WebSocket Communication for Real-Time Progress and Logging

Tomasz Olejarczuk

29-05-2024

Contents

WebSocket Communication for Real-Time Progress and Logging

1. Summary

This document details the implementation of WebSocket communication between the C# API wrapper and the MATLAB production server. This real-time communication channel enables the following key features:

- **Progress Tracking:** Clients can receive updates on the progress of their MATLAB calculations.
- **Log Streaming:** Logs generated by the MATLAB server are transmitted to the wrapper for storage and potential display to the user.

2. Introduction

2.1 Background

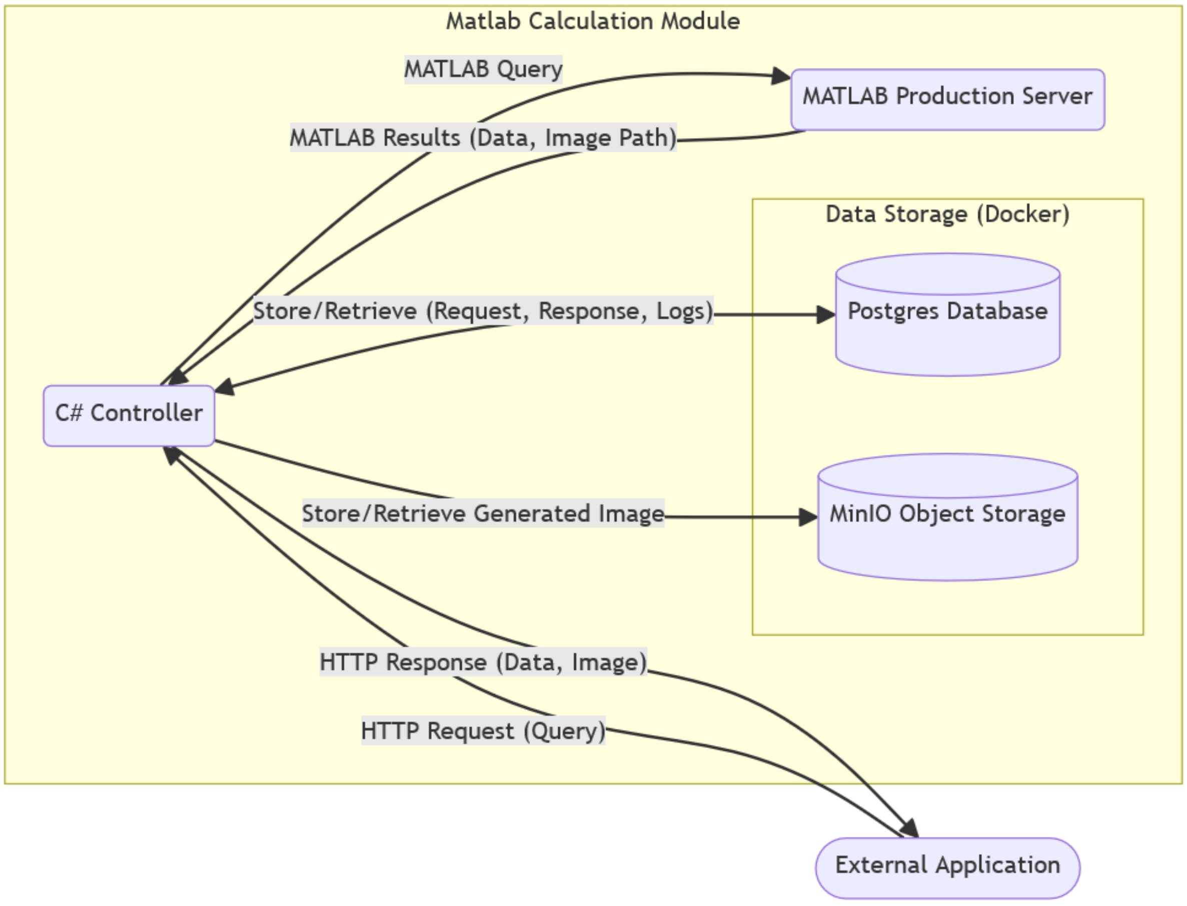
Providing real-time feedback to users during potentially long-running MATLAB calculations is crucial for enhancing the user experience. Additionally, capturing and storing MATLAB logs is essential for debugging, troubleshooting, and gaining insights into system behavior. WebSockets, a full-duplex communication protocol over a single TCP connection, are well-suited for these tasks due to their low latency and ability to push data from the server to the client.

2.2 Objectives

- Establish a WebSocket connection between the C# wrapper and the MATLAB server.
- Implement a mechanism for the MATLAB server to send progress updates.
- Stream MATLAB logs to the wrapper in real time.
- Store received logs in the PostgreSQL database.

