Scaling the Cloud for Smart Home Security: Efficient Video Data Deletion with MapReduce

Sibin Thomas

Tech Lead
sibin_thomas15@hotmail.com

Abstract:
Home security cameras create a lot of video footage, making it tough to keep track of what to save and what to delete. This paper looks at the challenges of handling a lot of videos every day. It focuses on how videos are deleted based on time, retention, and user choice, all while keeping user privacy and costs in mind. MapReduce programming model can be leveraged to provide a solution that can scale and adapt easily and help with data cleaning and managing access. Using smart optimizations such as metadata dumps and a "should serve" marker further improves efficiency and enhances scalability. This method helps smart home security cameras follow data retention rules, save on storage costs, and offer a safer and improved experience for users.

Keywords: Video data management, Data retention, Data deletion, MapReduce, Scalability, Cloud storage, User privacy, Cost optimization, Big data

INTRODUCTION

More and more smart home security cameras are popping up, which means there's a lot more video data being created. Lots of cameras are always on, recording and sending videos all the time. This makes it really tough for service providers to handle billions of videos every day. It's important to have good ways to take in and store data. It it important to manage video data properly, this includes the storage of video and purging video effectively. This is important for protecting user privacy, reducing storage costs, and ensuring the system works smoothly. Old ways of deleting data have a hard time managing the size and complexity of these systems. Checking a database with billions of records to find and remove old videos takes a lot of time and resources. Dealing with user deletions and making sure users can't access media right after it's deleted makes things a bit more complicated. This paper looks at the problems of keeping and deleting data in smart home security camera systems and suggests a scalable solution using the MapReduce programming model [2]. MapReduce can be used to carry out deletions based on time, retention, and user requests on a large scale while also keeping data private and being cost-effective. It shows how some optimization methods, like using metadata dumps and a "should serve" marker, can improve the deletion process to make it cheaper and easier to scale. Home security camera systems can handle video data well by using the suggested solution. This also helps meet certain data retention policies and gives users a safe and responsive experience.

ARCHITECTURE AND WORKFLOW OF A SMART HOME SECURITY CAMERA SYSTEM

This part gives a clear look at how a regular smart home security camera system is set up and how it works. It looks at the main parts that are important for handling and sharing media. We'll look at how the different parts work together to make video streaming, event detection, and user notifications happen—from when the camera catches an event to when it plays back on a user's device.

System Components

One big challenge in creating a smart home security camera system is dealing with the huge amounts of video data that millions of cameras produce around the world. These cameras keep sending video streams, usually when they detect motion or sound, leading to millions of uploads every second. To handle a lot of data well, you need a robust and flexible setup for taking in, processing, and storing videos efficiently.

Video Ingestion Workflow

A video ingestion workflow in a smart home security camera system involves the following steps:

Event Detection and Recording: The camera detects something happening, like movement or noise, and starts recording video on the device.

On-Device Chunking: The video that's captured gets turned into smaller pieces right on the device. This chunking process can be based on time (like x-second chunks) or size (like yMB chunks).

Upload to Cloud: The video fragments are uploaded to Cloud.

Authorization: The Cloud performs authorization checks to verify if the camera is authorized to upload video data. This is to ensure that only authorized devices can contribute to the media storage.

Storage: Once authorized, the video data is stored in a distributed storage system [3].

Media Storage Architecture

The media storage architecture typically consists of two main components:

Metadata Database: The camera ID, the start and end times, the type of event, and the location of the video are all stored in the database. For the video files saved in the object storage, there is also a link. This database is made so that metadata is easy to get to and can be either a normal SQL database or a NoSQL database.

Object Storage: The video clips are kept in a storage system designed to handle and access large amounts of unstructured data efficiently. This system should ideally be highly available, reliable, cost-effective and secure.

