Case Study

Enhancing Application performance with KMS Monitor

Kurento Media Server (KMS), a top-tier WebRTC media server in the MCU category, powers real-time multimedia applications. A client in the insurance sector faced a significant challenge with their conferencing application, which struggled to host multiple meetings on the media server and required frequent restarts, posing a major risk to their production system.

SpringCT investigated the system and identified the root cause: improper coding that prevented media pipelines—resources on the media server—from being released under certain conditions, leading to a gradual increase in server load. To resolve this, SpringCT developed KMS Monitor tool that can monitor media pipelines and endpoints present challenges, particularly with issues like lingering zombie endpoints after meetings.

Product Features

The KMS Monitor is a comprehensive monitoring tool designed for Kurento Media Server. Its key features include:

- **Pipeline Visualization**
  Real-time graphical representation of media pipelines and connections between endpoints.
- **Media Flow Monitoring**
  Visual indicators (green for healthy, red for issues) to show media flow directions and detect transmission problems.
- **Detailed Stats Analysis**
  Key metrics such as bitrate, packet loss, and latency for each connection.
- **WebRTC Negotiation Inspection**
  Displays offer and answer exchanges for effective debugging.
- **PlayerEndpoint Attachment**
  Allows direct playback of media streams for inspection and validation.

Key Technical Achievements

- **Zombie Pipelines and Endpoints**
  Zombie WebRTC endpoints and pipelines would occasionally persist even after conferences ended, consuming server resources unnecessarily and impacting new sessions.
- **Lack of Real-Time Feedback**
  Without a monitoring solution, identifying and diagnosing issues like packet loss, latency, or ICE candidate failures required significant manual effort.
- **Complex Debugging**
  Troubleshooting WebRTC sessions without clear visibility into offer/answer exchanges or endpoint stats slowed resolution times.

Technologies Used

- **Kurento API:** Kurento Client JS APIs for monitoring pipeline components and Stats API for real-time statistics on media flows and connections.
- **WebSocket Communication:** Enables low-latency updates between KMS and the monitoring dashboard.

- **Visualization Libraries:** Tools like D3.js to dynamically render the pipeline connections and media flows.
- **Backend Infrastructure:** Node.js and Express for handling communication between the KMS Monitor and Kurento Media Server.
- **UI Frameworks:** Modern web technologies like React.js for an interactive and responsive user interface.

Results

- Detection of Zombie Endpoints: The monitor identified zombie pipelines and endpoints post-conference, enabling cleanup and freeing up server resources, which also helps finding defects in KMS pipeline architecture on the KMS App Server.
- Improved Debugging: With real-time stats and WebRTC negotiation insights, developers reduced troubleshooting times by 50%.
- Enhanced Media Quality: Continuous monitoring allowed for proactive identification of packet loss and bitrate issues, ensuring smoother media delivery.
- Operational Efficiency: Media operations teams gained better visibility into active sessions, reducing downtime and enhancing user experience.

Conclusion

- The KMS Monitor has proven to be an indispensable tool for managing Kurento Media Server instances.
- By detecting and addressing issues like zombie endpoints and providing real-time feedback on media flows, SpringCT could significantly improved system performance and reliability by fixing the code.