МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «**АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ ТА РЕАЛІЗАЦІЇ АРБІТРАЖНИХ СТРАТЕГІЙ НА КРИПТОВАЛЮТНИХ ДЕРИВАТИВНИХ РИНКАХ»**

Виконав: студент 2 курсу, групи 8.1224-он
спеціальності 122 Комп'ютерні науки
освітньої програми Комп'ютерні науки
(назва освітньої програми)
А. Ю. Підгорний
(ініціали та прізвище)
Керівник ректор ЗНУ, професор, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н., Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Підгорному Артему Юрійовичу

1. Тема роботи Автоматизована система виявлення та реалізації арбітражних стратегій на криптовалютних деривативних ринках

керівник роботи Шило Галина Миколаївна, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 326-с

2. Строк подання студентом роботи 8 червня 2026 року

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Аналіз сучасного стану криптовалютних деривативних ринків та існуючих програмних рішень.
3. Розробка математичних моделей виявлення арбітражних можливостей та обґрунтування технологічного стеку.
4. Програмна реалізація автоматизованої системи моніторингу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20 квітня 2025 року

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	25.04. 2025	Виконано
2.	Збір вихідних даних.	25. 06. 2025	Виконано
3.	Обробка методичних та теоретичних	03.11.2025	Виконано
	джерел.		
4.	Розробка першого та другого розділу.	27.02.2026	Виконано
5.	Розробка третього розділу.	25.04.2026	Виконано
6.	Оформлення та нормоконтроль	05.06.2026	Виконано
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	20.06.2026	

Студент _____ А.Ю. Підгорний
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.М. Шило
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Автоматизована система виявлення та реалізації арбітражних стратегій на криптовалютних деривативних ринках»: 113 с., 16 рис., 11 табл., 23 джерел, 4 додатки.

АВТОМАТИЗАЦІЯ ТОРГІВЛІ, АРБІТРАЖ, БАЗИСНИЙ АРБІТРАЖ, БЕЗСТРОКОВІ Ф'ЮЧЕРСИ, КРИПТОВАЛЮТИ, МЕ, МОНІТОРИНГ РЕАЛЬНОГО ЧАСУ, ОНЛАЙН-СПОВІЩЕННЯ, СТАВКА ФІНАНСУВАННЯ, ТЕЛЕГРАМ-БОТ, ФАНДІНГОВИЙ АРБІТРАЖ.

Об'єкт дослідження – процес виявлення та автоматизованої реалізації арбітражних можливостей на криптовалютних деривативних ринках.

Предмет дослідження – методи, моделі, алгоритми та програмно-технологічні засоби побудови розподіленої мультибіржової системи моніторингу арбітражних можливостей різних типів з інтеграцією джерел on-chain даних.

Мета роботи – розробити та обґрунтувати архітектуру і програмну реалізацію автоматизованої екосистеми виявлення арбітражних можливостей на криптовалютних деривативних ринках та експериментально оцінити її ефективність за метриками точності виявлення, часу реакції та частки істинно-позитивних сигналів.

Методи дослідження – системний аналіз, метод аналізу ієрархій Т. Сааті, математичне моделювання, об'єктно-орієнтоване та подієво-орієнтоване програмування, методи математичної статистики (описова статистика, t-критерій Стьюдента, критерій χ^2 Пірсона).

Наукова новизна одержаних результатів: уперше запропоновано двоконтурну архітектуру моніторингу арбітражних можливостей із субсекундною роздільною здатністю; удосконалено алгоритм виявлення спот-ф'ючерсного арбітражу шляхом інтеграції перевірки умов реалізованості; розроблено ескалаційну схему сповіщень з диференційованими каналами

доставки; подальшого розвитку набуло застосування on-chain даних як випереджувального індикатора.

Практичне значення результатів – розроблено функціонуючу програмну екосистему з семи модулів-сканерів, розгорнуту на хмарному сервері Amazon Web Services, з місячною вартістю інфраструктури близько 30 доларів США. Експериментальна апробація на реальних ринкових даних протягом 14 діб підтвердила ефективність запропонованих рішень. Результати роботи можуть використовуватися учасниками криптовалютних ринків для систематизованого прийняття торгових рішень, а методологічні підходи – застосовуватися при розробці аналогічних систем моніторингу фінансових ринків. Запропоновані рішення мають потенціал подальшого розвитку через інтеграцію ML-класифікаторів, модулів бектестингу, розширення підтримуваних бірж та джерел даних.

SUMMARY

Master's Qualifying Theses «Automated System for Detection and Execution of Arbitrage Strategies in Cryptocurrency Derivatives Markets»: 113 pages, 16 figures, 11 tables, 23 sources, 4 supplements.

AUTOMATED TRADING, ARBITRAGE, BASIS ARBITRAGE, CRYPTOCURRENCIES, FUNDING ARBITRAGE, FUNDING RATE, ONLINE NOTIFICATIONS, PERPETUAL FUTURES, REAL-TIME MONITORING, TELEGRAM BOT.

The object of the study is the process of detecting and automatically implementing arbitrage opportunities in cryptocurrency derivatives markets.

The subject of the research is the methods, models, algorithms, and software-technological tools for building a distributed multi-exchange system for monitoring various types of arbitrage opportunities with integration of on-chain data sources.

The aim of the study is to develop and substantiate the architecture and software implementation of an automated ecosystem for detecting arbitrage opportunities in cryptocurrency derivatives markets, and to experimentally evaluate its effectiveness using detection accuracy, reaction time, and true-positive rate metrics.

The methods of research are systems analysis, the analytic hierarchy process by T. L. Saaty, mathematical modeling, object-oriented and event-driven programming, and methods of mathematical statistics (descriptive statistics, Student's t-test, Pearson's χ^2 test).

Scientific novelty of the obtained results: for the first time, a two-loop architecture for monitoring arbitrage opportunities with sub-second resolution is proposed; the algorithm for detecting spot-futures arbitrage is improved by integrating the verification of realizability conditions; an escalation notification scheme with differentiated delivery channels is developed; the application of on-

chain data as a leading indicator of future price movements on centralized exchanges has received further development.

Practical significance of the results – a functioning software ecosystem of seven scanner modules has been developed and deployed on the Amazon Web Services cloud server, with a monthly infrastructure cost of about 30 US dollars. Experimental validation on real market data for 14 days confirmed the effectiveness of the proposed solutions. The results of the work can be used by cryptocurrency market participants for systematic trading decision-making, and the methodological approaches can be applied in the development of similar financial market monitoring systems. The proposed solutions have potential for further development through the integration of ML classifiers, backtesting modules, and expansion of supported exchanges and data sources.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 6
Скорочення та умовні познаки 10
Вступ 12
1 Аналіз предметної області та постановка задачі 16
1.1 Сучасний стан криптовалютних деривативних ринків 16
1.2 Типологія арбітражних стратегій 18
1.3 Огляд існуючих програмних рішень 25
1.4 Джерела ринкових даних 29
1.5 Постановка задачі та формалізація вимог 32
2 Математичні моделі та архітектурне проєктування 36
2.1 Математична модель базису між спотовим та ф’ючерсним ринком 36
2.2 Математична модель фандінг-арбітражу 39
2.3 Критерії реалізованості арбітражного сигналу 42
2.4 Обґрунтування технологічного стеку методом аналізу ієрархій 44
2.5 Архітектура системи: модульний підхід 50
2.6 Двоконтурна модель моніторингу 53
2.7 Підсистема ескалаційних сповіщень 56
3 Програмна реалізація та експериментальні дослідження 59
3.1 Загальна архітектура реалізованої системи 59
3.2 Флагманський модуль SpreadMonitor 61
3.3 Реалізація двоконтурної моделі моніторингу 63
3.4 Алгоритм фільтрації хибно-позитивних сигналів 68
3.5 Реалізація підсистеми ескалаційних сповіщень 72
3.6 Інші модулі екосистеми 74
3.7 Налаштування експериментальної апробації 80

