**Порівняння реалізацій систем збору інформації з крипто бірж, у реальному часі, на мовах програмування С++, JavaScript та Python.**

*Дуда В. О. 1, Терентьєв О.М. 2.*

*1 Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2 Інститут телекомунікацій і глобального інформаційного простору Національної академії наук України.*

[*dudavolodimir@gmail.com*](mailto:dudavolodimir@gmail.com) *,* [*o.terentiev@gmail.com*](mailto:o.terentiev@gmail.com)

**Вступ**

З розвитком фінтеху у просторі всесвітньої економіки з'явилися сервіси для торгівлі криптовалютами та іншими об'єктами з унікальним криптографічним підписом. Для обміну цих елементів на інші активи, були створені криптобіржі.

Binance [1] є найбільшою у світі криптобіржею за обсягом торгів, із щоденним обсягом торгів, станом на серпень 2022 року, 76 мільярдів доларів США та 90 мільйонами клієнтів у всьому світі. Платформа зарекомендувала себе як надійний учасник криптопростору, де користувачі можуть купувати, продавати та зберігати свої цифрові активи, а також отримувати доступ до понад 350 існуючих криптовалют і тисяч торгових пар [1].

На ринку криптовалют щомісячні торгові об’єми складають трильйони доларів, оскільки трейдери користуються можливостями швидкого отримання прибутку незалежно від напрямку тренду. Завдяки деривативним контрактам [2] трейдери можуть заробляти на коливаннях цін, відкриваючи лонг (купив дешевше – продав дорожче) або шорт (продав дорожче – купив дешевше).

За торгову пару було роглянуто найпопулярнішу – BTC/USDT [3] (USDT – це стейблкоїн, ціна якого еквівалентна долару). Її торговий об'єм сягає від 300 тис. до 1.5 млн монет щоденно, що еквівалентно 6 млрд. – 42 млрд. доларів. Для відображення ціни певної валюти та попит на неї можна скористатись лише першими цінами на купівлю та продаж. На рис. 1 наведено скріншот зміни добових об’ємів.



Рис. 1. Приклад свічки цін та об'ємів продаж за період останніх 5 місяців. Верхня частина - зміни ціни, а знизу гістограма об’ємів криптоактиву за період часу.

**Реалізація системи для доступу даних біржи.**

Будь яка система прийняття рішень, що призначена для роботи на криптобіржі, потребує застосунки, які найшвидше передадуть дані від джерела до аналітичного ядра у повному обсязі.

Зазвичай для доступу до даних використовується REST [4] запит, проблема HTTP/HTTPS [4] протоколу в тому, що він повільний. Він працює таким чином, що спочатку сервер та клієнт встановлюють TCP підключення, потім на стороні клієнта формується запит за стандартом протоколу. Цей запит надсилається на сервер, який формує відповідь таким же чином та надсилає її, в кінці зв`язок розривається. Це не ефективно, так як на кожний запит формує нове підключення, а дані потрібно отримувати весь час. Для вирішення цієї проблеми формується одне підключення та по ньому йдуть усі дані. Зазвичай в реалізації цієї системи використовується звичайне TCP підключення, чи його аналоги такі як FIX (теж саме, але повідомлення описані за даним протоколом) та Websockets [5] – стандарт, який прийшов з HTML5 та використовується у багатьох WEB-сервісах.

Будь яка біржа має обмеження на кількість запитів за певний період. Зазвичай швидкості зміни цін значно перевищують їх. При подібній ситуації будь який запит з цього пристрою буде ігноруватись. Єдиним якісним засобом отримання інформації є Websockets підключення [6].

В рамках дослідження підключення до біржі було реалізовано на трьох різних мовах програмування, з метою порівняння з точки зору ефективності затрат ресурсів, які наведені у таблиці 1.

Таблиця 1. Порівняння бібліотек на різних мовах програмування.

|  |  |  |  |
| --- | --- | --- | --- |
| Мова програмуван-ня | Компільована / інтерпретована | Синхронна / асинхронна | Бібліотека |
| C++ | компільована | синхронна | Ixwebsocket[7] |
| JavaScript | інтерпретована | асинхронна | ws (для Node.js)[8] |
| Python | інтерпретована | асинхронна | aiohttp[9] |

При проведенні досліджень було виявлено, що синхронна реалізація на мові Python – неможлива, так як дані на наступній ітерації обробки вже були не актуальні.

