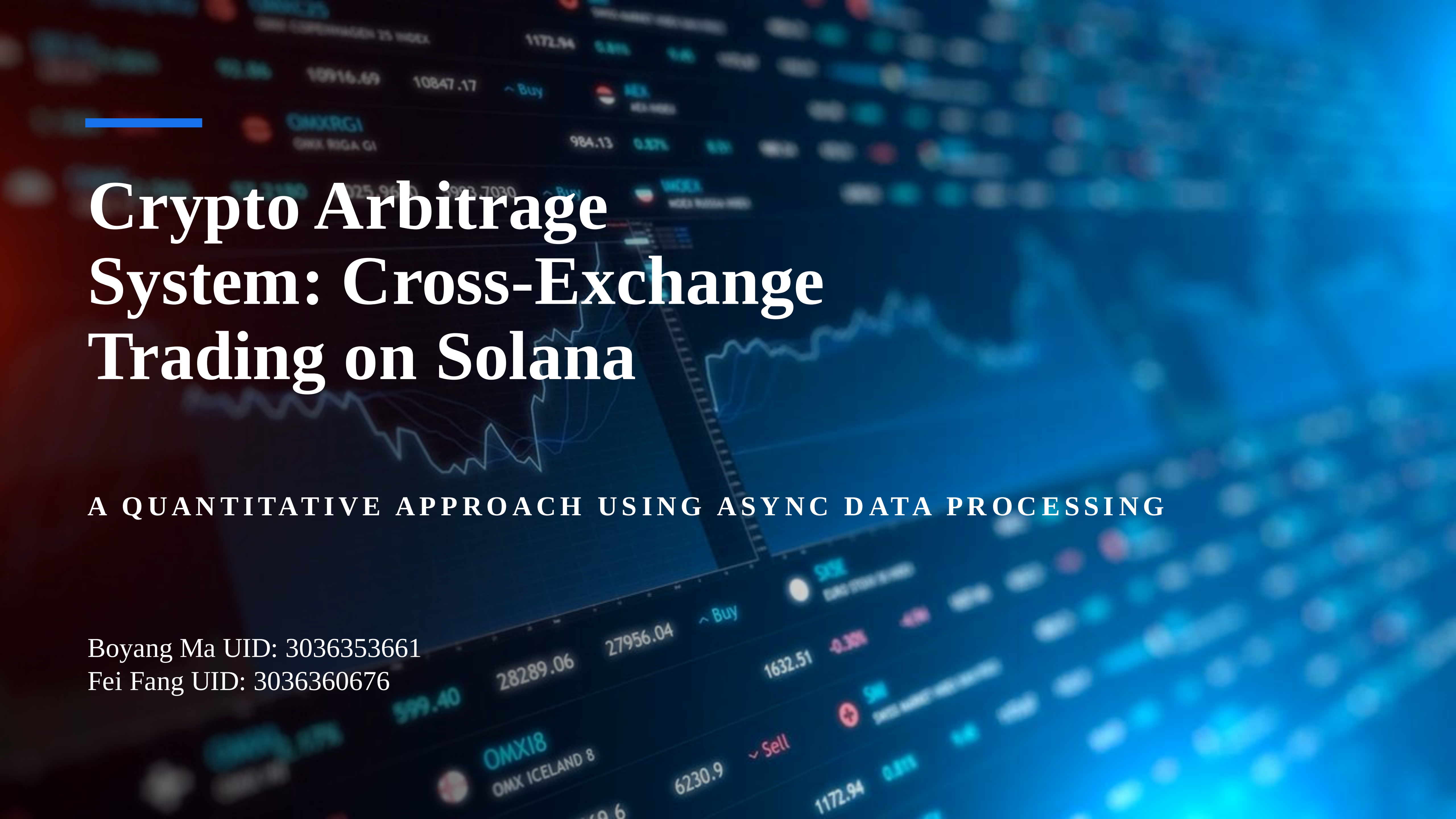


Crypto Arbitrage System: Cross-Exchange Trading on Solana

A QUANTITATIVE APPROACH USING ASYNC DATA PROCESSING

Boyang Ma UID: 3036353661
Fei Fang UID: 3036360676

Table of contents

Introduction

System Overview

Code Breakdown

Case Study

