Case Study: Intra-Exchange Arbitrage Analysis for Arbitrum (ARB) on Binance

Context:

Our High-Frequency Trading (HFT) firm aims to identify and capitalize on price inconsistencies within a single exchange. This case study focuses on analyzing all available trading pairs for Arbitrum (ARB) on Binance to detect potential arbitrage opportunities by comparing their implied USDT values.

Objective:

The goal is to develop a Golang application that monitors the public websocket streams from Binance for all ARB trading pairs. The system should continuously analyze the order books to determine if arbitrage opportunities exist by comparing the effective price of ARB in USDT across different routes.

Problem Statement:

Develop a Golang application that connects to the Binance public websocket API to get order book updates for all trading pairs involving ARB (ARB**/USDT,** ARB**/FDUSD,** ARB**/USDC,** ARB**/TUSD,** ARB**/BTC,** ARB**/ETH,** ARB**/TRY,** ARB**/EUR**). The application should identify instances where a sequence of trades between these pairs on Binance could yield a profit when evaluated against USDT.

Detailed Requirements:

1. **Websocket Connectivity:**
    - Implement a robust connection to the Binance public websocket API to receive order book updates for all listed ARB trading pairs. The system should handle potential connection interruptions and attempt to reconnect.
    - Implement efficient parsing of the incoming websocket messages for these pairs.
2. **Order Book Management:**
    - Maintain an in-memory representation of the top of the order book (best bid and best ask) for all monitored ARB trading pairs from Binance.
    - Ensure these local representations are updated in real-time based on the incoming websocket data.
3. **Implied USDT Value Comparison for Arbitrage Detection:**

    - For each ARB pair, determine the current price of ARB in its quote currency.
    - If the quote currency is not USDT, use other available pairs to calculate the implied USDT value of ARB.
    - Think of ask prices as the price you'd pay to buy the asset, and bid prices as the price you'd receive to sell the asset.
4. For example:

    - For ARB/BTC, if you know the price of ARB in BTC and the price of BTC in USDT, you can calculate the implied USDT price of ARB.

    - For ARB/EUR, if you know the price of ARB in EUR and the price of EUR in USDT (if available), you can calculate the implied USDT price of ARB.

    - Continuously compare the direct ARB/USDT price with the implied USDT prices derived from other ARB pairs. Identify situations where buying ARB through one route and selling it through another in terms of USDT would yield a profit.

    - Consider triangular or multi-leg arbitrage opportunities. For instance:

        - Buy ARB with EUR → Sell ARB for BTC → Sell BTC for USDT → Compare the final USDT amount with the initial implied USDT value of ARB/EUR.

5. **Signal Generation:**

    - When a potential arbitrage opportunity is detected where the calculated profit (in USDT terms, as a percentage of the initial investment) exceeds a predefined threshold (e.g., 0.1%), the system should output a signal.
    - The signal should clearly indicate:
        - The sequence of trades involved (e.g., Buy ARB/EUR, Sell ARB/BTC, Sell BTC/USDT).
        - The estimated profit percentage.
        - The price levels of the involved pairs that enabled the arbitrage.

6. **Order Book Analysis for Amount (Bonus):**

    - For identified arbitrage opportunities, analyze the top few levels of ask and bid prices on the order books for the relevant pairs on Binance to estimate the maximum amount of ARB (or the starting currency of the cycle) that could be traded to realize the arbitrage profit.

7. **Concurrency in Golang:**

    - The solution must effectively utilize Golang's concurrency primitives (goroutines and channels) to manage the websocket connections and the continuous analysis of order book data.

Deliverables:

- A well-structured Golang codebase that implements the requirements.
- Clear documentation explaining the architecture, concurrency handling, and the logic for comparing implied USDT values of ARB across different pairs to detect arbitrage. Clearly state the assumed trading paths for the arbitrage.
- Output demonstrating the system identifying potential arbitrage opportunities, including the estimated tradable amount (for the bonus part).

Evaluation Criteria:

- Correctness of the logic for calculating and comparing implied USDT values of ARB.
- Understanding of how to construct arbitrage trades across different ARB pairs.
- (For bonus) Realistic estimation of the tradable amount based on Binance's order books.
- Efficiency and clarity of the Golang code, especially concurrency.
- Robustness of websocket handling for Binance.
- Clarity of output and documentation.

You are going to use **spot** markets Websockets

https://developers.binance.com/docs/binance-spot-api-docs/web-socket-streams