Video Serving and Playback

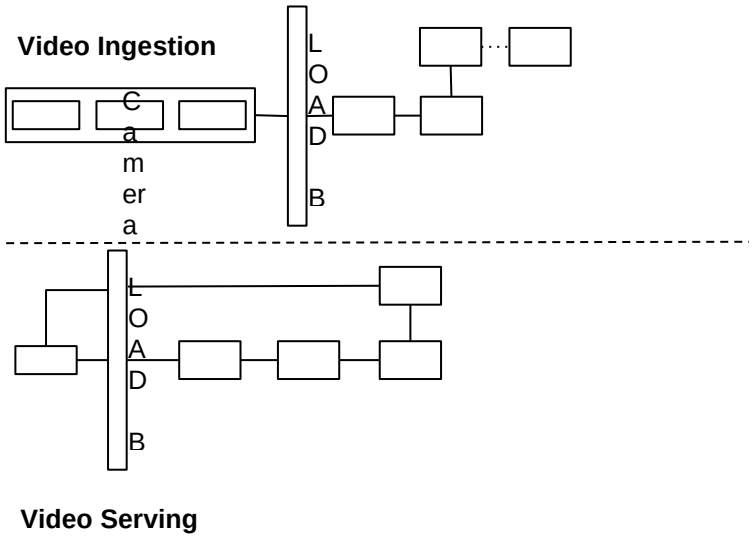
When a user requests to view a video, the following steps occurs:

Manifest Generation: The Adaptive Streaming Service creates a list that shows the URLs of the video segments needed for playback during a certain time. The metadata database helps find where the video chunks are stored, which is necessary for the manifest.

Segment Download: The app downloads the video segments from the storage using the URLs in the manifest.

Video Playback: The app's video player puts together and plays the downloaded video clips.

As discussed above, it is important to separate the metadata and video data storage for optimal performance and is more scalable.



CHALLENGES OF DATA RETENTION AND DELETION AT SCALE

Handling video data from start to finish for a smart home security camera system means taking care of how we take in and store the data, as well as having good plans for keeping or deleting it when needed. There are millions of cameras sending video data every second, leading to billions of videos saved in the cloud each day. Ensuring that video data gets deleted on time and with no dangling pointers to videos is a big challenge. There are two main challenges when it comes to purging video data in large amounts.

Time-Based and Retention-Based Deletion

Security camera services usually have different subscription plans, and the higher plan lets you retain videos for a longer period. So users can access their videos longer. Once the retention period has expired, it's important to purge expired video data. This process of deleting based on time and retention is important for a few reasons:

Customer Privacy: Getting rid of old video data helps keep user privacy safe by making sure personal videos aren't stored and accessible. So users can have trust that their privacy is maintained.

Storage Cost Optimization: Purging expired videos helps clear up storage space and lowers the costs of storing and managing outdated data in the cloud.

Database Efficiency: Getting rid of extra data in the metadata database makes queries run faster and cuts down on indexing work.

With so many cameras uploading video content, it’s likely that billions of videos need to be deleted every day to keep things running smoothly and follow data retention rules. We need strong and automatic ways to find and remove old video data from the system.

User-Initiated Deletion

The second challenge is when users want to delete their video data, whether it's for certain cameras or for their whole account. This deletion request should ensure that once user gets a successful response then the video should not be accessible. This also means eventually purging tons of video data, which could be months or years' worth, totaling billions of data points.

The main challenge is to make sure that during the deletion process, no video data set for deletion is shown to anyone, even the person who owns it. This is important for keeping user privacy safe and following data protection rules.

To tackle this issue, a solid approach is to use storage tags or markers to point out data that needs to be deleted. The system can block access to the data right away, even before it's actually taken out of storage. This makes sure that user privacy is kept safe during the deletion process.

Dealing with data deletion issues is really important for keeping user trust and making the most of storage costs. It's important to make sure the smart home security camera system works well and can grow as needed.

