Revolutionizing Surveillance Technology with Distributed Systems

SOFE 4790U Distributed Systems

Ahmaad Ansari
Software Engineering
Ontario Tech University
ahmaad.ansari@ontariotechu.net

Zubair Islam
Software Engineering
Ontario Tech University
zubair.islam@ontariotechu.net

Nabil Saleh
Software Engineering
Ontario Tech University
nabil.saleh@ontariotechu.net

Waddah Saleh
Software Engineering
Ontario Tech University
waddah.saleh@ontariotechu.net

Abdullah Waseem
Software Engineering
Ontario Tech University
abdullah.waseem@ontariotechu.net

***Abstract*— Closed-circuit surveillance systems have paved the way to distributed, microservices-based architectures like PulsePoint. This technology provides security, scalability, and improved user interfaces. PulsePoint illustrates flexibility by leveraging microservices, allowing interaction with developing technologies. Its primary features include security protections, scalability, analytics, and user interface enhancements.**

Keywords— microservices, security, video analytics, Docker, scalability.

I. Introduction

Surveillance systems have become integral in today's society as they provide a fundamental layer of security, monitoring, and data collection across several domains. As technology advances, traditional closed-circuit surveillance systems fall behind as they are no longer able to support increasing data volumes, lack flexible access controls, and struggle to integrate with modern technologies. Our solution proposes a distributed approach to mitigate the limitations of traditional closed-circuit systems. This approach enables our system, PulsePoint, to enforce security measures, provide data analysis yielding actionable insights, and is built off of a framework that is capable of accommodating future technological advancements.

The specific distributed technology that PulsePoint leverages is a microservices architecture that improves scalability and adaptability. PulsePoint responds well to quickly changing needs by separating functionality into independent services, guaranteeing scalability without sacrificing efficiency. PulsePoint can remain agile, and responsive, and is capable of integrating emerging technologies seamlessly thanks to its microservices-based platform.

II. Related Work

A. Closed-Circuit Surveillance Systems

Traditionally, in surveillance technology, we have Closed-Circuit Surveillance (CCS) Systems. Such facilities typically include cameras that are linked up with a limited number of monitors and recorders operated in a private, closed system [1]. There is nothing as simple and reliable as Closed-Circuit televisions (CCTVs). On the other hand, these solutions might prove rigid, call for lengthy cabling, and are not easy to scale up.

B. Distributed Surveillance Systems

They employ the use of modern technology such as microservice architecture for increased scalability and flexibility [2]. Such products can generally be compatible with different technologies and can also change for new requirements. Besides, distributed systems usually provide for remote access and control thus offering much versatility in diverse settings [2]. This capability makes them quite flexible since it allows them to be used as an example and to capture images on smartphones/tablet devices. The main improvements offered by distributed systems over traditional CCTVs include:

- *Scalability:* Easier to expand and adapt to larger networks without significant infrastructure changes.
- *Flexibility:* Can integrate with a broader range of technologies and adapt to various environments.
- *Accessibility:* Often support remote access and control, enhancing usability.
- *Adaptability:* Capable of using a wider range of devices such as cameras.

The initial cost of installing CCS normally is more expensive because of physical infrastructure. For example, their maintenance includes conducting a physical inspection as well as making adjustments for repairs, which result in further expenses in the long run. Unlike that, distributed systems can be cheaper in terms of start-up costs if they exploit already installed equipment and networks. However, they require constant upkeep, and that implies consistent software upgrades and installation of cybersecurity mechanisms for them to continue being functional and secure.

These differences signify a shift in surveillance technology, moving from fixed, hardware-dependent systems to more dynamic, software-driven solutions. This transition reflects broader trends in technology, emphasizing adaptability, user-centered design, and integration with the Internet of Things (IoT). While this overview provides a general comparison, specific details and examples from academic or industry studies would offer a more robust analysis.

III. Proposed Solution

A. System Overview and Operational Scope

The goal of our system is to create a solution that smoothly combines contemporary technologies while resolving the shortcomings of conventional closed-circuit systems. With a broad user base in mind, our solution is tailored to security professionals who want real-time monitoring in industrial settings as well as users who want personal home surveillance. Its design implies scalability, strong security measures, and an easy-to-use interface. Using Python-enabled devices with cameras, our solution enables users to easily turn compatible devices into security cameras, ensuring simplicity of setup and operation in a variety of contexts.