Challenges & Enhancements

Introduction

- **What is Arbitrage?**

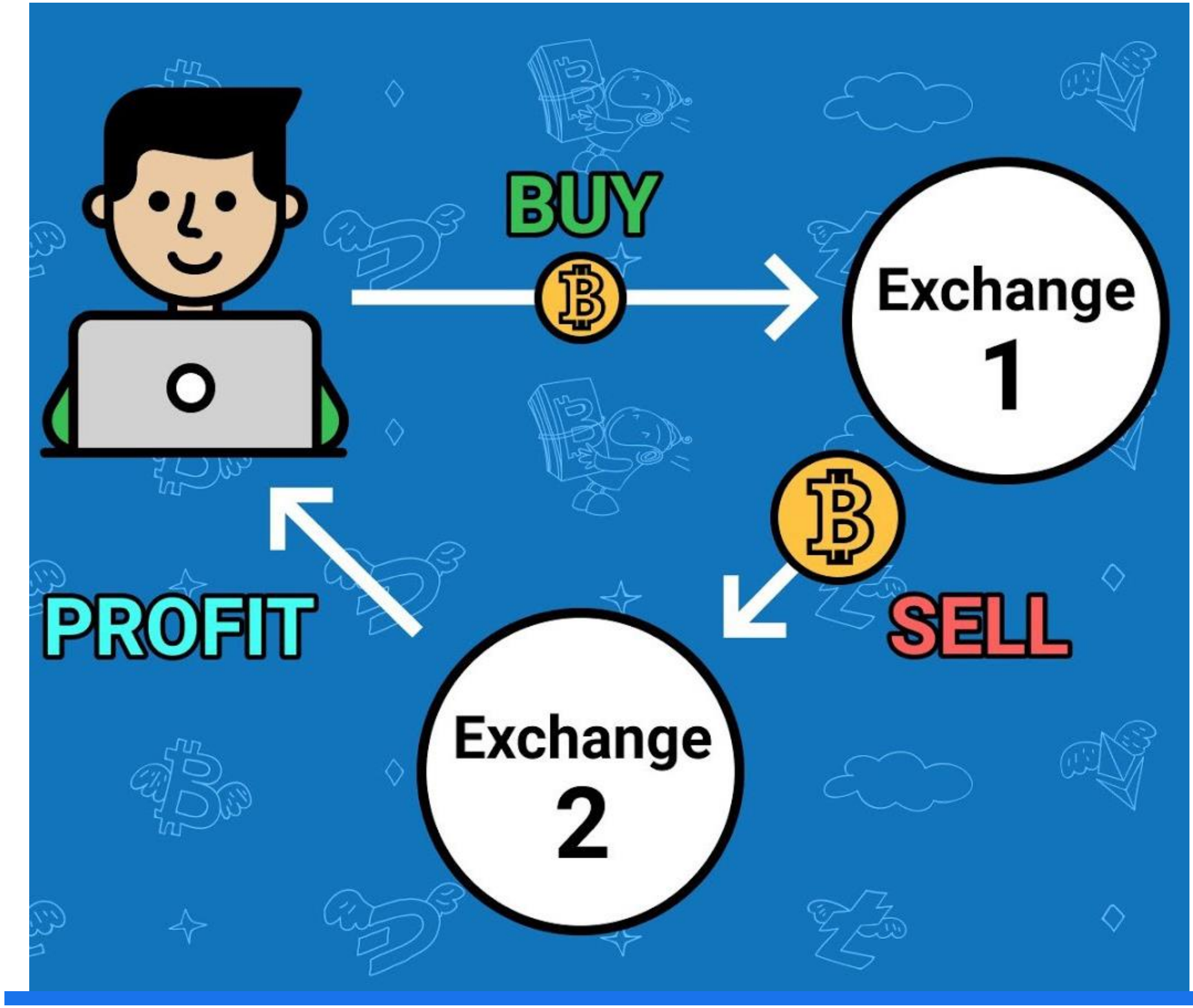
Buying an asset at a lower price on one exchange and selling it at a higher price on another.

- **Why Crypto Arbitrage?**

🌎 **24/7 Trading** allows continuous opportunity.

