IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-15, May 2019 E-ISSN: 2347-2693

An Efficient Implementation of Distributed Storage Protocol for Large Number of Video Streams

[1*]Amrutha B V, [2]Ananya Jeevan, [3]Bhumika R, [4]Chaitra P, [5]Dhanraj S

[1,2,3,4,5]Dept.of Computer Science ,East West Institute of technology, Vishveswaraya Technological University, Bangalore, India

DOI: https://doi.org/10.26438/ijcse/v7si15.1115 | Available online at: www.ijcseonline.org

Abstract—An ever-increasing number of installed surveillance cameras, higher network bandwidth requirements and larger storage space consumption are major considerations when designing a video surveillance storage system. Traditional file systems are not tailored for receiving and storage of large scale video streams. This project attempts to present vsStor, a PB-scale, network-based video surveillance storage system which is used for storing thousands of streams. Another major characteristics of vsStor is the scale-out architecture design which allows the performance to grow by adding more hardware. We make use of clustered architecture for connecting various machines and rely on the web service mechanism for communication between them.

Keywords—Video surveillance, storage system,Distributed file system, Big Data

I. INTRODUCTION

Traditional video surveillance systems require infrastructures including expensive servers with capabilities to process images and store video recordings. These surveillance systems produce and need to store a huge amount of data and to execute on them specific image analysis in real-time in order to detect safety events. We propose a video surveillance system based on Big Data that collects multimedia streams generated by surveillance cameras, optimizes their transmissions according to network condition and stores them in a storage system in an efficient and secure way. Due to limited size of bandwidth the streamed videos from the surveillance cameras might get lost or certain bits of data might go missing. In addition to surveillance, video cameras are now being routinely used in ambient-assisted living applications, in which ensuring visual privacy is also a critical concern. Due to difficulties of storing and analyzing this huge amount of multimedia data in local servers, deferring these tasks to the cloud servers has gained popularity. However, this further exacerbates privacy concerns as such data can also be acquired by unauthorized parties.

High network bandwidth requirement and larger storage space consumption are the major considerations while designing a video surveillance storage system. This project presents vsStor, a large-scale video surveillance storage system which is used for receiving and storing thousands of concurrent video streams. vsStor is a not traditional (i.e. POSIX-compliant) file system, therefore, it's not designed to provide normal file services for existing and legacy applications. Instead, vsStor runs on traditional file systems, so it is easier to implement and less error-prone.
The main contributions of this work are: 1) Design and implement a video storage system which can handle thousands of video streams concurrently. 2) Storage of metadata in database and on disks allows the system to scale to PB-level which is crucial in Big Data environment

We need an efficient solution to store the large volume of surveillance videos without compromising the network bandwidth efficiency. The main objective of this project is to 1) Design and implement vsStor, a large-scale video surveillance storage system which is used for receiving and storing thousands of video streams. 2) Provide ability for the users to scale the system horizontally when needed. 3) Provide an efficient means for accessing the stored media across the clustered environment. 4) Provide an ability to the users to know the health information of each of the machine in the cluster. 5) provide a means to the user to enable or disable the machine in the cluster at any instance of time.

vsStor is built on commodity hardware and consists of multiple nodes acted as clients and/or servers where we will be developing a web based Java/J2EE application. This application will be installed on a given server (say computer A1), the end users can access the application through a web browser from any number of different machines (say computer B1, B2, C1, C2 etc.) and login using register page. After logging in using the credentials provided by the user further process can be carried out where in nodes can be connected and data can be transferred for storage purpose using the concept of Distributed File System. The scope of this project is to provide horizontally scalable storage space

which means storage space literally won't run out of memory, network bandwidth not compromised, confidential data isn't exposed to any public cloud, can be scalable to very huge data sets, to design and implement vsStor, a large-scale video surveillance storage system which is used for receiving and storing thousands of video streams, to provide an ability for the users to scale the system horizontally when needed, to provide an efficient means for accessing the stored media across the clustered environment, to provide an ability to the users to know the health information of each of the machine in the cluster, to provide a means to the user to enable or disable the machine in the cluster at any instance of time.

