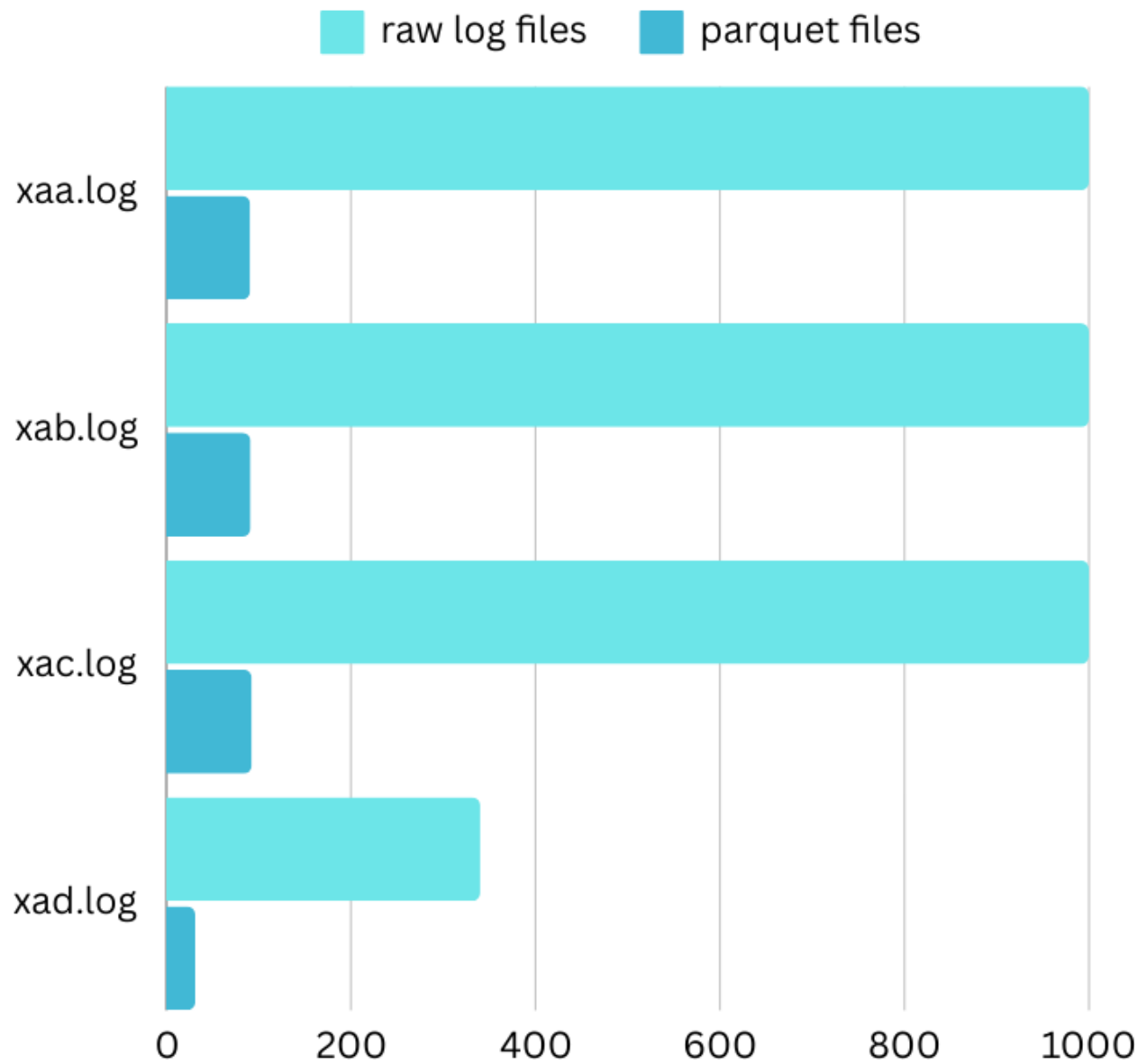


Fig 4.10 Storage space reduction achieved by transforming log files into parquet files