3.8 Статистичний аналіз результатів апробації.....82
3.9 Обмеження та подальший розвиток.....87
Висновки.....91
Перелік посилань.....95
Додаток А Лістинг ключових алгоритмів.....97
Додаток Б Конфігураційні файли.....101
Додаток В Інтерфейс Telegram-ботів.....105
Додаток Г Статистичні таблиці результатів апробації.....111

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	База даних
ЄС	Європейський Союз
ЗНУ	Запорізький національний університет
MAI	Метод аналізу ієрархій
СКБД	Система керування базами даних
API	Application Programming Interface
AWS	Amazon Web Services
CEX	Centralized Exchange
CI	Consistency Index
CR	Consistency Ratio
CSV	Comma-Separated Values
DEX	Decentralized Exchange
F1	F1-score (гармонічне середнє precision і recall)
FN	False Negative
FP	False Positive
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
MiCA	Markets in Crypto-Assets Regulation (ЄС)
ML	Machine Learning
OI	Open Interest
REST	Representational State Transfer
RI	Random Index (у методі MAI)
SEC	Securities and Exchange Commission (США)
SHA	Secure Hash Algorithm
SQL	Structured Query Language
TN	True Negative
TP	True Positive
UML	Unified Modeling Language

URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VPS	Virtual Private Server
WAL	Write-Ahead Logging
WS	WebSocket

ВСТУП

Стрімкий розвиток криптовалютних деривативних ринків протягом 2020–2025 рр. призвів до формування високофрагментованого фінансового середовища, у якому одночасно функціонують понад двісті централізованих бірж та десятки децентралізованих протоколів. За даними галузевих аналітиків [1, 2], середньоденний обсяг торгів безстроковими ф'ючерсами (perpetual futures) перевищує 100 млрд доларів США, що становить понад 75 % загального криптовалютного обороту. Подібна фрагментація ринку зумовлює виникнення стійких цінових неефективностей між майданчиками [3, 4], які створюють об'єктивні передумови для систематичного арбітражу.

Незважаючи на наявність значної кількості комерційних рішень для алгоритмічної торгівлі (Hummingbot, 3Commas, Cryptohopper та інших), переважна частина існуючих систем зосереджена на класичних формах міжбіржового арбітражу та не враховує специфіки деривативних інструментів – зокрема механізму фінансування (funding rate) безстрокових ф'ючерсів, обмежень доступності крипто-позик і статусу депозитно-вивідних шлюзів. Це призводить до високої частки хибно-позитивних сигналів та зниження практичної реалізованості виявлених арбітражних можливостей.

Актуальність дослідження зумовлена необхідністю розробки інтегрованої програмної системи, яка поєднувала б виявлення різнотипних арбітражних стратегій з перевіркою умов їх реалізованості в режимі реального часу, що дозволить підвищити ефективність прийняття торгових рішень в умовах високофрагментованого криптовалютного ринку.

Дослідження виконане відповідно до напряму наукової діяльності кафедри комп'ютерних наук Запорізького національного університету в галузі розробки інформаційних систем та автоматизації прийняття рішень.

Мета дослідження – розробити та обґрунтувати архітектуру і програмну реалізацію автоматизованої екосистеми виявлення арбітражних можливостей

на криптовалютних деривативних ринках та експериментально оцінити її ефективність за метриками точності виявлення, часу реакції та частки істинно-позитивних сигналів.

Для досягнення поставленої мети визначено такі **завдання**:

- проаналізувати сучасний стан криптовалютних деривативних ринків і типологію арбітражних стратегій;
- дослідити математичні моделі спот-ф'ючерсного, фандінгового та просторового арбітражу та формалізувати критерії виявлення сигналів;
- обґрунтувати вибір технологічного стеку методом аналізу ієрархій Т. Сааті;
- спроєктувати модульну архітектуру системи з централізованим керуванням процесами та підсистемою сповіщень;
- реалізувати сім спеціалізованих модулів-сканерів для виявлення різних типів арбітражних можливостей;
- розробити ескалаційну схему сповіщень з підтримкою аудіо-сповіщень критичної пріоритетності;
- провести експериментальну апробацію системи на реальних ринкових даних;
- виконати статистичний аналіз результатів та обґрунтувати ефективність розробленого рішення.

Об'єкт дослідження – процес виявлення та автоматизованої реалізації арбітражних можливостей на криптовалютних деривативних ринках.

Предмет дослідження – методи, моделі, алгоритми та програмно-технологічні засоби побудови розподіленої мультибіржової системи моніторингу арбітражних можливостей різних типів з інтеграцією джерел on-chain даних.

Методи дослідження. У роботі використано: *системний аналіз* для структуризації задачі арбітражу; *метод аналізу ієрархій Т. Саати* для обґрунтування вибору технологічного стеку; *математичне моделювання* для

формалізації арбітражних спредів та критеріїв виявлення сигналів; *об'єктно-орієнтоване та подієво-орієнтоване програмування* для реалізації модулів системи; методи *математичної статистики* (описова статистика, t-критерій Стьюдента, критерій Пірсона) для обробки експериментальних результатів.

Наукова новизна одержаних результатів полягає у такому:

- **уперше запропоновано** двоконтурну архітектуру моніторингу арбітражних можливостей, що поєднує повільний цикл повного сканування ринку зі швидким циклом цілеспрямованого спостереження за критичними монетами з субсекундною роздільною здатністю, що дозволяє знизити час реакції на появу необхідних умов реалізації сигналу;
- **удосконалено** алгоритм виявлення спот-ф'ючерсного арбітражу шляхом інтеграції перевірки доступності крипто-позик та статусу депозитно-вивідних шлюзів як необхідних умов реалізованості сигналу, що зменшує частку хибно-позитивних сповіщень;
- **розроблено** ескалаційну схему сповіщень, яка диференціює канали доставки (Telegram, аудіо-сповіщення Pushover) залежно від рівня критичності події та її часової чутливості;
- **подальшого розвитку набуло** застосування on-chain даних як випереджувального індикатора майбутніх цінових рухів на централізованих біржах через інтеграцію блокчейн-моніторингу з агрегованими даними щодо funding rate централізованих бірж.

Практичне значення одержаних результатів полягає у створенні функціонуючої програмної екосистеми, розгорнутої на хмарному сервері Amazon Web Services та такої, що забезпечує виявлення арбітражних можливостей на восьми та більше криптовалютних біржах у режимі реального часу. Результати роботи можуть використовуватися учасниками криптовалютних ринків для систематизованого прийняття торгових рішень, а методологічні підходи – застосовуватися при розробці аналогічних систем моніторингу фінансових ринків.

Апробація результатів роботи. Основні положення та результати дослідження доповідалися автором на Сімнадцятій Всеукраїнській, двадцять четвертій регіональній науковій конференції молодих дослідників «Актуальні проблеми математики та інформатики» (м. Запоріжжя, 24 квітня 2026 р., секція №2 «Інформатика та новітні комп'ютерні технології») [5].

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та додатків. Загальний обсяг роботи становить 116 сторінок, у тому числі: основний текст – 96 сторінки, 16 рисунків, 11 таблиць, 23 найменування використаних джерел, 4 додатки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан криптовалютних деривативних ринків

Криптовалютний дериватив – фінансовий інструмент, ціна якого є похідною від ціни криптовалютного активу [6]. На сучасному ринку домінують три категорії деривативів: класичні строкові ф'ючерси з фіксованою датою закінчення, безстрокові ф'ючерси (perpetual futures, або perpetual swaps) і опціони [7]. Першим інституційним продуктом цього типу стали ф'ючерси на Bitcoin, запущені на Чиказькій товарній біржі (CME) у грудні 2017 року. Принципово новим класом інструменту став безстроковий ф'ючерс, уперше представлений біржею BitMEX у 2016 році. На відміну від класичного ф'ючерсу, безстроковий контракт не має дати закінчення та утримується близько спотової ціни за рахунок механізму періодичного фінансування (funding rate), що компенсує відсутність природної точки конвергенції цін [1].