II. RELATED WORK

According to the literature, Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.[citation needed] The UML has become the standard language in object-oriented analysis and design.[citation needed] It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.[citation needed]

A. vsNode and vsSTOR

In this portal, we implement the vsNode application. Its an independent application which exposes numerous web-services which will be consumed by the vsStor application to perform various operations. vsNode essentially runs on the hardware where the video streams will be persisted. This vsNode application can be deployed on as many nodes (machines) as needed. More the number of nodes, more the storage space. To be more specific, the vsNode application exposes the following web-services

- Upload: This web service is used by VsSTOR application to upload a new video stream into the vsNode. The video stream must be sent as a form data inside the request body.
- Download: This web service is used by vsStor application to download a particular video stream. The filename to be downloaded must be sent as a request parameter to this web service.
- List Files: This web service will return the JSON data which includes the list of all the file names along with their size consumed and the timestamp at which the file was uploaded.
- Delete: This web service accepts the filename as the request parameter and deletes that particular file from its file system.
- Space Available: This web service returns the total space available in its file system. The unit here will be bytes.
- Space Consumed: This web service returns the total amount of space consumed by all the files in its filesystem. The unit here will be bytes.
- Heart beat: This web service is used by the vsStor application to check the health information of the vsNode. If this web service is reachable and acknowledges, then the node is considered as Healthy, else the node will be considered as Broken

B. CANDELA–storage and analysis

Video management research has largely been ignoring the increased attractiveness of using camera-equipped mobile phones for the production of short home video clips, mostly considering them as additional channels for video consumption. The CANDELA project, which is part of the European ITEA program, focuses on the integration of video content analysis with advanced retrieval, mobile, networked delivery, and distributed storage technologies. In this paper, we present the CANDELA personal mobile multimedia management platform, which implements an end-to-end system for personal video production, retrieval, and consumption utilizing mobile devices and distributed database.

C. Dynamic partial-parallel data layout

Video surveillance requires storing massive video data, which results in the rapid increasing of storage energy consumption. With the popularization of video surveillance, the green storage for video surveillance is very attractive. The existed energy-saving methods for massive storage most concentrate on the data centers mainly with random access, while the storage of video surveillance has inherent workload characteristics and access pattern, which can be fully exploited to save more energy. A dynamic partial-parallel data layout (DPPDL) is proposed for green video surveillance storage. It adopts dynamic partial-parallel strategy, which dynamically allocates the storage space with appropriate degree of partial parallelism according to performance requirement. Partial parallelism benefits energy conservation by scheduling only partial disks to work; dynamic degree of parallelism can provide appropriate performances for various intensity workloads. DPPDL is evaluated by a simulated video surveillance consisting of 60 to 300 cameras with 1920×1080 pixels. The experiment shows DPPDL is most energy efficient, while tolerating single disk failure and providing more than 20% performance margin. On average, it saves 7%, 19%, 31%, 36%, 56%, and 59% energy than Cache RAID, Semi-RAID, Hibernator, MAID, eRAID5, and PARAID respectively.

D. Hadoop distributed file system

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed

to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject.

III. METHODOLOGY

vsStor segmentation is described in this section, which consists of five major modules: the data access layer, the configuration and health check, the data upload, the data access, the statistical report.

A. System Architecture

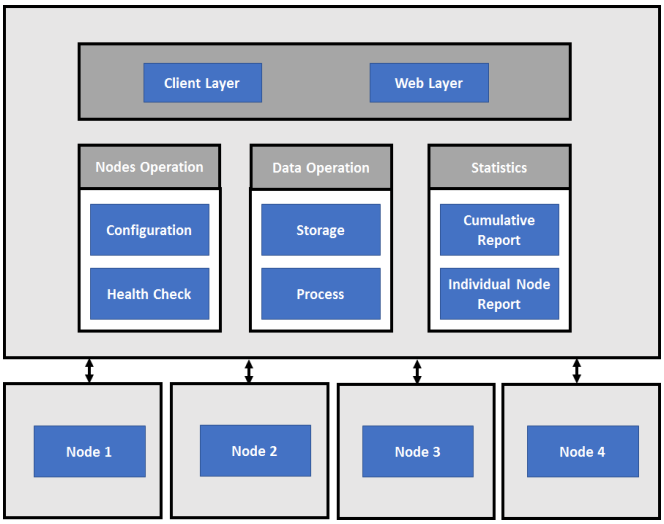

Figure 1: Block Diagram of Activities Performed

vsSTOR, a large-scale video surveillance storage system which is used for receiving and storing thousands of videos stream. The design will contain the specification of all these modules, their interaction with other modules and the desired output from each module.
The design phase is followed by two sub phases:

- High Level Design
- Low Level Design

B. Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Util's as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

1. Account operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

2. vsNode

In this portal, we implement the vsNode application. It's an independent application which exposes numerous web-services which will be consumed by the vsStor application to perform various operations. vsNode essentially runs on the hardware where the video streams will be persisted. This vsNode application can be deployed on as many nodes (machines) as needed. More the number of nodes, more the storage space.