3. Methodology

3.1 Architecture


Matlab Calculation Module
MATLAB Query
MATLAB Production Server
MATLAB Results (Data, Image Path)
Data Storage (Docker)
Store/Retrieve (Request, Response, Logs)
Postgres Database
C# Controller
Store/Retrieve Generated Image
MinIO Object Storage
HTTP Response (Data, Image)
HTTP Request (Query)
External Application

3.2 Technologies

- **WebSockets:** Communication protocol for real-time data exchange.
- **C#:** Programming language for the API wrapper's WebSocket implementation.
- **MATLAB:** Programming language for the MATLAB server's WebSocket implementation.
- **System.Net.WebSockets (C#):** Built-in .NET library for WebSocket handling in the C# wrapper.
- **MatlabWebSocket (MATLAB):** Third-party library (https://github.com/jebej/MatlabWebSocket) for WebSocket handling in MATLAB.
- **PostgreSQL:** Database for storing logs.

3.3 Implementation Details

- **C# Wrapper:**
  - **WebSocket Endpoint:** The wrapper exposes a WebSocket endpoint (e.g., `/ws/logs`) for clients to connect to.
  - **Connection Handling:** Upon receiving a connection request, the wrapper establishes a WebSocket connection using `System.Net.WebSockets` and initiates log streaming.
  - **Log Processing:** The wrapper receives log messages from the MATLAB server, parses them, and inserts them into the PostgreSQL database.
- **MATLAB Server:**
  - **WebSocket Client:** The MATLAB server acts as a WebSocket client, connecting to the wrapper's WebSocket endpoint using the `MatlabWebSocket` library.
  - **Progress Updates:** The server periodically sends progress updates (e.g., percentage complete) as WebSocket messages.
  - **Log Transmission:** The server sends log messages to the wrapper as they are generated.

Matlab code snippet:

```
client = SimpleClient('ws://localhost:8765'); % Connect to wrapper
WebsocketsSend(client, TeamName, progress);   % Send progress update
```

4. Results & Insights

While the WebSocket communication between the C# wrapper and MATLAB server is functional, the current implementation reveals areas for optimization and refinement:

4.1 Key Outcomes

1. **Real-Time Progress Updates:** The WebSocket connection successfully delivers progress updates from the MATLAB server to the client, enhancing the user experience by providing visibility into the calculation process.
2. **Log Streaming:** MATLAB logs are transmitted in real-time to the C# wrapper, where they can be processed and stored in the database for later analysis.
3. **Integration with Existing Code:** The WebSocket functionality has been integrated into the existing MATLAB code (`teamiumalgorithmjson.m`) and the C# wrapper, demonstrating the feasibility of this communication approach.

4.2 Challenges and Observations

1. **Error Handling:** The current MATLAB implementation lacks robust error handling for WebSocket communication. If the connection fails or encounters errors, the MATLAB script might terminate abruptly, potentially leaving the client in an uncertain state.
2. **Message Structure:** The format of messages sent over the WebSocket connection (e.g., how progress updates and log messages are structured) could be standardized and documented more clearly to ensure consistent parsing and handling on both the client and server sides.
3. **Security:** The current implementation does not include authentication or authorization mechanisms for the WebSocket connection, which could pose a security risk in production environments.
4. **Scalability:** The current approach, where the MATLAB server initiates the WebSocket connection to the wrapper, might not be optimal for scaling to multiple MATLAB instances. A more scalable approach could involve the wrapper acting as a WebSocket server, accepting connections from multiple MATLAB instances.

4.3 Potential Enhancements

1. **Robust Error Handling:** Implement comprehensive error handling in both the MATLAB and C# code to gracefully manage connection failures, timeouts, and other potential issues.
2. **Message Protocol:** Define a clear and well-documented message protocol for WebSocket communication, specifying the format and content of progress updates, log messages, and other types of messages.
3. **Authentication and Authorization:** Introduce authentication and authorization mechanisms to secure the WebSocket connection and ensure that only authorized clients can receive updates and logs.
4. **Scalability Considerations:** Explore architectural changes, such as having the wrapper act as a WebSocket server, to facilitate scaling to multiple MATLAB instances.
5. **Client-Side Feedback:** Enhance the client-side implementation to provide more informative and visually appealing feedback based on the received progress updates and log messages.

By addressing these challenges and implementing the suggested enhancements, the WebSocket communication between the C# wrapper and MATLAB server can be made more robust, secure, and scalable, ultimately improving the overall user experience and system reliability.

5. Conclusion

The implementation of WebSocket communication between the C# API wrapper and the MATLAB server represents a significant step towards enhancing the user experience and system observability. The ability to provide real-time progress updates and stream MATLAB logs directly to the client empowers users with valuable insights and control over their calculations.

While the current implementation demonstrates the feasibility and benefits of WebSocket communication, there is a lot of space for refinement and optimization. By addressing the identified challenges, such as robust error handling, message standardization, and security enhancements, the WebSocket functionality can be elevated to a production-ready state.

In conclusion, the WebSocket integration lays the groundwork for a more responsive, transparent, and user-friendly experience when interacting with the MATLAB Production Server. As the implementation matures and addresses the identified areas for improvement, it will become an indispensable component of the overall system, enhancing both user satisfaction and developer productivity.