За даними галузевих оглядів станом на 2024–2025 рр., безстрокові ф'ючерси сформували понад 75 % загального обороту криптовалютного ринку, а їх середньоденний обсяг торгів перевищує 100 млрд доларів США. Основну частку (близько 50 % загального обороту) тримає біржа Binance; інші значущі майданчики – OKX, Bybit, Bitget, Gate.io, MEXC. Сумарний відкритий інтерес (open interest, OI) безстрокових ф'ючерсів на Bitcoin на початок 2025 року перевищував 35 млрд доларів США, а на Ethereum – 18 млрд доларів США. Поряд із централізованими біржами активно розвиваються децентралізовані протоколи безстрокових ф'ючерсів (dYdX, GMX, Hyperliquid, Lighter, Paradex, Pacifica), які реалізують клірингові механізми безпосередньо у блокчейні.

Ключова особливість безстрокових ф'ючерсів – механізм фінансування. Замість дати поставки, що забезпечує збіжність ціни деривативу до спотової у класичних ф'ючерсів, у безстрокових кожні 4 або 8 годин (залежно від біржі) відбувається периодичний обмін виплатами між довгими та короткими

позиціями. Якщо ставка фінансування позитивна – довгі позиції сплачують короткими; якщо негативна – навпаки. Ставка розраховується як функція премії ф'ючерсу над індексною (спотовою) ціною з урахуванням процентної ставки за маржинальною валютою (зазвичай близько 0,01 % за 8 годин). У типовому ринковому стані абсолютне значення ставки не перевищує 0,1 % за період. У моменти суттєвих ринкових дисбалансів – масових ліквідацій, новинних шоків, монетарних подій – ставка може досягати декількох відсотків за період, що створює короткострокові арбітражні можливості з річною дохідністю еквівалентом понад 100 %.

Конкретні приклади ринкових подій 2023–2025 рр., що супроводжувалися екстремальними значеннями ставок фінансування та базисних аномалій: ліквідаційний каскад на ринку Ethereum у січні 2024 року з обсягом ліквідованих позицій понад 800 млн доларів США за 4 години; флеш-крах ринку альткоїнів у травні 2024 року, коли ціни активів другого ешелону на низьколіквідних біржах розходилися з основними майданчиками на 5–15 %; масштабна корекція після оголошення про відстрочку рішення SEC щодо ETF на Ethereum у березні 2024 року. Кожна з цих подій породжувала десятки реалізуємих арбітражних можливостей, які могли бути виявлені та використані лише за умови наявності автоматизованої системи моніторингу.

Структура криптовалютного ринку характеризується високим рівнем фрагментації. За даними CoinMarketCap, на 2025 рік функціонують понад 200 централізованих та сотні децентралізованих торгових майданчиків, кожен з яких має власну книгу заявок, набір лістингованих контрактів та політики маржинальної торгівлі. Як наслідок, у момент часу одна й та сама пара (наприклад, BTC/USDT-PERP) може мати помітно різні ціни на різних біржах, особливо для активів з нижчою капіталізацією. Для топ-10 монет за капіталізацією типова розбіжність цін між основними майданчиками не перевищує 0,1–0,3 %, проте для активів за межами топ-50 розбіжність може досягати 1–5 % і більше у періоди підвищеної ринкової активності. Ця

фрагментація, а також відсутність централізованого клірингу, створюють об’єктивні передумови для регулярного виникнення цінових неефективностей.

Регуляторний фон останніх років суттєво вплинув на структуру ринку. Введення Регламенту ЄС про ринки криптоактивів (MiCA) [8], що набув повної сили з грудня 2024 року, формалізувало вимоги до криптовалютних посередників у Європейському Союзі. Посилення позиції Комісії з цінних паперів і бірж США (SEC) щодо класифікації більшості альткоїнів як цінних паперів і подальші судові процеси проти провідних бірж (Binance, Coinbase, Kraken) у 2023–2024 рр. спричинили часткову міграцію учасників та обмеження доступу резидентів США до деяких бірж. Крах біржі FTX у листопаді 2022 року також змінив структуру ринку [2]: значна частина ліквідності перемістилася на Bybit, OKX та децентралізовані протоколи. Усі ці зрушення посилили фрагментацію ринку та різницю в ліквідності між майданчиками, що, у свою чергу, підвищило частоту виникнення арбітражних можливостей.

Підсумовуючи, сучасний ринок криптовалютних деривативів характеризується великими обсягами торгів, високою фрагментацією та наявністю специфічних механізмів (funding rate, маржинальна торгівля, крипто-позики), що принципово відрізняє його від традиційних фінансових ринків та обумовлює необхідність розробки спеціалізованих програмних інструментів моніторингу арбітражних можливостей.

1.2 Типологія арбітражних стратегій

Під арбітражем у фінансовій теорії розуміють отримання прибутку без ринкового ризику за рахунок одночасної купівлі та продажу однакового або еквівалентного активу на різних ринках за різними цінами [9]. Класична теорема арбітражного ціноутворення передбачає, що в умовах ефективного ринку арбітражні можливості існують лише короткочасно і швидко закриваються діями учасників ринку. На криптовалютних ринках, через

зазначені вище особливості (фрагментацію, регуляторні бар’єри, обмеження транзакційної інфраструктури), арбітражні можливості виникають значно частіше та утримуються довше, ніж у класичних ринках [3, 4]. Розглянемо основні типи арбітражних стратегій, релевантні для криптовалютних деривативів.

1.2.1 Просторовий (міжбіржовий) арбітраж

Просторовий арбітраж – найдавніша і концептуально найпростіша стратегія. Полягає у купівлі активу на біржі А, де його ціна нижча, та одночасному продажу на біржі Б, де ціна вища. Прибуток складає різницю цін за вирахуванням комісій бірж, витрат на переказ активу та потенційного прослизання ціни (slippage).

Структура комісій бірж суттєво впливає на прибутковість стратегії. Для активних учасників (VIP-рівні з обсягом торгів понад 10 млн доларів США на місяць) комісії становлять: на Binance – 0,02 % taker / 0 % maker; на Bybit – 0,025 % taker / −0,01 % maker (тобто маркет-мейкер отримує rebate за надання ліквідності); на OKX – 0,02 % taker / 0,005 % maker. Для роздрібних учасників базові комісії в декілька разів вищі – близько 0,1 % taker та 0,08 % maker. У такій структурі ціновий розрив між біржами має перевищувати щонайменше 0,2–0,3 % для отримання чистого прибутку.

Практична реалізація просторового арбітражу на криптовалютному ринку обмежена кількома факторами. По-перше, перенесення активу між біржами займає від кількох хвилин (швидкі мережі – Solana, Tron, Polygon) до декількох годин (мережі з повільним підтвердженням транзакцій – Bitcoin, при низькій комісії може займати понад годину). За цей час ціновий розрив зазвичай закривається діями інших учасників. По-друге, біржі періодично призупиняють депозити та виводи через технічне обслуговування, форс-мажорні обставини або проблеми з гарячими гаманцями. У 2024 році тільки на Binance зареєстровано понад 40 випадків призупинення депозиту/виводу для

конкретних монет тривалістю від 2 до 48 годин. По-третє, на біржах діють ліміти добового виводу для неверифікованих рахунків: типовий ліміт без розширеної верифікації – екв. 2 BTC на добу, що обмежує масштаб операцій.

У практиці інституційних учасників (Wintermute, Jump Trading, Hudson River Trading, GSR) просторовий арбітраж реалізується у формі низьколатентного (low-latency) арбітражу, що вимагає колокації серверів поряд із дата-центрами бірж, доступу до приватних API, прямого підключення до book feeds [10]. Затримка між отриманням оновлення та виставленням ордера у таких системах вимірюється одиницями мікросекунд. Для роздрібних учасників та невеликих фірм такий підхід економічно недоцільний; натомість для них реалізованими є форми арбітражу, що не вимагають передачі коштів між біржами в момент сигналу, – насамперед стратегії, що ґрунтуються на одночасних позиціях у спотовому та деривативному ринку (підрозділ 1.2.2).

1.2.2 Спот-ф’ючерсний арбітраж (basis trade)

Спот-ф’ючерсний арбітраж (cash-and-carry, basis trade) – ключова стратегія, на якій сфокусовано це дослідження [9, 11]. Полягає у одночасному відкритті протилежно спрямованих позицій на спотовому та ф’ючерсному ринках одного й того ж активу з метою заробляння на базисі – різниці між їх цінами.