B. Technologies Used

To provide a robust and scalable surveillance system, PulsePoint was built using many modern technologies:

- *Node.js with Express.js:* Used for backend logic and fast API administration.
- *JavaScript and React.js:* Used for an interactive user interface and a seamless user experience.
- *Python and Flask:* Used for camera integration, settings, and streaming capabilities.
- *MongoDB:* Chosen because of its flexibility and scalability in managing data.
- *Docker:* Used to streamline service deployment, scalability, and administration.

C. Database Schema

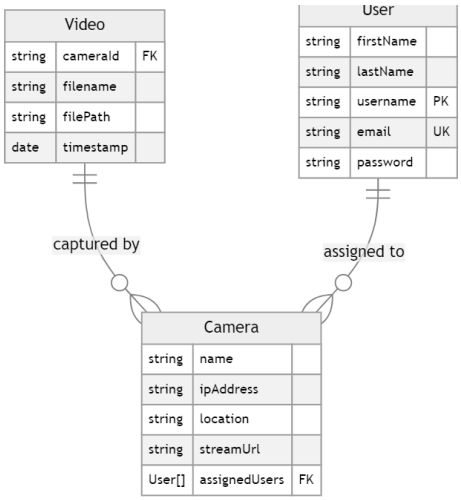

Fig. 1. MongoDB Models Entity-Relationship Diagram.

The figure above visually represents the relationships and structure of three MongoDB models used in the system: Video, User, and Camera.

- *Video Entity:* Represents videos captured by cameras. It has attributes like cameraId (foreign key linking to the Camera entity), filename, filePath, and timestamp.
- *User Entity:* Represents users in the system. It includes attributes such as firstName, lastName, username (primary key), email (unique key), and password.
- *Camera Entity:* Represents camera devices. It has attributes like name, ipAddress, location, streamUrl, and assignedUsers (an array of foreign keys linking to the User entity).

The Video entity is linked to the Camera entity, indicating that each video is associated with a specific camera (a one-to-many relationship, as one camera can have multiple videos). The User entity is linked to the Camera entity, showing that users can be assigned to cameras (also a one-to-many relationship, as one user can be assigned to multiple cameras).

D. System Architecture and Design

PulsePoint is built on a microservices-based approach, enabling modularity, scalability, and autonomy of services. Microservices were used in our system as they partition the system into smaller, self-contained units, each responsible for distinct functionalities. This modular design offers several advantages:

- *Modularity:* Services are independent, allowing for isolated development, deployment, and scaling, which improves the system's agility.
- *Scalability:* Individual services can scale independently based on the demand for specific services, allowing the system to optimize resource utilization.
- *Resilience:* When and if a service fails, the complete system remains operational, guaranteeing fault isolation and reliability.

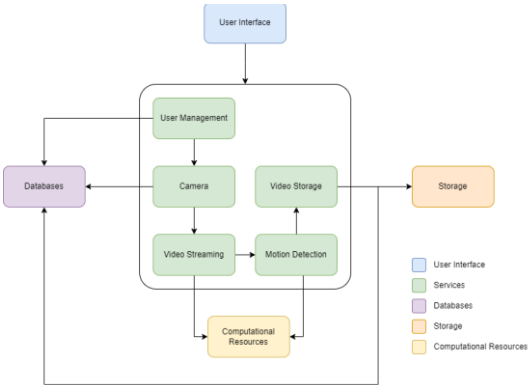

Fig. 2. PulsePoint Microservices System Architecture.

PulsePoint's architecture is designed to be modular, which supports easy maintenance and rapid development cycles.

- *User Interface:* Presents a responsive and adaptable web interface for enhanced user interaction.
- *User Management Service:* Handles user profiles, access permissions, and subscription levels.
- *Camera Service:* Aids in camera integration, configuration, and streaming access.
- *Video Streaming Service:* Guarantees efficient video stream encoding and transmission, adaptable to various bandwidth conditions.
- *Motion Detection Service:* Applies AI and machine learning for sophisticated motion analysis and notifications.
- *Video Storage Service:* Uses secure, scalable storage solutions for managing substantial video data volumes.

The system primarily uses API-based communication to interact between different services. The API gateway within PulsePoint is mainly tailored towards inter-service communication. Its functionalities include.

E. Implementation and Operational Flows

This section describes the operational workflows within four fundamental services: User Management, Camera Management, Video Analytics, and Video Storage. All of these services come together to form the surveillance ecosystem, encompassing user authentication and access control, camera integration and configuration, as well as the storage and analysis of video data.