C. Configuration and Health Check

This module allows the end users to add or remove as many vsNodes as they want from the system. The users will have to provide the details of the vsNode machine to add it to the system. The details include the host name, the port number where the vsNode application server is running and its context root to access the web services exposed by it. The system then stores this information in the database so that this new node can be considered while uploading the video streams.
This module also provides a feature to the end users where they can check the health information of all the nodes added so far in the system. The vsStor application will invoke the REST API health check exposed by the vsNode application. If the API acknowledges the request, the node will be considered as reachable, else the node will be considered as Broken.

D. Video Upload

This portal is used to upload a new video streams to the system. When this module is requested by the end users, the system will invoke the space available REST API on all the vsNodes configured so far to find out the best candidate to store the new video streams. The best candidate is simply the node with the maximum space available among all the configured nodes. The user will then be provided with an HTML interface where they can browse the video stream and upon invoking the request, the vsStor application will send

this video stream in the request body (multipart form data) to the upload REST API exposed by the target vsNode.
However, the acknowledgement won't be received because the video stream uploading would take significantly more time depending upon the size of the input stream. Hence the user will be notified about the same in the HTML interface.

E. Upload Access

In this module, the users will be able to access the video streams uploaded to the vsNodes. By accessing means, the users can perform the read, download, and delete operations on all the video streams uploaded by them to the vsNode machines. The system will show an HTML interface to the users where they can visualize the list of al the files available in the vsNodes. The user can then select any of the stream and download it or perform the delete operation in case the stream is no longer required to be saved on the vsNodes.
The vsStor application will be making use of 'list files', 'download', and 'delete' REST web services exposed by the vsNode applications to provide these features to the end users.

F. Statistical Report

This module is helpful in analysing the state of the system in terms of the storage space consumed and storage space available. This module provides an HTML interface to the end users where they can visualize the report of the system.
This module presents two types of reports for the users, namely, cumulative reports and individual reports.
The cumulative report is simply a summation of all the metrics from the individual reports which provides the information about the space consumed and space available on each of the nodes.

IV. RESULTS AND DISCUSSION

The major contribution of how the data node and the system depend upon the final statistical report of the storage. The below table shows the operations for the storage of the nodes.

TABLE 1: VARIOUS OPERATIONS

Account Operations	User must be creating a new account in our portal to get access to the rest of the modules. The user can also perform various other account operations like retrieving the forgotten password, login, logout, delete profile, change password, and edit profile
Configuration	Here the end users can add and remove the nodes from the system.
Health Check	Here the end users can perform the health check of the nodes configured in the system
Video Upload	Here the end users can perform the video upload operation into the configured nodes
Video Access	Here the end users can access the stored videos and even delete them in case they are no longer
Reporting	Here the end users can visualize the statistical information about the system.

V. CONCLUSION AND FUTURE SCOPE

In this project, we proposed the design and implementation of vsStor, a web service based storage system used for storing large scale surveillance data. vsStor is capable of simultaneously receiving and storing thousands of video streams. Another major characteristics of vsStor is the scale-out architecture design which allows the performance to grow by adding more hardware. In future, we work towards extending our solution to perform various types of video analytics on the stored surveillance data.

A. Abbreviations and Acronyms

The various words and there definitions which are used this paper are as follows:

- **vsStor application**: Ability for the end users to modify/update the configuration information provided.
- **VsNodes:** Enabling the end users to perform the Health check on all the vsNode applications
- **Configured:** Enabling the end users to upload the video stream to the target vsNode which have the maximum storage space at the time of uploading.

