Optimizing Event Management in Home Security Cameras: A Micro-Batching Approach for burst of Event Streams

Sibin Thomas

Tech Lead
sibin_thomas15@hotmail.com

Abstract
For strong event detection and analysis, modern smart home security cameras combine on-device and cloud computing. This usually entails sending to the cloud bursts of sub-events, sometimes called as "event tracks," connected to a single observable occurrence, or "event session." Although this method is good for precise event detection, it can cause inefficiencies, more delay, and more running expenses. This work investigates these difficulties and suggests a micro-batching event data based approach to maximize cloud processing and enhance general system performance and user experience. We introduce the concept and execution of this micro-batching technique together with possible advantages.

Keywords: Smart home security, event detection, cloud processing, micro-batching, latency optimization, event session, event track, optimistic lock failure, resource contention.

1. INTRODUCTION

The growing popularity of smart home security cameras has created demand for smart systems competent of precisely identifying and classifying events. Usually using a two-tiered architecture, these systems first on-device processing for preliminary event detection then cloud-based analysis using advanced artificial intelligence and machine learning [1]. While the cloud infrastructure handles more complicated activities such discriminating between distinct sounds and camera observed events [2], on-device processing addresses fundamental event identification. To precisely identify and classify the overall "event session," this method sometimes calls for sending several sub-events—or "event tracks"—to the cloud. A person approaching a door, ringing the doorbell, and then departing might, for instance, create several "person detected," "doorbell pressed," and maybe "motion detected" events. These various events together create the whole "package delivery" event session by means of their interaction. There can be difficulties depending on this reliance on cloud computing and the broadcast of many sub-events. Transmitting each event separately can specifically result in more cloud load, more latency resulting from optimistic lock failures in the database [3], more running expenses, and finally a less responsive user experience. This work attempts to solve the problems by suggesting, based on micro-batching, an optimal event session management technique.

2. CONTEXT

When talking about home security cameras, a "session" is a group of video and audio data that corresponds to an event that a person can see. Some examples are someone walking up to the door, a friend ringing the doorbell, or a smoke alarm going off. After these video and audio snippets are categorized, users can access them through mobile apps or the web. How these sessions are recorded and used depends on the recording mode:

Record Video All the time: During event sessions, parts of a continuous video stream that might be interesting are found and emphasized. For example, if a camera is recording all the time, an event session could be set up when a person is seen, which would make that part of the video stand out for the user.
Record only when events are detected: When certain events are recognized, line-powered cameras start recording during event sessions. The camera only records when it needs to in this setting, which uses less power.
Record only when events are detected for cameras on battery power source: To save power, event sessions control when recording starts and stops for cameras that are driven by batteries. For wireless cameras to get the most out of their batteries, this is a must.
Terminology:
A device event is a message from the camera that tells you about a certain observation or state change, like when a person is seen or the doorbell is hit.An event track is a series of device events that show a single, ongoing event, like a person moving across the camera's field of view.An event session is a group of event tracks that together make up a full event that the user can see, like a package delivery where someone comes up, delivers the package, and rings the doorbell.
Among other things, new event sessions start when the perception service picks up on a key event, like when a person is identified. Feeling the doorbell ring.
Find tampering.
Recording that starts in the cloud (for example, when a security warning goes off).
The strategy for creating sessions needs to be able to be changed to fit new features and changing needs. Session IDs connect video segments and gadget events so that analysis and presentation can be done more easily.

3. LIFECYCLE OF AN EVENT SESSION

The following steps make up the lifecycle of an event session [4]:
Session Start: The camera sends out an event called "session start," which has information about the session, such as a unique session ID. The start of a new event session is marked by this event.
Track Initiation: A track started event with a unique track_id is sent out for each track in the session. As an example, a "Person Track Started" event is made when a person is found.
Track Updates: New events that happen on the track are connected to the session by their related_event_id. The first "Person Track Started" event is linked to all future "Person Detected" events as the person moves.
Track Ending: An event called "track ended" is sent out when a track is finished. This lets you know that that track is over.When the last track stops, the camera starts a timer that lasts for a certain amount of time. The session moves on to the next stage if there are no new track changes during this time.
Session End: To mark the end of the session, a closing event called "session end" is sent out. The whole event session is over when this event happens.When you stop using the program for a while and then start it again, the session will continue if a new track update happens during that time. This stops multiple broken sessions from being made for a single ongoing event.
Event sessions are very important for defining the time limits of useful video segments, which makes it easier for users to watch and analyze them. This process can be complicated because it needs different camera parts that are in charge of finding events, taking video, and managing timestamps to work together.

4. CHALLENGES

Individual events being sent to the cloud too often, especially when there are a lot of them, can waste time and money [4]. This is mostly because:

Increased Cloud Load

Each event starts a different cloud worker, which can cause resource contention and optimistic lock failures if multiple workers try to change the same database record at the same time. For instance, if a lot of "Person Detected" events happen at once, many cloud workers may attempt to change the same event session record, which could cause problems and require more attempts.

Increased Latency

When an optimistic lock fails, it has to be tried again, which takes longer to process and delays notifying users. This can make it take a long time to get important messages.

Costs go up because more cloud resources are used for processing and retries, which drives up running costs. It takes more resources to process each event one at a time than to process them all at once.

Increased Queries Per Second (QPS)

A lot of separate events can make the system's QPS load go up, which could affect its general performance and ability to grow.

Higher Latency for End Users

In the end, these errors can make things slower and less responsive for end users. Users might have to wait longer to get notifications or watch videos that have been taken.