Формально базис визначають як $B = (F - S) / S \times 100\ \%$, де F – ціна ф’ючерсного контракту, S – ціна спотового активу. У стані рівноваги базис безстрокового ф’ючерсу близький до нуля, оскільки механізм фінансування забезпечує його збіжність до спотової ціни. Однак у періоди ринкового дисбалансу (масові ліквідації, новинні події, маніпуляції великих гравців) базис може набувати значень від ±0,5 % до ±10 % і більше.

Розрізняють два типи аномалій базису. Позитивний базис (F > S, контанго) виникає в періоди ажіотажного попиту на ф’ючерсні позиції. Стратегія: купити спот, відкрити коротку позицію в ф’ючерсі; при збіжності цін

фіксується прибуток. Негативний базис (F < S, бекворадейшн) виникає при масових ліквідаціях довгих позицій або при інтенсивному відкритті коротких позицій. Стратегія: відкрити довгу позицію в ф'ючерсі, продати спотовий актив (або взяти його у крипто-позику для шорту); при збіжності цін також фіксується прибуток.

Очікуваний прибуток арбітражної операції із врахуванням комісій можна виразити формулою: $P = |B| - f_spot - f_fut - slippage$, де f_spot і f_fut – комісії за відкриття позицій на спотовому та ф'ючерсному ринках відповідно, *slippage* – прослизання ціни при виконанні ордера. Для активного учасника з комісіями близько 0,02 % на кожній стороні угоди мінімальний базис, що покриває транзакційні витрати, становить приблизно 0,15 %. На практиці автор системи встановлює нижній поріг чутливості $|B| \geq 3\ \%$, що дозволяє формувати позиції з очікуваним прибутком 2–3 % за операцію.

Розглянемо приклад розрахунку для негативного базису. Припустимо, що для деякої монети спостерігається базис B = −5 %; спотова ціна S = 1,000 USDT; ф'ючерсна ціна F = 950 USDT; розмір позиції – 10,000 USDT капіталу. Стратегія: відкрити довгу позицію в ф'ючерсі обсягом 10,000 USDT (купити 10,53 одиниць активу за ф'ючерсною ціною), одночасно взяти у крипто-позику 10 одиниць спотового активу і продати їх за 10,000 USDT. У момент збіжності цін (наприклад, до базису 0 %) ф'ючерсну позицію закриваємо з прибутком 500 USDT, спотову коротку позицію також закриваємо з приблизно еквівалентним результатом. Сумарний валовий прибуток – близько 500 USDT (5 %) на 10,000 USDT капіталу, без урахування ставки за крипто-позику (зазвичай 0,02–0,05 % на добу) та комісій.

Особливістю спот-ф'ючерсного арбітражу є те, що його прибутковість залежить не лише від виявлення базисної аномалії, але й від можливості її практичної реалізації. Зокрема, у випадку негативного базису для здійснення продажу спотового активу необхідна доступність крипто-позики (loan) для шорт-операції на цільовій біржі. Якщо позика на потрібну монету недоступна або вже вичерпана, теоретично прибуткова арбітражна можливість не може

бути практично реалізована. Аналогічно, призупинення депозитів або виводів конкретної монети унеможливлює перенесення позиції між біржами при необхідності її закриття. Окремий ризик становить можлива ліквідація позицій у разі різкого розширення базису у протилежний бік – для його уникнення необхідне ретельне управління плечем та забезпечувальною маржею.

Спостереження автора в процесі експлуатації власної системи моніторингу показують, що за фільтрації аномалій базису лише за критерієм $|B| \geq 3$ % частка практично нереалізованих сигналів становить понад 60 % (через відсутність позик, закриті депозити, низьку ліквідність книги заявок). Це обумовлює необхідність інтеграції до алгоритму виявлення сигналу додаткових перевірок умов реалізованості, що буде детально розглянуто у Розділі 2 і складає першу позицію наукової новизни дослідження.

1.2.3 Фандінговий арбітраж

Фандінговий арбітраж (funding rate arbitrage) – стратегія, специфічна саме для безстрокових ф'ючерсів [1]. Полягає у відкритті дельта-нейтральної позиції (одночасно довгої на спотовому ринку та короткої на ф'ючерсному), з метою отримання періодичних виплат фінансування без прийняття ринкового ризику. Якщо ставка фінансування позитивна – коротка позиція в ф'ючерсі регулярно отримує виплати від довгих позицій; у разі негативної ставки сторони змінюються.

Біржі застосовують різні алгоритми та інтервали розрахунку фінансування. На основних майданчиках виплати здійснюються тричі на добу: на Binance, OKX, Bybit, Bitget, Gate.io – кожні 8 годин (о 00:00, 08:00, 16:00 UTC). На MEXC – кожні 4 години (шість виплат на добу). На KuCoin – кожні 8 годин, але з різницею у часі (04:00, 12:00, 20:00 UTC). На BitMEX історично – кожну годину. Формула розрахунку ставки на більшості бірж включає компонент індексу премії (різниця між поточною ф'ючерсною та спотовою

ціною) та фіксовану складову процентної ставки (interest rate), що відображає вартість позики базової валюти.

У типовому ринковому стані середня ставка фінансування коливається в діапазоні ±0,01 % за період, що відповідає річній дохідності близько 10 % при дельта-нейтральній позиції. У періоди ринкових екстремумів ставки можуть досягати 0,1 % і більше за період, що відповідає еквіваленту дохідності понад 100 % річних. Зокрема, у листопаді 2024 року на тлі бичачого тренду Bitcoin ставки фінансування на низці бірж досягали 0,3 % за 8 годин (близько 328 % річних) для топ-10 активів.

Особливий випадок фандінгового арбітражу – кросс-біржовий фандінговий арбітраж: одна й та сама монета може мати протилежно спрямовані ставки фінансування на різних біржах через дисбаланси локального попиту. Наприклад, на основі реальних даних uainvest зафіксовано випадки, коли для монети RIVER ставка фінансування на Binance становила +0,08 % за 8 годин (тобто довгі позиції сплачують), а на Gate.io одночасно −0,12 % за 8 годин (короткі позиції сплачують). У такому разі стратегія полягає в одночасному відкритті довгої позиції на Gate.io та короткої позиції на Binance, з ефективним надходженням 0,20 % за 8 годин, або еквівалентом понад 200 % річних, без ринкового ризику. Реалізація такої стратегії є основним призначенням розробленого автором функціонального модуля find_ops, що буде описано в Розділі 3.

Основні ризики фандінгового арбітражу – раптова зміна знака ставки протягом розрахункового періоду (що нівелює очікуваний прибуток), різниця цін відкриття позицій на двох біржах (entry spread), а також ризик ліквідації при значних цінових рухах базового активу. Для мінімізації цих ризиків необхідне ретельне управління розміром позиції та забезпеченням, а також безперервний моніторинг поточних ставок з оперативним закриттям позиції у разі несприятливої динаміки.

1.2.4 Статистичний та триангулярний арбітраж

Статистичний арбітраж (statistical arbitrage) – стратегія, що ґрунтується на виявленні стійких статистичних залежностей між цінами активів і відкритті позицій у моменти відхилення поточних цін від цих залежностей. Класичний приклад – парний трейдинг (pairs trading) між активами з високою історичною кореляцією: при тимчасовому розходженні цін відкривають довгу позицію в активі, що відстав, і коротку в активі, що випередив, з очікуванням відновлення співвідношення. Для криптовалют пари типу BTC/ETH, ETH/SOL, BNB/BTC демонструють кореляцію 0,7–0,9 у довгостроковій перспективі, однак ця кореляція схильна до різких розривів під час новинних шоків, специфічних для окремих проєктів.

Триангулярний арбітраж використовує співвідношення між трьома торговими парами на одній біржі. Наприклад, для пар BTC/USDT, ETH/USDT та ETH/BTC у стані рівноваги має виконуватися співвідношення: *(ціна BTC/USDT) × (ціна ETH/BTC) ≈ (ціна ETH/USDT)*. Відхилення від цієї рівності створює можливість для прибутку шляхом послідовного обміну валют по колу. На високоліквідних парах основних бірж такі відхилення зазвичай не перевищують 0,01–0,05 % і закриваються алгоритмами market makers за мілісекунди, що робить стратегію недоступною для звичайних учасників.

