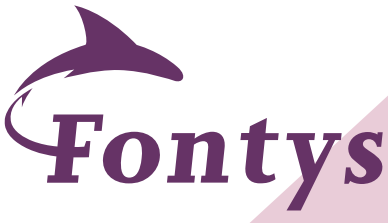
Enhancing MATLAB Production Server with DAPR

A Proposal for Simplified Distributed Application Development

Tomasz Olejarczuk



29-05-2024

Contents

Enhancing MATLAB Production Server with DAPR

1. Summary

This document proposes the integration of the Distributed Application Runtime (DAPR) into the MATLAB Production Server API wrapper project. DAPR is a portable, event-driven runtime that simplifies the development of resilient, microservice-based applications. This proposal outlines the potential benefits, implementation considerations, and a roadmap for integrating DAPR into the existing architecture.

2. Introduction

2.1 Background

The current MATLAB Production Server API wrapper project is a monolithic application, with the C# API wrapper, MATLAB server, database, and storage tightly coupled. While functional, this architecture presents challenges in terms of scalability, maintainability, and the adoption of modern microservice patterns.

DAPR, as a runtime for building distributed applications, offers a compelling solution to these challenges. By providing a set of building block APIs and a sidecar architecture, DAPR abstracts away the complexities of distributed systems, enabling developers to focus on business logic rather than infrastructure concerns.

2.2 Objectives

- Explore the potential benefits of integrating DAPR into the project.
- Outline a roadmap for DAPR adoption, considering the existing architecture and requirements.
- Identify specific DAPR building blocks that can address current and future challenges.

3. Potential Benefits of DAPR Integration

1. **Simplified Service-to-Service Invocation:** DAPR's service invocation building block enables reliable and secure communication between the API wrapper and the MATLAB server, handling retries, timeouts, and service discovery automatically.

2. **State Management:** DAPR's state management API abstracts away the complexities of persisting and retrieving state data, allowing the API wrapper to easily store and access information in a distributed environment.
3. **Event-Driven Architecture:** DAPR supports pub/sub messaging, enabling loosely coupled components to communicate through events. This can be used to trigger MATLAB calculations based on specific events or conditions.
4. **Bindings and Triggers:** DAPR's bindings enable seamless integration with external systems (e.g., message queues, cloud services). This can simplify tasks like uploading results to cloud storage or triggering actions based on external events.
5. **Observability:** DAPR provides built-in observability features, making it easier to monitor and troubleshoot the distributed system.
6. **Reduced Operational Overhead:** DAPR handles many of the complexities of distributed systems (e.g., service discovery, retries, state management), freeing developers from writing boilerplate code.
7. **Cloud-Native Alignment:** DAPR aligns with cloud-native principles, making it easier to deploy and manage the application on cloud platforms like Kubernetes.

4. Roadmap for DAPR Integration

1. **Assessment:** Evaluate the current architecture and identify specific pain points that DAPR can address.
2. **Proof of Concept:** Implement a small-scale proof of concept to demonstrate the feasibility of DAPR integration and its impact on development and operational efficiency.
3. **Incremental Adoption:** Gradually introduce DAPR building blocks into the existing system, starting with the most critical functionalities (e.g., service invocation, state management).
4. **Microservice Decomposition:** Refactor the monolithic application into smaller, independent microservices, leveraging DAPR's building blocks to manage communication and state between them.
5. **Monitoring and Optimization:** Continuously monitor the system's performance and make adjustments to DAPR configuration as needed.

5. Conclusion

Integrating DAPR into the MATLAB Production Server API wrapper project has the potential to significantly simplify development, improve scalability, enhance reliability, and unlock new capabilities. By

leveraging DAPR's powerful building blocks and runtime features, the project can evolve into a modern, cloud-native, distributed application, better positioned to meet the growing demands of the future.