B. Equations

The capacity of the storage node is the major concert in which we need to find the path of the storage using the following code:

File file = new File(Constants.STORAGE_PATH+ File.separator + f); (1)

With the help of the above formulation we can find the storage path.

result.put(f, String.valueOf(file.length() / 1000)); (2)

The maximum space capcacity depends upon

MAXIMUM_SPACE_IN_KB = "1000000"; (3)

Usually the space size is generally represented in the KiloBytes

C. Authors and Affiliation

Amrutha B V, BE, department of Computer Science and Engineering, East West Institute of Technology.
Ananya Jeevan, BE, department of Computer Science and Engineering, East West Institute of Technology
Bhumika R, BE, department of Computer Science and Engineering, East West Institute of Technology
Chaitra P, department of Computer Science and Engineering, East West Institute of Technology
Assistant Professor Dhanraj S , BE, MTech, department of Computer Science and Engineering, East West Institute of Technology

ACKNOWLEDGMENT

We would like to thank our principal Dr. K Chennakeshavalu and head of department Dr. Arun Biradar , East West Institute of Technology, Computer Science and Engineering for supporting us to carry out our project with clear guidance from Assistant Professor Mr Dhanraj S .

REFERENCES

[1] Video Surveillance Market worth 75.64 Billion USD by 2022.MarketsandMarkets,2017, https://www.marketsandmarkets.com/PressReleases/global-videosurveillance- market.asp.
[2] D. Rodriguez-Silva, L. Adkinson-Orellana, F. Gonz'lez-Castano, I. Armino-Franco and D. Gonz'lez-Martinez, "Video Surveillance Based on Cloud Storage," 2012 IEEE Fifth International Conference on Cloud Computing(CLOUD), IEEE, Jun. 2012, pp. 991-992, doi:10.1109/CLOUD.2012.44.
[3] M.R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?," Proc. 2008 international workshop on Data-aware distributed computing, ACM, Jun. 2008, pp. 55-64, doi:10.1145/1383519.1383526.
[4] S. Wang, J. Yang, Y. Zhao, A. Cai, and S.Z. Li, "A surveillance video analysis and storage scheme for scalable synopsis browsing," In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, Nov. 2011, pp. 1947-1954, doi: 10.1109/ICCVW.2011.6130487.
[5] E. Jaspers, R. Wijnhoven, R. Albers, J. Nesvadba, J. Lukkien, A. Sinitsyn, et al., "CANDELA–storage, analysis and retrieval of video content in distributed systems," In International Workshop on Adaptive Multimedia Retrieval, Springer, 2005, pp. 112-127. doi: 10.1007/11670834_10.
[6] Z. Sun, Q. Zhang, Y. Li, and Y. Tan, "Dppdl: a dynamic partialparallel data layout for green video surveillance storage," IEEE Transactions on Circuits and Systems for Video Technology, IEEE, 2016, doi: 10.1109/TCSVT.2016.2605045.
[7] P. Pillai, Y. Ke, and J. Campbell, "Multi-fidelity storage," Proc. ACM 2nd international workshop on Video surveillance & sensor networks, ACM, Oct. 2004, pp. 72-79, doi: 10.1145/1026799.1026812.
[8] T. Thomas, A.K. Rajan, T. Johnson, S. Anjana, and A. Bithin, "Policy based storage abstraction for video surveillance systems," In Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on, IEEE, Dec. 2016, pp. 1-4, doi: 10.1109/ICCIC.2016.7919565.
[9] T. Zhang, A. Chowdhery, P.V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," Proc. 21st Annual International Conference on Mobile Computing and Networking, ACM, Sep. 2015, pp. 426-438, doi: 10.1145/2789168.2790123.
[10] D. Borthakur, "HDFS architecture guide," Hadoop Apache Project 53 (2008).
[11] Smart, Safe, Secure Surveillance Hard Drives Data Sheet. Seagate, 2016, http://www.seagate.com/www-content/productcontent/ skyhawk/files/skyhawk-ds-1902-3-1608us.pdf.
[12] J. Tai, D. Liu, Z. Yang, X. Zhu, J. Lo, and N. Mi, "Improving flash resource utilization at minimal management cost in virtualized flashbased storage systems," IEEE Transactions on Cloud Computing 5(3), IEEE, Jul. 2017, pp. 537-549, doi:10.1109/TCC.2015.2424886.