На криптовалютному ринку обидві ці стратегії мають обмежену застосовність. Статистичний арбітраж ускладнений високою волатильністю та схильністю криптоактивів до асиметричних реакцій на новинні події (хакерські атаки, регуляторні рішення щодо конкретного протоколу), що порушує стійкість історичних кореляцій. Триангулярний арбітраж технічно реалізуємий, але вимагає інфраструктури мінімальної затримки. У контексті цього дослідження зазначені типи арбітражу не розглядаються як основні, але враховуються при формуванні архітектури системи як потенційні напрями подальшого розширення (підрозділ 3.11).

1.3 Огляд існуючих програмних рішень

На ринку доступний широкий спектр програмних рішень для алгоритмічної торгівлі та арбітражу на криптовалютних ринках. Класифікувати їх можна за моделлю розгортання (локальні / хмарні), моделлю ліцензування (відкритий / закритий код, безкоштовні / за підпискою), а також за основними цільовими стратегіями (market making, виконання сигналів, арбітраж, копі-трейдинг). Розглянемо ключових представників різних категорій з точки зору їх архітектури, функціональних можливостей та обмежень у контексті задач спот-ф’ючерсного та фандінгового арбітражу.

1.3.1 Hummingbot

Hummingbot – проєкт з відкритим вихідним кодом (Apache 2.0), що розробляється спільнотою Hummingbot Foundation з 2019 року. Реалізований переважно мовою Python з елементами C++ для критичних до швидкості модулів. Архітектура заснована на концепції стратегій (strategies) як окремих Python-класів, що користувачі можуть писати самостійно. Підтримує понад 30 централізованих та десятки децентралізованих бірж через уніфікований шар конекторів (connectors). Основне призначення – алгоритмічний market making з виставленням двосторонніх котирувань. Має модуль міжбіржового арбітражу (cross-exchange market making), однак цей модуль працює саме як market making з хеджуванням, а не як виявлення дискретних арбітражних можливостей. Не підтримує специфічних для деривативів механізмів – зокрема, моніторингу ставок фінансування на множині бірж, відслідковування доступності крипто-позик, перевірки статусу депозитно-вивідних шлюзів. Розгортання вимагає технічної експертизи: локальна установка через Docker, конфігурація API-ключів, написання користувацьких стратегій. Цільова аудиторія – технічно підготовлені користувачі та інституційні учасники.

1.3.2 3Commas

3Commas – комерційна хмарна платформа, що працює за моделлю SaaS. Базується в Естонії, запущена у 2017 році. Пропонує готові торгові боти за моделлю підписки (тарифи від 29 до 99 доларів США на місяць). Користувач підключає API-ключі своїх бірж до хмарної платформи через web-інтерфейс і налаштовує параметри ботів без необхідності написання коду. Підтримує понад 18 бірж. Основні продукти – GRID-бот (виставляє ордера на сітку цін у заданому діапазоні), DCA-бот (поетапна купівля з усередненням позицій), SmartTrade (інтерактивне закриття позиції з кількома цільовими цінами). Не реалізує функціонал виявлення базисних або фандінгових аномалій; арбітраж між біржами обмежений простими сценаріями. Платформа орієнтована на роздрібного користувача без технічної експертизи; зворотним боком цього є обмежена гнучкість і відсутність можливості реалізації нестандартних стратегій.

1.3.3 Cryptohopper

Cryptohopper – ще одна комерційна хмарна платформа, базується в Нідерландах, запущена у 2017 році. Архітектурно схожа на 3Commas (SaaS-модель, тарифи від 19 до 99 доларів США на місяць), але з акцентом на копі-трейдинг та виконання сигналів від зовнішніх провайдерів. Має маркетплейс сигналів, де треті провайдери продають доступ до своїх торгових сигналів. Має конструктор стратегій з технічними індикаторами (ковзні середні, RSI, MACD, Bollinger Bands та ін.), однак не пропонує інструментів для спот-ф’ючерсного або фандінгового арбітражу. Підтримує 14 бірж. Включає бектестинг – можливість перевірки стратегії на історичних даних.

1.3.4 Bitsgap

Bitsgap – платформа, орієнтована саме на арбітражні стратегії. Базується в Естонії, запущена у 2018 році, тарифи від 23 до 149 доларів США на місяць. Має вбудований модуль міжбіржового арбітражу та індикатори ставок фінансування для понад 15 бірж. Виявляє цінові розриви між біржами та надає рекомендації щодо їх використання. Проте, як показує практика, виявлення базисних аномалій не доповнюється перевіркою умов реалізованості (доступність позик, статус депозитно-вивідних шлюзів), що призводить до значної частки нереалізованих сигналів. Інтегрує також функціонал автоматизованих GRID-ботів та портфельного моніторингу. Підтримує спот-ринки, ф'ючерсні ринки підтримуються частково.

1.3.5 Інституційні рішення

Інституційні рішення (Wintermute, Jump Crypto, Hudson River Trading, GSR, Cumberland) – закриті системи провідних торгових фірм, що використовують низьколатентний доступ, колокацію та патентовані алгоритми. Інформація про їх архітектуру публічно недоступна, але з періодичних інтерв'ю та технічних публікацій членів цих команд відомо, що їх системи реалізують комплексні стратегії, що поєднують просторовий, статистичний, фандінговий арбітраж та market making. Об'єми торгів цих учасників оцінюються в десятки мільярдів доларів США на місяць. Однак це рішення фактично недоступне для роздрібних учасників ринку та невеликих фірм через бар'єри технологічної інфраструктури, регуляторних вимог та капіталу.

Для систематизованого порівняння розглянутих рішень з точки зору ключових функціональних характеристик у таблиці 1.1 наведено зведений аналіз.

Таблиця 1.1 – Порівняння існуючих програмних рішень для крипто-арбітражу

Характеристика	**Hummingbot**	**3Commas**	**Cryptohopper**	**Bitsgap**	**Власна система**
Кількість підтриманих бірж	30+	18	14	15+	13
Просторовий арбітраж	Так	Обмеж.	Ні	Так	Так
Спот-ф’ючерсний арбітраж (basis)	Ні	Ні	Ні	Частково	Так
Фандінговий арбітраж	Ні	Ні	Ні	Індикатори	Так
Моніторинг доступності позик	Ні	Ні	Ні	Ні	Так
Перевірка статусу депозитів	Ні	Ні	Ні	Ні	Так
Швидкий цикл (sub-second)	Ні	Ні	Ні	Ні	Так
Аудіо-сповіщення (Pushover)	Ні	Ні	Ні	Ні	Так
On-chain дані	Ні	Ні	Ні	Ні	Так
Відкритий вихідний код	Так	Ні	Ні	Ні	Так
Вартість, USD/міс	0+VPS	29-99	19-99	23-149	~30 VPS

Як видно з таблиці, жодне з наявних рішень не реалізує одночасно: спот-ф’ючерсного арбітражу з перевіркою умов реалізованості, моніторингу доступності позик у режимі реального часу, інтеграції on-chain даних та ескалаційних сповіщень з аудіо-сигналом критичної пріоритетності. Цей розрив у функціональності і визначає нішу для розробки спеціалізованої системи, представленої у цій роботі.

1.4 Джерела ринкових даних

Якість виявлення арбітражних можливостей безпосередньо залежить від якості, повноти та своєчасності вхідних даних. У контексті задачі моніторингу криптовалютного арбітражу можна виділити три основні категорії джерел даних: безпосередні API торгових майданчиків, агрегатори ринкових даних та on-chain (блокчейн) дані. Розглянемо кожну категорію детальніше з точки зору технічних характеристик, обмежень та доцільності використання у проєктованій системі.