📊 **Market Fragmentation** leads to price differences.

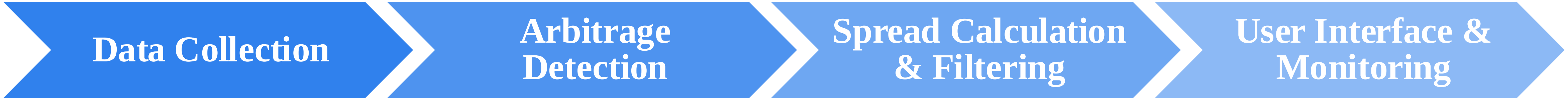
⚡ **High Volatility** creates frequent spreads.



Source: marketfeed Team. (2020, October 14). *What is arbitrage trading & arbitrage funds?* marketfeed. https://www.marketfeed.com/read/en/what-is-arbitrage-trading-what-are-arbitrage-funds

System Overview

- **Monitors OKX & Binance** – Chosen due to high trading volume and liquidity.
- **Supports Multiple Market Types** – Spot, Futures, Swap (Default: Spot).
- **Aggregated Data Sources** – Ticker Data..



Data Collection

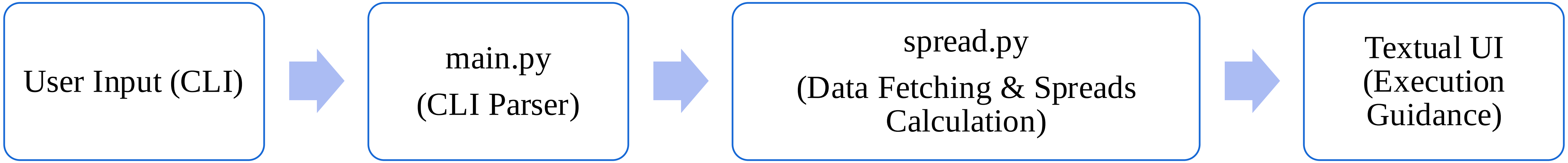
Arbitrage Detection

Spread Calculation & Filtering

User Interface & Monitoring

Code Breakdown

File Name	Function
spread.py	Fetches data, calculates spreads, manages price monitoring.
main.py	CLI interface, user input handling, launches UI.
codeUsageExamples.txt	Provides command-line execution examples.



User Input (CLI)

main.py
(CLI Parser)

spread.py
(Data Fetching & Spreads Calculation)

Textual UI (Execution Guidance)

Data Collection

- WebSockets **reduce latency and improve execution timing**, making arbitrage **more profitable**.

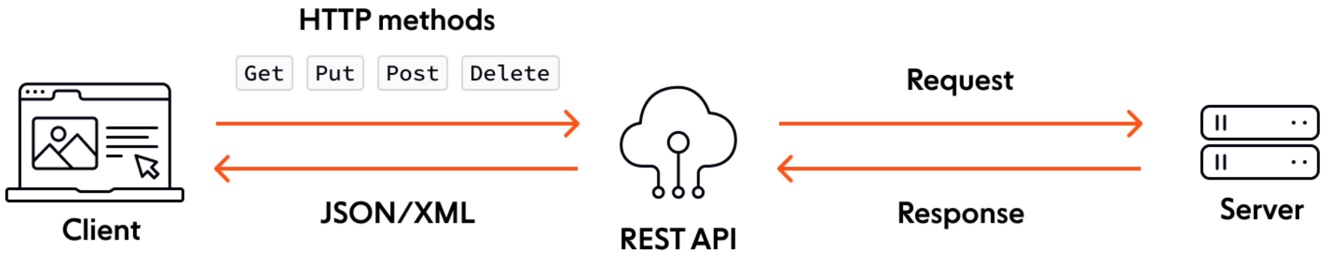


Fig.1 Representational State Transfer (REST)