Getting more done with microscopic bathing events

In order to deal with these problems, this paper suggests a method for event session data called "micro-batching." To do this, events that happen within a short time frame (like 100ms) are put together into a single "batch" and then sent to the cloud. This method has a number of advantages:

Lighter Cloud Load

Micro-batching cuts down on the number of individual cloud workers that are needed, which lowers the risk of resource contention and hopeful lock failures. We reduce the number of simultaneous database changes by batching events. This lowers the chance of conflicts.

Lower Latency

Micro-batching speeds up reaction times and notification delivery by cutting down on processing overhead and retries. This makes sure that users get alerts at the right time.

Better efficiency

Grouping events into batches makes the best use of cloud resources and data transfer. Batching cuts down on network traffic and handling work.

Lower QPS

Micro-batching lowers the system's QPS load, which makes it easier to scale and run faster.

Better User Experience

Less latency and faster processing make the experience smoother and more quick for the end user.

Implementation: We use a method called "optimistic upload, lazy micro-batching" to make sure that batching works well with as few extra delays as possible. This method takes advantage of the fact that most events happen a while apart and uploads the first track-event right away so that alerts can be sent out on time.

Here's how the event session downloader works:

I**nitial Events**

The first track event and event session start event are posted right away in small batches. This makes sure that the process starts quickly in the cloud.

Events That Follow

For events that follow, the time since the last post is checked. If it goes over the micro-batching time (Xms), the event is uploaded right away. If not, events are grouped together for a short time (like 100ms - time_delta) before being uploaded. This lets batching change based on how often events happen.

Early Boot Handling

For devices that run on batteries, events that happen before Wi-Fi startup are "micro-batched" and sent as soon as the connection is made. This cuts down on the number of uploads while still making sure that notifications are sent on time.

Session End

The session end event is always uploaded right away so that the device doesn't wait to shut down and the battery lasts longer.

5. APPLICATIONS BEYOND SECURITY CAMERAS

This study's main goal is to find the best way to manage event sessions in smart home security cams, but the micro-batching method shown here can be used in more situations [1]. This approach could help any cloud-based system or Internet of Things gadget that makes a lot of event streams.

A few examples are given below:

Touch sensors for smart homes

Imagine a smart home with lots of sensors that check for things like activity, temperature, humidity, and the state of the doors and windows. If all of the data from these sensors is sent at once, the network and cloud tools might not be able to handle it. Micro-batching, which groups events together before sending them, can greatly reduce this cost. If the sensor doesn't want to send a new temperature reading every few seconds, it could read all the values it collected in a minute at once and send them all as one packet. Because of this, there are fewer transmissions and less info in general.

Wearable Tech

Fitness trackers and health monitors that you can put on your body send you a lot of information about your heart rate, exercise level, sleep patterns, and other things. By micro-batching, this data can be sent to the cloud to be analyzed and stored while the wearable device's delay and power use are kept to a minimum. In order to get the best balance between data quality and transmission speed, a heart rate monitor might read all the data it has gathered over 5 minutes at once and send it as a single data point [5].

Industrial Monitoring Systems

In industrial settings, sensors keep an eye on things like flow rates, pressure, and temperature, which creates big data streams. This data can be sent to the cloud in groups to save time and space on the network. This makes research go faster. For example, shaking sensors on important machines in a factory could read data in batches over a short period of time and send it regularly, making it possible to keep an eye on the health of the machines without putting too much stress on the network [6].

Platforms for Trading Financials

High-frequency trading systems make trades very quickly while also processing huge amounts of market data. Micro-batching can be used to group market data updates and order executions together before sending them to the trade engine. This can cut down on latency and make the system work better overall. This can make it a lot easier to stay ahead of the competition and take advantage of short-term trade opportunities. While micro-batching can be useful for smart home security cams, it can also be useful in the following situations:

Fewer separate requests for the cloud to handle: By grouping several events into one batch, we reduce the number of requests that the cloud has to handle. This makes its resources less busy and lowers the risk of hopeful lock failures.

Less latency: Fewer communications and less processing overhead lead to faster reaction times and faster delivery of notices.

Better efficiency: Batching makes it easier to send large amounts of data quickly. They can make things more scalable and lower prices.

6. CONCLUSION

The issues with managing event sessions in smart home security cams were looked into in this paper, focusing on the waste and longer wait times that come from sending separate sub-events to the cloud. To fix these issues, we offered a method called "micro-batching," which groups events that happen quickly into a single batch before sending them. This way reduces the load on the cloud, cuts down on latency by stopping optimistic lock failures, makes better use of resources, and ultimately makes the user experience better. We made sure that notifications were sent on time by using a method called "optimistic upload, lazy micro-batching" to balance the need for effective batching with as few unnecessary delays as possible.

The suggested micro-batching method is a big step toward making event session control in smart home security cameras work better. Even though this study was mostly about one application, the concepts and methods can be used with other cloud-based systems and IoT devices that handle high-frequency event data in a similar way. To make these kinds of systems work better and respond faster, we might need to learn more about adaptive micro-batching intervals that take into account network conditions and event traits. It might be even more efficient and cost-effective to look into the benefits of combining micro-batching with other optimization techniques, such as edge computing and data compression.

REFERENCES

1. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).
2. Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4580-4584). IEEE.
3. Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media, Inc.
4. 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.
5. He, Z., & Jin, L. (2019). An energy efficient data collection scheme for wearable sensor networks. *IEEE Transactions on Mobile Computing*, *19*(3), 568-580.
6. Si, W., Wan, S., Zhou, Z., & Fu, Q. (2018). Real-time big data processing for industrial sensor data: A review. *IEEE Transactions on Industrial Informatics*, *15*(11), 6210-6221.