1.4.1 Біржові API: REST та WebSocket

Усі провідні централізовані біржі надають публічні програмні інтерфейси (API), що можуть бути доступні через два основні протоколи. REST API працює за моделлю request-response: клієнт надсилає HTTP-запит, отримує відповідь у форматі JSON, повторює цикл за необхідності. REST зручний для отримання снапшотів даних і виконання операцій (відкриття/закриття ордерів), але має обмеження за частотою запитів (rate limits). На Binance, наприклад, дозволено до 1200 «зважених» запитів на хвилину з одного IP-адреса, причому різні endpoints мають різну вагу (від 1 до 50 одиниць) [12]. На OKX – 600 запитів на хвилину [13], на Gate.io – 900 [14], на MEXC – 1200, на Bybit – 600 [15]. Перевищення лімітів призводить до тимчасового блокування доступу на термін від 1 до 5 хвилин, що критично для real-time моніторингу.

WebSocket забезпечує повнодуплексний потік оновлень у реальному часі: після встановлення з’єднання сервер біржі надсилає клієнту оновлення (тікери, зміни книги заявок, виконання угод) без додаткових запитів. Затримка доставки оновлень становить від 10 до 100 мілісекунд залежно від географічної відстані до серверів біржі та якості мережного з’єднання. WebSocket принципово ефективніший за REST для задач, де потрібно безперервно відслідковувати багато активів, оскільки не споживає rate limits. Проте він складніший у

реалізації: необхідне управління станом з’єднань, обробка переходів через переподключення, синхронізація локальної копії книги заявок з потоком оновлень.

1.4.2 Агрегатори даних

Для задач, що потребують одночасного порівняння даних з багатьох бірж, доцільне використання спеціалізованих агрегаторів. У цій роботі основним обраним агрегатором є сервіс uainvest.com.ua [16]. Він надає агреговані дані щодо ставок фінансування і цін безстрокових ф’ючерсів з восьми централізованих бірж (Binance, Bybit, OKX, Bitget, Gate.io, KuCoin, MEXC, Phemex) у єдиному форматі та одному запиті. Це суттєво знижує сумарне навантаження на власні rate limits бірж та спрощує логіку співставлення даних. Структура API uainvest: один HTTP-запит повертає до 30 монет за сторінку, із поточними цінами безстрокових ф’ючерсів та ставками фінансування за останні 24 години на всіх восьми біржах. Затримка оновлення даних на uainvest – до 60 секунд, що достатньо для задач фандінгового арбітражу, але недостатньо для виявлення короткострокових базисних аномалій.

Альтернативні агрегатори – CoinGlass [17] та Coinalyze. CoinGlass має ширший набір функціональних можливостей (історичні дані щодо ліквідацій, open interest за монетами, статистика по long/short ratio, heatmaps), однак має жорсткіші обмеження на безкоштовних тарифах (10–30 запитів на хвилину) та закритий API для багатьох endpoints. Coinalyze орієнтований на історичні дані для бектестингу. У цій роботі ці сервіси не використовуються основними, але можуть бути інтегровані у наступних версіях системи для розширення функціоналу.

1.4.3 On-chain дані

Блокчейн-провідники (Etherscan, BSCscan, Solscan, Arbiscan, Polygonscan) та публічні RPC-вузли надають програмний доступ до даних блокчейну: руху коштів між адресами, активності великих гравців (whales), обсягів випуску та спалювання токенів, активності децентралізованих бірж. Для кожної з основних мереж (Ethereum, BSC, Solana, Arbitrum, Polygon) існують REST API провідники з безкоштовним базовим тарифом (5 запитів на секунду після реєстрації API-ключа) та платними тарифами (до 100 запитів на секунду).

У контексті криптовалютного арбітражу on-chain дані можуть слугувати випереджувальним індикатором. Наприклад, переказ великих обсягів стейбл-коінів (USDT, USDC) на гарячі гаманці централізованих бірж зазвичай передує підвищеній торговій активності та може корелювати з виникненням базисних аномалій. Активізація великих гаманців (whales) – перерозподіл коштів, новий лістинг – також часто супроводжується суттєвими ціновими рухами. Цей напрямок є відносно новим у академічній літературі та становить частину наукової новизни цієї роботи (підрозділ 2.6).

Зведене порівняння обраних джерел даних з точки зору ключових характеристик представлене у таблиці 1.2.

Таблиця 1.2 – Порівняння джерел ринкових даних, що використовуються в системі

Джерело	Тип	Покриття	Затримка	Rate limit	Вартість
Binance API	REST/WS	1 біржа	10-50 мс	1200/хв	Безкоштовно
OKX API	REST/WS	1 біржа	20-100 мс	600/хв	Безкоштовно
Bybit API	REST/WS	1 біржа	20-100 мс	600/хв	Безкоштовно
Gate.io API	REST/WS	1 біржа	30-150 мс	900/хв	Безкоштовно
MEXC API	REST/WS	1 біржа	30-200 мс	1200/хв	Безкоштовно
uainvest	REST	8 бірж	до 60 с	Помірний	Безкоштовно
CoinGlass	REST	20+ бірж	1-10 с	Жорсткий	Безкоштовно / Pro
Etherscan	REST	Ethereum	1-5 с	5/с	Безкоштовно (з API-ключем)
BSCscan	REST	BSC	1-5 с	5/с	Безкоштовно (з API-ключем)

Обраний у цій роботі стек джерел даних поєднує безпосередній доступ до п’яти найбільших централізованих бірж через REST API (для актуальності цін і деталей ринку), агреговані фандінг-дані з uainvest (для масштабованого моніторингу 8 бірж одночасно) та on-chain дані через провідники мереж Ethereum, BSC та Solana (для випереджувальних сигналів). Така комбінація забезпечує баланс між покриттям, затримкою та обмеженнями rate limits. Деталі архітектурної реалізації описано у Розділі 2.

1.5 Постановка задачі та формалізація вимог

На основі аналізу предметної області (підрозділи 1.1–1.4) сформулюємо задачу дослідження як побудову розподіленої програмної екосистеми, що автоматично виявляє арбітражні можливості різних типів на криптовалютних деривативних ринках у режимі реального часу та сповіщає користувача про знайдені можливості через канали з диференційованим рівнем критичності.

Задача має ознаки складної інформаційно-аналітичної системи реального часу: необхідно одночасно обробляти потоки даних з 8+ зовнішніх API, виконувати обчислення (порівняння спредів, перевірки умов реалізованості, статистичну обробку), формувати та диференційовано доставляти сповіщення, забезпечувати відмовостійкість і безперервність роботи. Це обумовлює необхідність модульної архітектури з чітко визначеними інтерфейсами між компонентами.

Функціональні вимоги (FR) до системи:

FR1 Виявлення спот-ф’ючерсних базисних аномалій на щонайменше п’яти централізованих біржах із подальшою перевіркою умов реалізованості сигналу (доступність крипто-позики, статус депозитно-вивідних шлюзів).

FR2 Виявлення кросс-біржових фандінгових арбітражних можливостей між вісьмома централізованими біржами шляхом агрегованого моніторингу ставок фінансування.

FR3 Сегментоване виявлення арбітражних можливостей для специфічних класів активів (товарні токени золота й срібла; малокапіталізовані активи з підвищеною волатильністю).

FR4 Швидкий цикл цілеспрямованого моніторингу (інтервал ≤ 5 с) для пріоритетних активів, що знаходяться в стані очікування появи додаткових умов реалізації арбітражу.

FR5 Інтеграція on-chain даних (рухи коштів, активність великих гаманців) як випереджувального індикатора подальших цінових подій на централізованих біржах.

FR6.Підсистема сповіщень з диференційованим рівнем критичності: стандартні повідомлення через Telegram, критичні події – через Pushover з аудіо-сигналом emergency-пріоритету.

FR7 Можливість конфігурації порогових значень кожного модуля без перезапуску системи (через інтерфейс Telegram-ботів).

FR8 Підтримка чорних списків монет з можливістю динамічного додавання та видалення.

FR9 Логування всіх виявлених сигналів та результатів подальших дій з метою подальшого статистичного аналізу.

FR10 Автоматичне відновлення роботи модулів після збоїв (мережева недоступність API, перевищення rate limits, рестарт сервера).

Нефункціональні вимоги (NFR) до системи:

NFR1 Цілодобова безперебійна робота на хмарному сервері (uptime ≥ 99%).