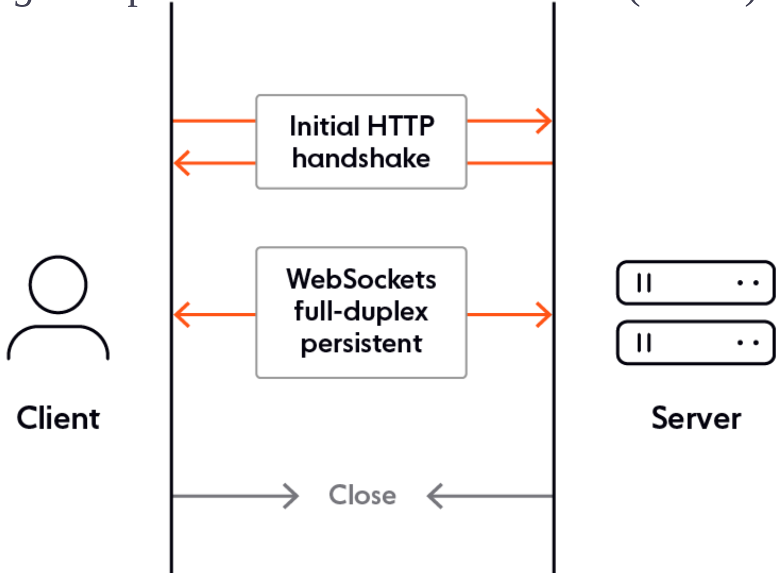


Fig.2 WebSockets

Metric	***REST API***	***WebSockets***
Data Update Speed	300-500ms delay per request	**Instant updates** (sub-50ms latency)
Request Needed	Multiple per second	Single connection, continuous stream
System Resource Usage	High	Low
API Rate Limits	Easily exceeds rate limits with frequent polling	Fewer requests, less likely to be blocked
Data Freshness	Slightly outdated	Always live

Table 1: REST API vs. WebSockets

Source: Ably. (n.d.). *WebSocket vs REST: Understanding the differences*. Ably. Retrieved from https://ably.com/topic/websocket-vs-rest

Data Processing – Speed & Accuracy

Feature	***What It Does***	***Why It Matters***
Latency Tracking	Track **round-trip time (RTT)** & adjust timestamps to sync price across exchanges	Ensures price accuracy for arbitrage
Async Processing	Runs **multiple tasks in parallel** without blocking execution (*asyncio*)	Faster price updates, no lag
Multiprocessing	Offloads calculations to a **separate thread pool** (*ThreadPoolExecutor*)	Prevents slow UI & computation delays

By using these features, our system processes data **~50% faster**, improving arbitrage execution success.

Arbitrage Identification & Spread Calculation

- Spread calculation

$$Spread\ Percentage = \frac{Best\ Bid\ \ - Best\ Ask}{Best\ Bid} * 100$$

- Filtering Out Unprofitable Trades where **Spread % < Fees + Slippage %**.

$$Final\ Spread = Raw\ Spread - (Fees + Slippage)$$

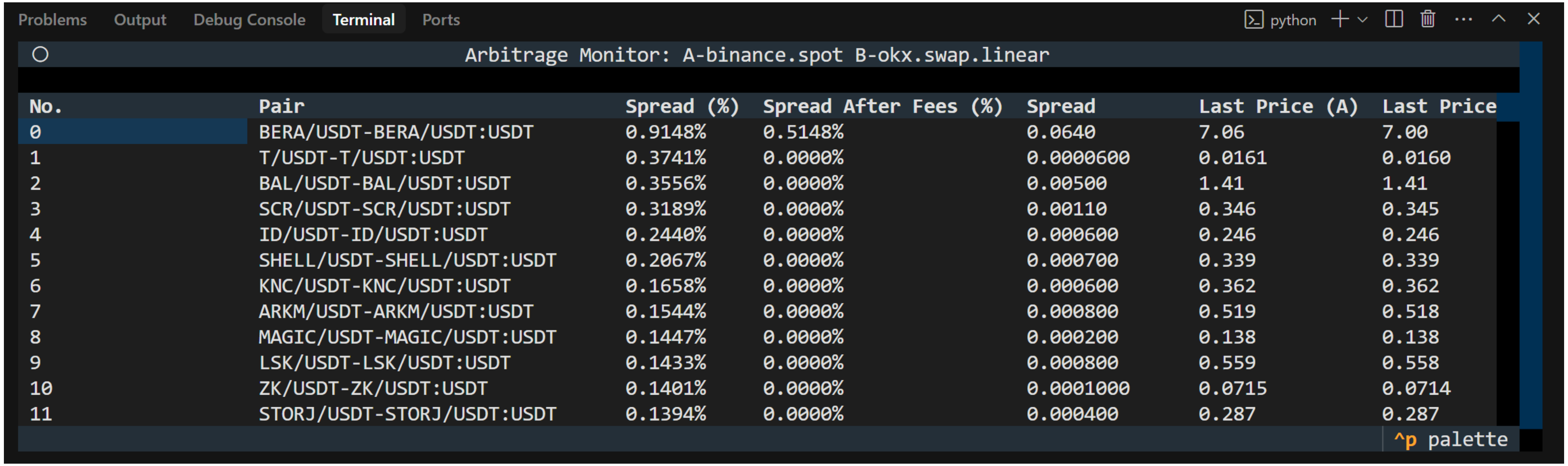
Slippage refers to the difference between the expected price of a trade and the actual price at execution. A conservative estimation is **0.1%–0.15%.**