The User Management Service serves as a gateway to the entire system. The process begins with user registration, where new users provide the necessary information to gain access to the system. Once a user is registered, they must authenticate themselves which is evaluated by a JSON Web Token (JWT). This authentication allows the users to access the services within the system.

The Camera Management Service allows for devices to be integrated and configured on the system as surveillance cameras. This service is constantly identifying and authenticating new cameras discovered on the system's network. New cameras can be configured by running a Python script that initializes a Flask server to stream video from the main camera on the device. This Python script holds the details of the camera including its name and location, which can later be changed through the system's user interface. The device's local IP address and specified port are used to create the video stream URL which is then saved into a database of cameras on the system through the Camera Management Service.

The Video Analytics Service is configured to analyze all video streams on the system. In our current implementation, this service performs motion detection. The service periodically checks for new cameras on the system and uses multithreading to enable concurrent analysis of every camera stream available. Upon detecting motion, this service will start to capture video and communicate with the Video Storage Service to store the frames in which motion was detected for future user retrieval.

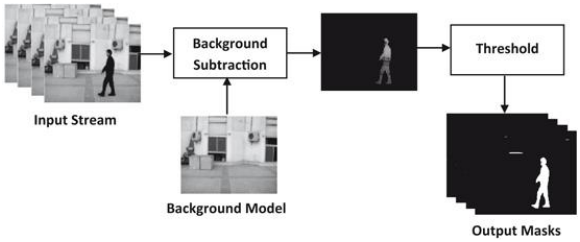

Fig. 3. Background subtraction and contour analysis [3].

The motion detection integrated into our systems Video Analytics Service leverages a background subtraction technique in parallel with contour analysis [3]. The service will begin by comparing differences between consecutive frames. This process involves creating a mask that highlights areas of change. After the mask has been generated the system will apply a threshold and contour analysis to identify distinct shapes within the camera feed. Any contours that exceed a predefined threshold are then flagged and indicate motion within the scene.

Lastly, the Video Storage Service manages the storage, retrieval, and deletion of video data captured and recorded by the Video Analytics Service. The recorded videos stored on the system can only be accessed by authorized and authenticated users.

F. Scalability, Expansion, and Future Preparedness

The system architecture utilizes a microservices architecture and containerization to implement scalable and adaptable operations.

1) Microservices Architecture: Using a microservices-based approach allows system components to work independently. This structure allows for the horizontal scaling of certain services in response to changes in demand without interfering with overall system performance.

2) Containerization and Orchestration: Using Docker for containerization simplifies the encapsulation of services into flexible and portable containers. This solution accelerates deployment, replication, and administration operations, allowing for easy scalability while reducing dependency on the underlying infrastructure.

G. Addressing Challenges with Innovative Solutions

*1) Identified Challenges:*We encountered several challenges when developing our monitoring system. Addressing these issues ensured that our system could meet changing demands.

- *Authorized Access Control:* Implementing a strong authorization system to ensure secure access to system services and recorded films proved difficult.
- *Dynamic Camera Discovery:* Integrating additional cameras into the network while keeping current operations running was a huge issue.
- *Real-time Motion Analysis:* It was difficult to ensure quick and accurate motion detection while regulating system resource allocation.

2) Innovative Solutions Implemented: We overcame the challenges mentioned above by developing innovative solutions and improving the system's performance and dependability.

- *JWT Standardization:* A centralized JWT secret management strategy was created to solve the issue of allowed access. This guaranteed consistency across all containers, providing access privileges only to users who had valid JWTs.
- *Auto Camera Registration Protocol:* Created an automatic protocol to identify and incorporate newly discovered cameras into the system without disrupting service.
- *Motion Detection Algorithm Optimization:* Improved the motion detection algorithm to run efficiently on a variety of hardware, balancing accuracy, and resource usage.

IV. Evaluation and Results

The evaluation of our solution is an essential factor in the construction of our application, to ensure that it meets the legal and ethical standards. To validate our solution, we used multiple evaluation metrics: Security, Privacy, Usability, and Performance.

A. Secuirty

To ensure that our application has an impenetrable design, we implemented multiple security measures such as authentication mechanisms, authorization control, and audit trails and logging. Authentication mechanisms ensure that users are logged in as verified users before accessing any components of the application. Additionally, authorization control is used to grant or deny access to those components. Furthermore, we implemented audit trails and logging as a further precautionary measure, to keep a record of actions performed by users, with their respective time stamps.