EFFICIENT DATA DELETION AT SCALE WITH MAPREDUCE

Handling the storage and removal of billions of videos created every day by smart home security cameras is quite a challenge. This part suggests a solution using the MapReduce programming model to tackle these issues in a way that is both affordable and dependable.

MapReduce for Time-Based and Retention-Based Deletion MapReduce works really well for handling big datasets at the same time across a group of machines. We can use MapReduce to quickly find and delete old video data based on time and retention rules.

Map Phase: During this stage, several map tasks operate at the same time, each handling a portion of the metadata database. Every task looks through the records they are assigned and finds videos that have gone past their retention period by checking their timestamps and subscription tier details. The map tasks produce key-value pairs required for the next step. The key should be a unique ID for the video, and the value should indicate if the video needs to be deleted or retained.

Reduce Phase: The reduce tasks get the output from the map tasks. They gather the results and create a list of video IDs that need to be removed. This list is used to remove the related video data from storage and the metadata from the database.

MapReduce helps quickly find and delete billions of video records by spreading the work across several machines. This method helps to clear out data on time without bothering other system tasks too much.

MapReduce for User-Initiated Deletion

For user-initiated deletion, MapReduce can again be leveraged.

Map Phase: The map tasks work on part of the metadata database to find videos linked to the camera(s) or account the user has chosen. The tasks then provide key-value pairs, with the key being the video ID and the value containing details such as the storage location and a deletion flag.

Reduce Phase: The reduce tasks process the output from the map tasks. They can perform the following actions:

Immediate Access Blocking: Put a "delete" tag or marker on the video data in the object storage to prevent access to it, even before it's physically deleted. This keeps user privacy safe while deleting things.

Deletion Coordination: Final step is to physically delete the actual video along with the video metadata. This keeps things consistent and stops any data from being left behind.

Optimizing MapReduce with Metadata Dump

Scanning the entire metadata database for each MapReduce job is expensive and not the only option, especially for frequent deletion tasks. To further optimize the process, we can introduce a metadata dump mechanism:

Periodic Dump: There can be a job that periodically (say daily) scans the entire database and dump all the necessary records and if needed perform joins and store the data in a de-normalized way which will prevent any database access for the actual deletion run.

MapReduce on Dump: Run the MapReduce jobs on this periodic dump instead of querying the database directly. This lightens the database's workload and speeds up processing.

Challenges of Real-Time Tagging

A simple way to stop showing deleted videos is to quickly add a "delete" tag to the metadata of each video when a user asks for it to be removed. This brings up some big challenges:

Size and Expense: Changing billions of records all at once requires a lot of computing power and can really strain database resources.

Atomicity and Reliability: It's really hard to make sure everything runs well and reliably for a big update like this. If something goes wrong, it can lead to issues and let some information get out.

A Different Way: "Should Serve" Indicator

Rather than tagging each video, we could use a marker at the camera level to manage who can access the videos. This marker can be a growing whole number. Let's call this a "should serve" marker.

Ingestion Tagging: Each video that a camera uploads gets tagged with the current "should serve" marker value linked to that camera.

Before serving any video, the system needs to check the video's marker tag with the camera's current "should serve" marker. Videos should only be shown if they have the same tags.

When a user starts the deletion process, the camera's "should serve" marker needs to go up by 1. This makes all the videos that were uploaded before unavailable because their tags don't match the camera's marker anymore.

Sample:At time 0, the camera's "should serve" marker is set to 1. From time 1 to time N, every video that gets uploaded is marked with 1. At time M, a user asks to delete something. The camera's "should serve" marker goes up to 2. Any later tries to watch videos uploaded between time 1 and N will be turned away because their tags (1) are below the current "should serve" marker (2).