NFR2 Затримка між виявленням сигналу та доставкою сповіщення користувачу – не більше 5 секунд для стандартних подій і не більше 2 секунд для критичних.

NFR3 Помірне споживання обчислювальних ресурсів – система має ефективно працювати на віртуальному сервері середньої потужності (2 vCPU, 4 ГБ RAM, як AWS EC2 t3.medium).

NFR4 Модульна архітектура з можливістю незалежного запуску, зупинки та оновлення кожного функціонального компоненту.

NFR5 Прозорість та зрозумілість виявлених сигналів для користувача-аналітика: кожне сповіщення містить детальні параметри сигналу (ціни, спред, доступні обсяги позик, статус депозитів, посилання на джерела даних).

Обмеження проєкту (scope limitations):

OL1 Система здійснює виявлення та сповіщення про арбітражні можливості, але не виконує автоматичну торгівлю від імені користувача (за винятком експериментального модуля автоматичного взяття крипто-позик).

OL2 Не розглядаються децентралізовані біржі (DEX) та їх протоколи безстрокових ф'ючерсів – концентрація лише на централізованих майданчиках.

OL3 Не реалізується модуль формального бектестингу – оцінка ефективності проводиться через експериментальну апробацію на реальних даних протягом обмеженого періоду.

OL4 Не реалізується інтеграція з системами податкового обліку та звітності.

Узагальнену діаграму використання системи наведено на рисунку 1.1.

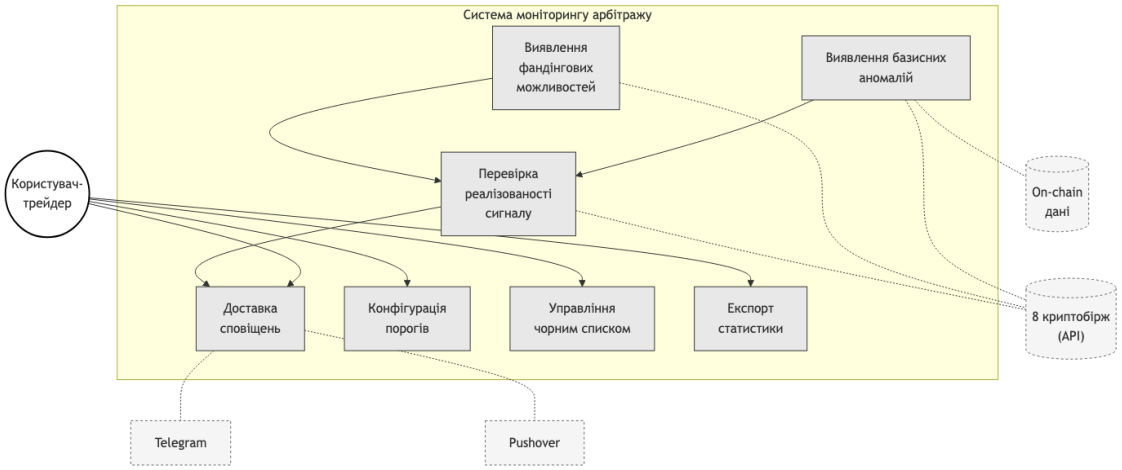


Рисунок 1.1 – Узагальнена діаграма використання (use case) системи

Сформульовані функціональні та нефункціональні вимоги, а також зазначені обмеження проєкту, становлять основу для проєктування архітектури системи (розділ 2) та її програмної реалізації (розділ 3). Завдання, поставлене у цій роботі, належить до класу задач побудови розподілених інформаційно-аналітичних систем реального часу з інтеграцією гетерогенних джерел даних та диференційованим керуванням пріоритетами обробки подій.

2 МАТЕМАТИЧНІ МОДЕЛІ ТА АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ

2.1 Математична модель базису між спотовим та ф’ючерсним ринком

Базисний арбітраж ґрунтується на формальному математичному відношенні між цінами спотового і ф’ючерсного активу. Розглянемо побудову моделі базису для двох типів контрактів – класичного строкового ф’ючерсу і безстрокового ф’ючерсу (perpetual swap), оскільки механізми збіжності цін у цих двох випадках принципово різні.

Для класичного ф’ючерсу із датою закінчення згідно з класичною моделлю cost-of-carry теоретична справедлива ціна виражається формулою:

$$F = S \times e^{((r - q) \times T)}, \tag{2.1}$$

де F – теоретична ф’ючерсна ціна;

S – спотова ціна;

r – безризикова процентна ставка;

q – дохідність зберігання активу (для криптовалют ≈ 0);

T – період до закінчення контракту у роках.

На дату закінчення $T = 0$, і ціни теоретично сходяться: $F = S$.

Для безстрокового ф’ючерсу формула (2.1) непридатна, оскільки відсутня дата закінчення. Натомість механізм збіжності забезпечується періодичним обміном виплатами між учасниками ринку – функцією фінансування (funding rate). Ставка фінансування на більшості бірж розраховується як обмежене значення премії індексу над поточною ставкою:

$$f = clamp(PI + IR, -f_max, +f_max), \tag{2.2}$$

де f – ставка фінансування за період (зазвичай 8 годин);

PI – premium index, нормалізоване середнє різниці між ціною ф’ючерсного контракту і ціною спотового індексу за період;

IR – interest rate, фіксована процентна ставка (на Binance, OKX, Bybit IR = 0,01 % за 8 годин);

f_max – максимальна абсолютна ставка (зазвичай *0,75 %* за період).

Якщо ставка позитивна, довгі позиції сплачують коротким; якщо негативна – навпаки.

Базис безстрокового ф’ючерсу – поточна різниця між ф’ючерсною та спотовою цінами, нормалізована на спотову ціну:

$$B = (F - S) / S \times 100\ \%. \tag{2.3}$$

У стані рівноваги абсолютне значення базису не може довго перевищувати очікувані сумарні виплати фінансування за період утримання позиції, оскільки в іншому випадку арбітражні стратегії учасників ринку відновлять баланс. Формально, для горизонту утримання у N періодів фінансування, очікувана арбітражна різниця обмежується формулою:

$$E[\Delta B] = N \times E[f]. \tag{2.4}$$

Базисна аномалія виникає, коли поточне значення базису суттєво перевищує теоретично виправдане за формулою (2.4). Емпіричне дослідження автора з аналізу власних логів за 2024–2025 рр. показало, що сигнал $|B| \geq 3$ % практично завжди перевищує очікувані виплати фінансування за наступні 8–16 годин і може вважатися арбітражною аномалією. Очікуваний валовий прибуток від реалізації базисного арбітражу з урахуванням транзакційних витрат описується формулою:

$$P = |B| - f_spot - f_fut - s_slippage - r_loan \times t_hold, \tag{2.5}$$

де *f_spot* і *f_fut* – комісії за відкриття позицій на спотовому та ф'ючерсному ринках;

s_slippage – оцінене прослизання ціни;

r_loan – ставка крипто-позики за період (для негативного базису, коли потрібна позика спотового активу для шорту);

t_hold – очікуваний термін утримання позиції.

Розглянемо чисельний приклад. Для активного учасника з комісією 0,02 % на кожній стороні, прослизанням 0,1 %, ставкою позики 0,03 % на добу і терміном утримання 1 добу мінімальний базис для беззбиткової стратегії становить:

$$|B|_{min} = 0{,}02 + 0{,}02 + 0{,}10 + 0{,}03 = 0{,}17\ \%. \qquad (2.6)$$

Таким чином, формальний поріг чутливості системи моніторингу повинен встановлюватися не нижче 0,2 % для покриття транзакційних витрат. На практиці автор системи встановлює помітно вищий поріг (3 %) – з метою отримання сигналів із суттєвим профіт-маржином, що компенсує ризик невдалої реалізації та нелінійні ефекти (переоцінка прослизання при збільшенні позиції, зростання ставки позики при підвищеному попиті).