B. Privacy

Preserving the privacy of our users is a major focus of our application. When a user creates an account and registers their cameras on their network, these cameras must be only accessible to them. Therefore, we designed our application to have client-centric data with the use of authentication measures, which ensures that data is only tied to the account that was registered and cannot be accessible by any other user. We also have data minimization, which reduces the amount of data that is collected and stored. Since we only record footage during motion detection and only retain that footage for a week at maximum, we can prevent the scale of the violation of user privacy in the case of potential data breaches.

C. Usability

To enhance the quality of our application, we have designed a simple intuitive user interface to enhance the quality of our application. Our UI ensures an excellent user experience and straightforward interaction and is optimized for the satisfaction of the users. We have also added error handling, which is used for communicating to the user whenever they use the wrong credentials or when they have added a non-existing camera, which overall enhances the functionality of our application.

D. Performance

The performance of our application is a dominating factor when it comes to usability. We need to be able to ensure multiple metrics such as response time, precision, scalability, resource utilization, and fault tolerance. Our application must have a quick response time, such as the speed at which videos are streamed to the user. We must also have accurate precision in detecting the motion of objects to avoid false positives. Additionally, it must also ensure scalability, so as multiple users add multiple cameras, our application must be able to scale itself to handle the increased demand. Furthermore, our application must be very efficient in its resource utilization, to make sure that energy is saved and no component is left idle unnecessarily. Lastly, our application must be adaptable to system failures or disruptions. It needs to detect, isolate, and repair faults to avoid interrupting any services and increase the availability and reliability of the system.

E. Results

We conducted testing of our application by separately testing the frontend and the backend. For the backend, we used Postman to extensively test out all our API calls for each service, such as POST, DELETE, GET, and PATCH. We also conducted tests on verifying the camera availability on the network after running the Python script on the client cameras, which exposes the IP address through a port. We then tested the frontend, with tests ranging from registration of users, authentication of users, camera management (Add, Edit, View, Delete), and viewing the live feed footage and also testing the storage access and download of recorded footage consisting of motion detection.

V. Conclusions and Contributions

Finally, PulsePoint brings out the first innovative step to surveillance technology with the use of Microservices Architecture which concentrates on the design for the user while also maintaining the ability of scaling. Its system distinguishes itself through key contributions formed with modular emphasis: Therefore, User Management services combine with Camera Service features as well as Motion Detection competencies to form a cohesive system of surveillance. Through the seamless incorporation of modern technologies, PulsePoint establishes its cardinal place in upgrading surveillance tech. It also takes care of scalability concerns, enabling real-time monitors at diverse setups, depicting crucial roles vividly. As PulsePoint takes off into the next stage it becomes a revolutionary power within distributed security surveillance systems. Security, scalability, and innovations will form part of a vision that assures improved functionality combined with user-friendly design.

VI. Future Works

Continued research and development will center on utilizing emerging technologies enhancing current features and exploring new ways to improve system efficiency and security. Listed below are multiple future research and development avenues:

- *Real-time Alerts Integration:* Investigation and integration of real-time alert systems to improve

responsiveness and rapid action on discovered security breaches.

- *Advanced Analytics Implementation:* Extending analytics capabilities beyond motion detection to include features such as face detection, tripwire systems, object recognition, and behavior analysis, allowing for a more thorough analysis.
- *Distributed Processing Implementation:* Developing strategies for spreading computationally expensive tasks among numerous system nodes or servers. This will enable the system to improve overall performance, especially under peak loads.
- *Load Balancing Strategies:* Using load balancing strategies to spread workloads evenly across microservices. This reduces service bottlenecks and maintains performance.
- *Potential Camera Health Check System:* Create a monitoring method that analyzes camera health, guaranteeing consistent and optimal performance across all connected devices.
- *Enhanced Data Security Measures:* Increase security by using encryption algorithms to protect against data files, protecting them from threats and unauthorized access.

REFERENCES

[1] Video security solutions: CCTV & IP Video Surveillance Guide," Vista Security, https://vistasecurity.com/insights/video-security-solutions-cctv-ip-guide/ (accessed Dec. 5, 2023).

[2] "Distributed Systems Monitoring: The essential guide: Loggly," Log Analysis | Log Monitoring by Loggly, https://www.loggly.com/use-cases/distributed-systems-monitoring-the-essential-guide/ (accessed Dec. 5, 2023).

[3] A. Gulati, "Vehicle Motion Detection using Background Subtraction," Analytics Vidhya, Mar. 15, 2022.