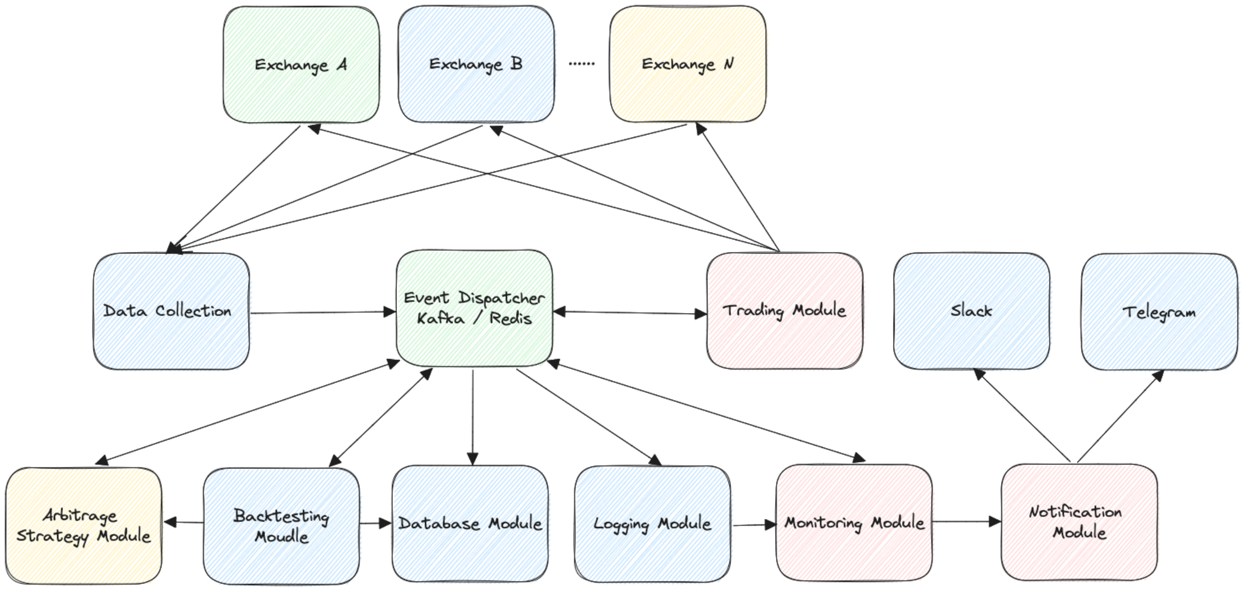
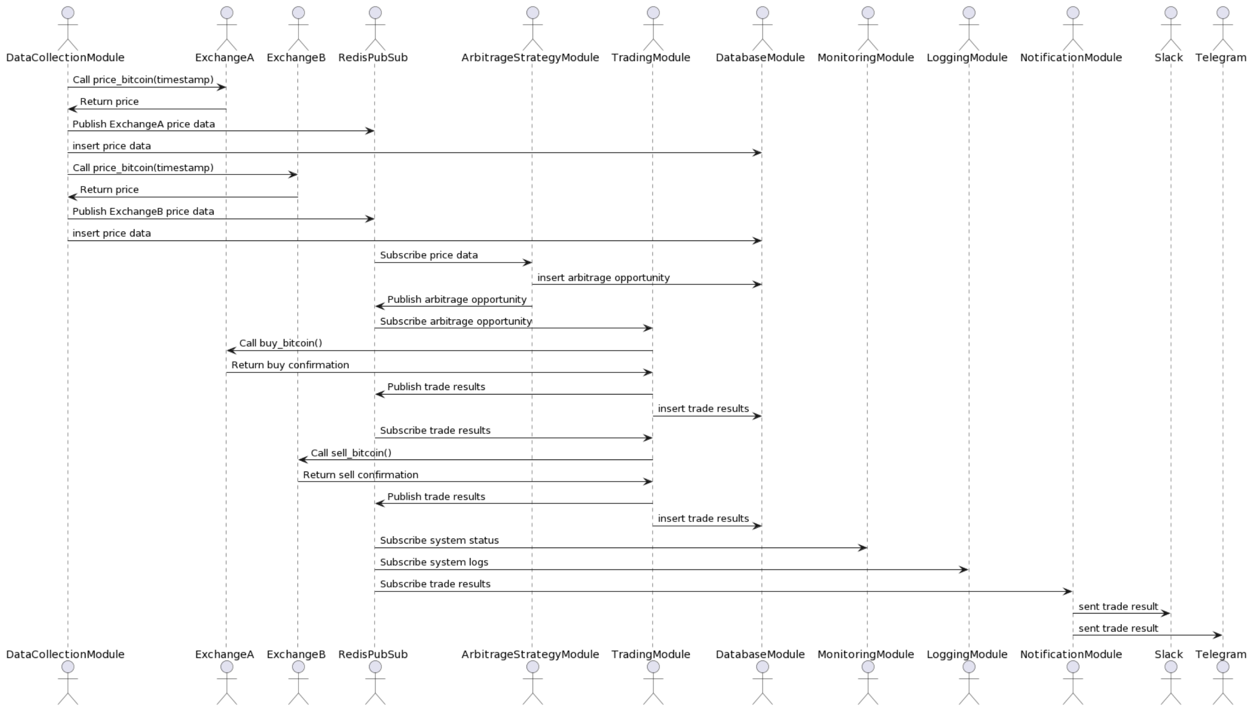
solution