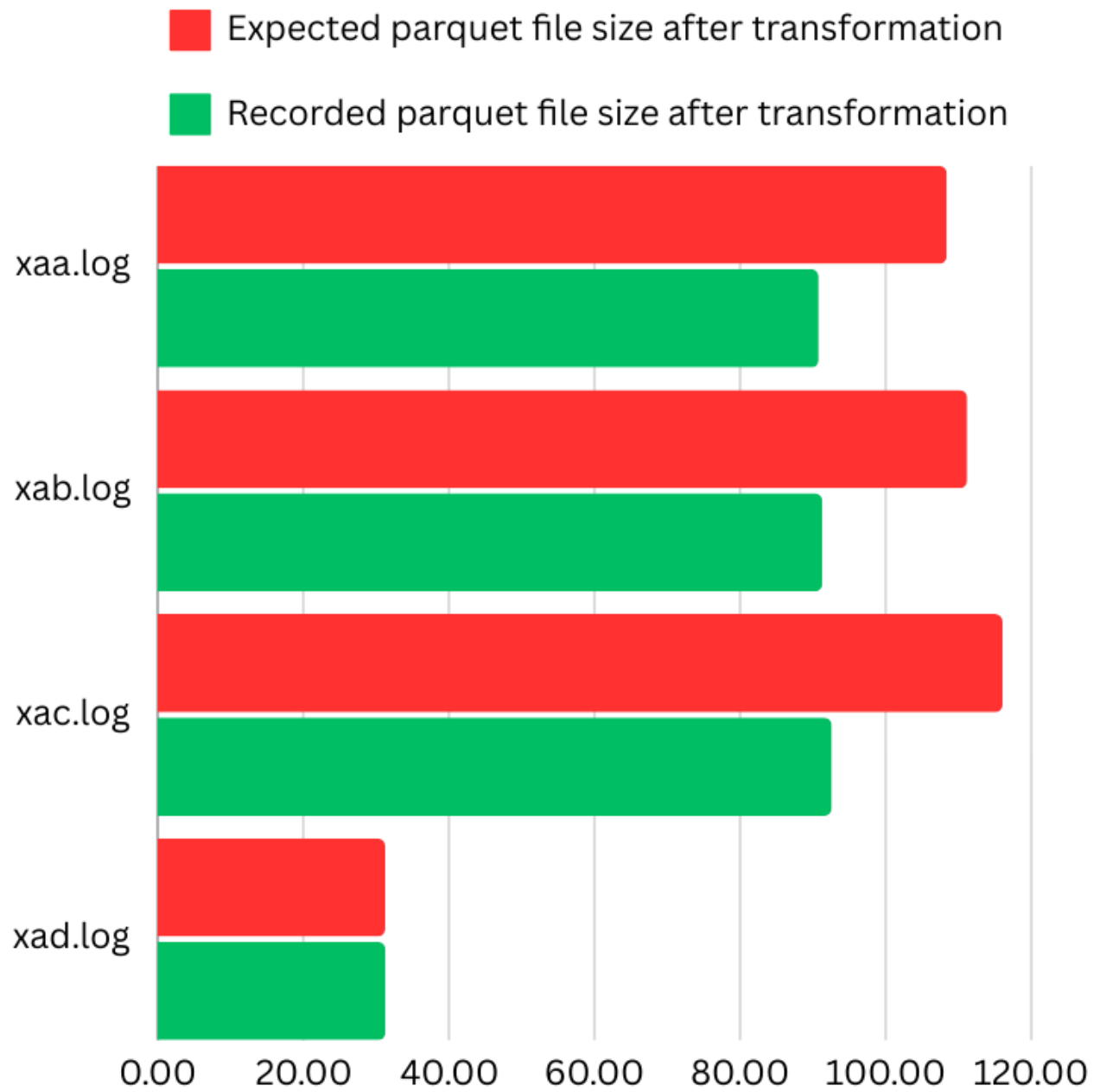


Fig 4.11 Comparing expected and recorded outputs

CHAPTER 5

Conclusion and Future Plans

5.1 Conclusion:

This project underscores the significance of leveraging advanced cloud-based data lake (DL) technology for storing vast amounts of unstructured data. Such technology provides comprehensive services for data ingestion, storage, processing, and analysis. A key advantage of storing data in S3 is its support for hierarchical data storage, offering fine-grained access control. Subsequently, we successfully converted log files into Parquet files, resulting in a 90% reduction in size. Additionally, the implementation of a data lifecycle policy promises significant cost savings. This represents one instance of data lake architecture, specifically implemented within AWS. Similarly, similar data lake architectures can be established across different public cloud providers like Microsoft Azure and Google Cloud.

5.2 Future Plans:

We have effectively transformed log files into Parquet files, utilizing a columnar format that simplifies analysis and decreases query time. This format minimizes redundant queries during analysis. At present, the Parquet files reside in S3 buckets, enabling seamless integration with AWS analytics services like Amazon Redshift, Amazon Athena, Amazon EMR, and AWS Glue, facilitating a wide range of analytical operations.

CHAPTER 6

References

1. **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
2. **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
3. **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
4. **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
5. **A. A. Munshi and Y. A.-R.-I. Mohamed**, “*Data lake lambda architecture for smart grids big data analytics*,” IEEE Access, vol. 6, pp. 40463–40471, 2018
6. **H. Fang**, “*Managing data lakes in big data era: What’s a data lake and why has it become popular in data management ecosystem*,” in Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER), Shenyang, China, Jun. 2015, pp. 820–82
7. **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
8. **S. Ramchand and T. Mahmood**, “*Big data architectures for data lakes: A systematic literature review*,” in Proc. IEEE 46th Annu. Comput., Softw., Appl. Conf. (COMPSAC), Los Alamitos, CA, USA, Jun. 2022, pp. 1141–1146.

CHAPTER 7

Appendix

Base Paper

Elisabeta Zagan and Mirela Danubianu“Data Lake Architecture for Storing and Transforming Web Server Access Log Files ”

Keywords: Data Lake,Log Files

doi: 10.1109/ACCESS.2023.3270368

URL: <https://ieeexplore.ieee.org/document/10107911>