Усі три реалізації були запущені на двох серверах, які розміщені у різних хостинг провайдерах, що наведені у таблиці 2.

Таблиця 2. Порівняння серверів на різних платформах.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Хостинг провай-дер | Місце розта-шування серверу | Процесор | Кількість RAM пам'яті | Архіте-ктура процесора | Швидкість інтернет підключення | Ціна за місяць |
| Hetzner [10] | Німеччи-на | AMD EPYC 7502P 32-Core Processor | 750 GB | x86\_64 | 10 Gbps | 550 – 650 $ |
| AWS [11] | Токіо, Японія | 8 x cors of AWS Graviton2 processor | 64 GB | arm64 | 10 Gbps | 600 - 1000 $ |

Нижче наведений код програми реалізації доступу до біржи на мові програмування Node.js:

const { WebSocket } = require('ws');

const wsUrl = 'wss://fstream.binance.com/ws';

const ws = new WebSocket(wsUrl);

ws.on('ping', () => ws.pong());

ws.on('open', () => {

ws.send(JSON.stringify({

method: "SUBSCRIBE",

params: ["btcusdt@bookTicker"],

id: 1

}));

});

ws.on('message', (data) => {

const tik = JSON.parse(data);

const tradeTime = tik.T;

if (tradeTime) {

const ts = (new Date()).getTime();

console.log(`${tradeTime};${ts};${ts - tradeTime}`);

}

})

**Висновки**

В рамках даного дослідження, всі реалізовані програми, запускалися як різні процеси, одночасно на двох серверах. Час реєстрації затримки було реєстровано відносно часу виставлення чи проходження ордеру на самій біржі, та часу отримання даних на клієнті. Усі вихідні данні виводились у STDOUT, які зберігались потім у файлі формату csv.

Середнє значення затримки на Hetzner [11] становить 148 мс, а на AWS [10] – 5 мс. Можна зробити висновок, що на швидкість отримання даних найбільше впливає мережева затримка, яка залежить від розташування пристрою, бо чим ближче до біржі, сервери якої знаходяться в Японії, то менша затримка на отримання даних.

Головний результат дослідження – немає різниці на якій мові пишеться модуль потокового отримання даних, а головним є місце його розташування відносно біржі. Крім того, на швидкість отримання даних може впливати архітектура ARM процесора.

Таким чином, для реалізації схожих систем, не потрібні висококваліфіковані кадри, які можуть написати код на швидких компільованих мовах. Тобто з даного списку тестів, можу впевнено виділити JavaScript, як найкращий інструмент для вирішення таких задач, так як він простий в навчанні та легкий в налаштуванні асинхронних подій, крім того є вже багато готових бібліотек та застосунків, які спрощують інтеграцію в будь яку систему.

З порівняння серверів можна побачити що Hetzner [11] потужніший за AWS [10]. За таких умов для реалізації важких систем треба оцінювати рентабельність, так як на AWS [10] хостинг коштує значно дорожче.

**Список літератури**

1. What Is Binance? URL: <https://coinmarketcap.com/exchanges/binance/> (дата звернення: 23.10.2022).
2. Початок роботи на Binance Futures. URL: https://www.binance.com/uk-UA/blog/futures (lдата звернення: 23.10.2022).
3. Інструмент торгівлі фьютчерсами на Binance. URL: <https://www.binance.com/uk-UA/futures/BTCUSDT> (дата звернення: 23.10.2022).
4. Hypertext Transfer Protocol (HTTP). URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP> (дата звернення: 23.10.2022).
5. The WebSocket API (WebSockets). URL: <https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API> (дата звернення: 23.10.2022).
6. Binance documentation - Websocket Market Streams. URL: <https://binance-docs.github.io/apidocs/futures/en/#websocket-market-streams> (дата звернення: 23.10.2022).
7. IXWebSocket (C++ library for WebSocket). URL: <https://github.com/machinezone/IXWebSocket> (дата звернення: 23.10.2022).
8. ws: a Node.js WebSocket library. URL: <https://www.npmjs.com/package/ws> (дата звернення: 23.10.2022).
9. aiohttp works with client websockets out-of-the-boxю URL: <https://docs.aiohttp.org/en/stable/client_quickstart.html#websockets> (дата звернення: 23.10.2022).
10. Хостинг Hetzner - <https://www.hetzner.com/> (дата звернення: 23.10.2022).
11. Хостинг Amazon AWS. URL: <https://aws.amazon.com/> (дата звернення: 23.10.2022).