System Design





Module

- **Data Collection Module**: This module is responsible for gathering market data from different exchanges. It interacts with the APIs of each exchange to get the price of Bitcoin at a given timestamp.
- **Redis Pub/Sub**: The Redis Pub/Sub acts as a message broker between different modules. It enables the decoupling of modules by allowing modules to publish messages to Redis channels. Other modules subscribe to these channels and receive the messages, triggering specific operations in response. This mechanism allows for swift propagation of data or commands from one module to another, maintaining a loosely-coupled architecture.
- **Arbitrage Strategy Module**: This module takes the price data from different exchanges and makes decisions on whether there's an arbitrage opportunity or not. If there's an arbitrage opportunity, it sends the signal to the Trading Module via the Event Dispatcher.
- **Trading Module**: The Trading Module acts upon the arbitrage opportunities received from the Arbitrage Strategy Module. It interacts with the APIs of the exchanges to execute buy and sell orders.
- **Database Module**: This module is responsible for persisting data such as price information, arbitrage opportunities, and trade records. It ensures that all the important data is stored for further analysis, backtesting or auditing.
- **Backtesting Module**: The Backtesting Module uses historical data to test the effectiveness of arbitrage strategies. After backtesting, it updates the Arbitrage Strategy Module with improved strategies.
- **Monitoring Module**: The Monitoring Module keeps track of the overall system's performance and health. It detects anomalies or issues in the system and signals the Notification Module in case of problems.
- **Logging Module**: This module is responsible for logging all the system activities. The logged information can be used for debugging purposes, for tracking the system's activities, and for analyzing the performance of the system over time.
- **Notification Module**: The Notification Module sends out alerts based on system status or trade events. This could involve sending messages to a Slack or Telegram group or even triggering some other action when certain conditions are met.

Why using Redis Pub/Sub not Kafka for event-driven:

In the context of our trading system, which involves real-time arbitrage strategies, the choice of message broker becomes critical. We require a solution that provides fast, near real-time data delivery, along with an intuitive API and design that simplifies the process of managing these data flows.

Redis Pub/Sub fits our needs perfectly due to the following reasons:

1. **Real-time data processing**: Redis Pub/Sub provides low latency and high throughput, enabling us to process and react to market data in real-time. This ensures that our arbitrage strategies can be executed promptly, a critical factor in their success.
2. **Simplicity and ease of use**: The design and API of Redis Pub/Sub are simpler compared to some other alternatives, such as Kafka. This means that we can more easily set up and manage our data flows, reducing overhead and freeing up resources to focus on our core trading logic.
3. **Memory-based storage**: Redis stores data in memory, which provides faster data access. Although this means Redis is not as durable or persistent as disk-based solutions like Kafka, this isn't a significant concern for us. We are primarily interested in the immediate consumption of real-time data for our arbitrage strategies, not long-term data retention.
4. **Data Persistence in Database**: While Redis Pub/Sub handles the real-time message brokering, we persist our trading data into a database for historical analysis and backtesting. This provides us the durability and data retention we need, without relying on our message broker to serve this purpose.

In summary, Redis Pub/Sub's strengths align well with the requirements of our trading system, making it the optimal choice for our event-driven architecture.