Advantages of the "Should Serve" Marker
This method has the following:
It only takes one record update to stop access to billions of videos by just changing the camera's “should serve” marker.
Simplicity. Putting a video tag when the video is uploaded and having the service check it while serving the video helps avoid the need to update billions of records all at once. Once the marker goes up, you can't access deleted videos anymore.
This method allows us to quickly block access to deleted videos and promptly remove the marked videos and their backups, ensuring we comply with data retention rules and privacy laws.
This optimization makes the MapReduce-based deletion process more efficient and scalable, helping smart home security camera systems handle user deletions more easily while improving data privacy.

FUTURE ENHANCEMENTS

While the proposed MapReduce-based solution with metadata dump and "should serve" marker optimizations offers significant improvements in data deletion efficiency and scalability, there are further enhancements that can be explored to further refine the system:

Incorporating Machine Learning for Predictive Deletion

You can use machine learning to figure out how people watch videos and to get rid of data that isn’t accessed often. Looking at past access logs and how users behave, machine learning models can spot videos that probably won't be accessed much in the future. This helps to delete those videos ahead of time, making storage cheaper and easing the deletion process. A machine learning model may be able to infer that for certain users, videos older than 2 weeks have no interactions are hardly ever watched and could automatically flag them for deletion by notifying the user in advance before doing so and may be able to offer some discounts to users depending on savings.

Exploring Alternative Distributed Processing Frameworks

MapReduce is a good way to handle big data, but looking into other distributed processing options might help us find even better ways to be efficient.

Enhancing Metadata Dump Efficiency

The efficiency of the metadata dump process can be further improved by exploring ideas such as:
Incremental Dumps: Rather than dumping the whole metadata database blindly, a job can do incremental dumps that only include the changes since the last one. This cuts down on the amount of data and the time it takes to process each dump.
Data Compression: Opt in for various compression techniques on the metadata dumps to reduce storage space and network transfer costs.
Optimized Dump Scheduling: Change the dump frequency based on things like how often data gets updated and deleted to make the best use of resources.

CONCLUSION

This paper looked at the important issues of handling data storage and deletion in smart home security camera systems, where billions of videos are created and kept every day. We suggested a solution that is easy to scale and budget-friendly, using the MapReduce programming model to tackle the challenges of time-based, retention-based, and user-initiated deletions. Using MapReduce's ability to process things at the same time, we can easily find and remove old videos. This helps us follow data rules and save on storage costs. The new "should serve" marker optimization allows for quick blocking of access to deleted videos, ensuring user privacy without the need to update billions of individual records. This approach using MapReduce and optimizing metadata dumps offers a strong and flexible way to handle video data in smart home security

camera systems. This helps delete data efficiently, cuts down on storage costs, and boosts user privacy, making the system more sustainable and focused on users. Future research can look into more ways to improve things, like using machine learning to guess how people watch videos and getting rid of data that isn't accessed often. Looking into other distributed processing frameworks and how they might work in this area could help make things more efficient and scalable.

REFERENCES

1. Ahmad, A., & Stavrou, A. (2018). Real-time big data analytics for smart homes: A survey. IEEE Access, 6, 42117-42146.
2. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
3. Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. In 2010 IEEE 26th symposium on mass storage systems and technologies (MSST) (pp. 1-10). Ieee.
4. Min, C., Lee, J. W., Jung, I., & Lee, Y. (2017). Secure and scalable storage system for smart home media data in cloud environment. Cluster Computing, 20, 1429-1438.
5. Gartner, Gartner Glossary, "Data Privacy," [Online]. Available: [invalid URL removed]. [Accessed 15 October 2024].
6. Lv, Z., Hu, H., Wang, G., & Wu, Z. (2014). Big data analytics for network intrusion detection. In 2014 IEEE International Conference on Big Data (Big Data) (pp. 747-756). IEEE.
7. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. HotCloud, 10(10-10), 95.
8. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 36(4), 28-38.
9. Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R., Lax, R., ... & Whittle, J. (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. Proceedings of the VLDB Endowment, 8(12), 1792-1803.
10. Ishii, M. (2015). Data lifecycle management in the era of cloud computing and big data. Computer, 48(12), 44-51.