- Execute the trade where **Final Spread > 0**.

Exchange	*Best Bid*	*Best Ask*	*Raw Spread%*	*Fees (0.1%)*	*Estimated Slippage (0.1%)*	*Final Spread%*
OKX	$95.00	$95.80	+0.84%	-0.2%	-0.2%	+0.44% **Profitable**
Binance	$95.60	$96.10	+0.52%	-0.2%	-0.2%	+0.12% **Non-profitable**

UI & Monitoring System

- Terminal-based UI dynamically updates **latency, execution time, and detailed order book spreads.**
- Displays **top arbitrage opportunities**.
- CLI options allow customization of monitored exchanges.
- Rate-limiting & Proxy Handling prevents API request issues.
- Retries automatically under API failures, Cancels running tasks and safely closes connections when shut down.



```
○                          Arbitrage Monitor: A-binance.spot B-okx.swap.linear

No.                 Pair                           Spread (%)  Spread After Fees (%)  Spread        Last Price (A)  Last Price
0                   BERA/USDT-BERA/USDT:USDT       0.9148%     0.5148%                0.0640        7.06            7.00
1                   T/USDT-T/USDT:USDT             0.3741%     0.0000%                0.0000600     0.0161          0.0160
2                   BAL/USDT-BAL/USDT:USDT         0.3556%     0.0000%                0.00500       1.41            1.41
3                   SCR/USDT-SCR/USDT:USDT         0.3189%     0.0000%                0.00110       0.346           0.345
4                   ID/USDT-ID/USDT:USDT           0.2440%     0.0000%                0.000600      0.246           0.246
5                   SHELL/USDT-SHELL/USDT:USDT     0.2067%     0.0000%                0.000700      0.339           0.339
6                   KNC/USDT-KNC/USDT:USDT         0.1658%     0.0000%                0.000600      0.362           0.362
7                   ARKM/USDT-ARKM/USDT:USDT       0.1544%     0.0000%                0.000800      0.519           0.518
8                   MAGIC/USDT-MAGIC/USDT:USDT     0.1447%     0.0000%                0.000200      0.138           0.138
9                   LSK/USDT-LSK/USDT:USDT         0.1433%     0.0000%                0.000800      0.559           0.558
10                  ZK/USDT-ZK/USDT:USDT           0.1401%     0.0000%                0.0001000     0.0715          0.0714
11                  STORJ/USDT-STORJ/USDT:USDT     0.1394%     0.0000%                0.000400      0.287           0.287
                                                                                                                 ^p palette
```

Challenges & Enhancements

Challenge	*Impact*	*Enhancements*
Latency	Prices change too quickly.	Use **co-location servers & WebSockets**.
Slippage	Prices change before execution.	Break orders into smaller trades.
Withdral Limits	Transfers take too long.	Use market-based execution instead of transferring assets.
API Rate Limits	Too many API calls get blocked.	Implement **smart API throttling**.
Data Sources	Single source of data.	**User-selected** – order book or ticker data

Case Study

Find our code on GitHub!

HTTPS://GITHUB.COM/TONYMA1/PYTHON-ARBITRAGE-MONITOR/

THANK YOU!
THANK YOU!
THANK YOU!