Окремої уваги заслуговують граничні випадки моделі. По-перше, для активів з недостатньою глибиною книги заявок прослизання s_slippage набуває нелінійної залежності від обсягу позиції: при спробі виконати ордер обсягом понад 5–10 % доступної ліквідності цінове прослизання може зростати кратно. У формулі (2.5) це означає, що для практичних розрахунків необхідно оцінювати s_slippage не як константу, а як функцію обсягу позиції за поточним станом книги заявок. По-друге, для волатильних альткоїнів ставка крипто-позики r_loan не є фіксованою, а коригується біржею у режимі реального часу в залежності від коефіцієнту використання пулу. При повному вичерпанні доступної суми позики ставка може стрибати у 5–10 разів, що повністю нівелює очікуваний прибуток.

По-третє, термін утримання позиції t_hold у формулі (2.5) для безстрокових ф’ючерсів не фіксований і теоретично не обмежений. У практиці зазвичай розглядають часові горизонти від декількох годин до декількох днів, після чого позиція або закривається з прибутком (при збіжності цін), або модифікується через зміну ринкових умов. Емпіричне дослідження автора показує, що медіанний термін збіжності аномалії $|B| \geq 3$ % становить близько 4 годин, а 90 % аномалій сходяться протягом 24 годин. Це означає, що для типового сценарію вартість крипто-позики додає до сумарних витрат приблизно 0,01 % за весь термін утримання (за середньої ставки 0,03 % на добу) – суттєво менше, ніж комісії та прослизання.

У підсумку математична модель базису, представлена формулами (2.1)–(2.6), забезпечує формальну основу для побудови алгоритму виявлення базисних аномалій у режимі реального часу. Параметри моделі калібруються емпірично за історичними даними або задаються користувачем через інтерфейс конфігурації Telegram-ботів. Подальші розширення моделі – врахування глибини книги заявок, динамічної корекції порогів за поточною волатильністю, інтеграції з прогнозованими оцінками тривалості аномалії – складають можливі напрями подальшого розвитку, обговорені у підрозділі 3.11.

2.2 Математична модель фандінг-арбітражу

Фандінг-арбітраж заснований на отриманні виплат фінансування за рахунок утримання дельта-нейтральної позиції. На відміну від базисного арбітражу, де прибуток генерується одноразово при збіжності цін, у фандінг-арбітражі дохід надходить регулярно при кожному розрахунку фінансування.

Сформулюємо математичну модель для двох варіантів стратегії: моно-біржового (позиція відкривається на одній біржі) та кросс-біржового (позиції на двох різних біржах одночасно).

Моно-біржовий випадок. Учасник відкриває довгу позицію в спотовому ринку обсягом V і коротку позицію в безстроковому ф’ючерсі того ж активу

того ж обсягу на тій самій біржі. Ринкова експозиція дельта-нейтральна: будь-яка зміна ціни активу компенсується протилежним рухом на іншій позиції. Очікуваний дохід за N періодів розраховується за формулою (2.7):

$$R_mono = V \times (\Sigma f_i - f_entry - f_exit), \quad (2.7)$$

де f_i – ставка фінансування за і-й період;

f_entry та f_exit – сумарні транзакційні витрати відкриття і закриття обох позицій.

Для типових значень (комісія 0,02 % на кожній стороні, відкриття і закриття дають $f_entry + f_exit = 0{,}08\ \%$) точка беззбитковості досягається при сукупному фінансуванні понад 0,08 %.

Кросс-біржовий випадок. Учасник відкриває довгу позицію на біржі A (де фінансування негативне, тобто виплачується довгим) і коротку позицію на біржі B (де фінансування позитивне, тобто виплачується коротким) того ж активу одного обсягу V. Сумарний дохід:

$$R_cross = V \times [\Sigma (f_B,i - f_A,i) - f_entry - f_exit - e_spread], \quad (2.8)$$

де додатково e_spread – різниця цін відкриття позицій на двох біржах (entry spread).

Через несинхронну ліквідність ціни в момент відкриття можуть розходитися на 0,01–0,1 %, що становить помітну витрату стратегії.

Важливою метрикою при ранжуванні арбітражних можливостей є показник кількості періодів фінансування, що необхідні для покриття транзакційних витрат – days-to-breakeven (T_BE). Він розраховується як:

$$T_BE = (f_entry + f_exit + e_spread) / |f_B - f_A|. \quad (2.9)$$

Для прикладу: транзакційні витрати 0,1 %, дельта фандінгу між біржами 0,05 % за 8 годин – точка беззбитковості досягається за 2 розрахункові періоди, або за 16 годин. Якщо очікуваний термін аномалії перевищує цей показник, стратегія розглядається як прибуткова.

У реалізованому функціональному модулі find_ops (Розділ 3) ранжування арбітражних можливостей здійснюється за комбінованим критерієм, що поєднує: статус біржової пари у класифікації top-tier (як індикатор ліквідності), значення T_BE (як інтегральний показник прибутковості), очікуваний 24-годинний прибуток і комплексну оцінку ризику біржі.

Основні ризики фандінг-арбітражу описуються трьома параметрами. Ризик зміни знака ставки – ймовірність того, що ставка зміниться на протилежну до завершення розрахункового періоду; емпірично частота таких випадків становить близько 8 % за 8-годинний період для топ-50 активів. Ризик ліквідації – ймовірність виходу ціни базового активу за межі підтримки маржі при різких рухах; мінімізується консервативним вибором плеча (не більше 3×). Базисний ризик – раптове розширення базису у несприятливий бік при закритті позиції, що може призвести до додаткових втрат у розмірі 0,1–1 % від обсягу позиції.

Розглянемо чисельний приклад кросс-біржового фандінг-арбітражу за реальними даними автора з логів системи. У січні 2025 року для монети SAGA на біржі Binance ставка фінансування становила +0,082 % за 8-годинний період (тобто довгі позиції сплачують короткими), а одночасно на біржі Bybit для тієї ж монети ставка становила −0,054 % (тобто короткі позиції сплачують довгим). Учасник, що відкриває коротку позицію на Binance і довгу позицію на Bybit обсягом 5000 USDT кожна, отримує дельту фінансування 0,082 + 0,054 = 0,136 % за період, або 4,1 USDT (0,068 % від сумарного капіталу 10000 USDT, оскільки виплати йдуть від обсягу позиції, не сумарного капіталу). За добу – 12,3 USDT, або 0,123 % від сумарного капіталу. Транзакційні витрати на відкриття обох позицій складають близько 4 USDT (0,02 % × 2 позиції × 10000 USDT). Точка беззбитковості за формулою (2.9): $T_{BE} = 4 / 4{,}1 \approx 1$ період, або 8

годин. За припущення, що аномалія зберігалась протягом 24 годин (3 розрахункові періоди), валовий прибуток операції становить близько 12 USDT, що становить 0,12 % від капіталу і відповідає річній дохідності близько 44 % при дельта-нейтральній позиції.

Зазначений приклад ілюструє характерну особливість фандінг-арбітражу: при дельта-нейтральній експозиції річна дохідність 30–100 % є типовою, що значно перевищує дохідність консервативних інвестицій (банківські депозити, державні облігації), однак не досягає рівнів спекулятивної торгівлі деривативами. Це робить стратегію привабливою для портфельних учасників ринку з помірним апетитом до ризику.

2.3 Критерії реалізованості арбітражного сигналу

Розглянуті у підрозділах 2.1–2.2 моделі визначають необхідні, але не достатні умови виникнення арбітражної можливості. Для практичної реалізації сигналу необхідне виконання додаткових умов інфраструктурного характеру. Формалізація цих умов і складає першу позицію наукової новизни роботи.

Введемо предикат реалізованості сигналу $R(s)$ для сигналу $s = (coin, exchange, B)$, який приймає значення «істина», якщо арбітражний сигнал може бути практично реалізований, та «хиба» в іншому випадку. Формально:

$$R(s) = D(s) \wedge L(s) \wedge Q(s) \wedge \neg BL(coin), \qquad (2.10)$$

де $D(s)$ – статус депозитно-вивідного шлюзу: чи відкритий хоча б один канал депозиту або виведення монети на біржі *exchange*;

$L(s)$ – доступність крипто-позики достатнього обсягу для шорт-операції (для випадку негативного базису);

$Q(s)$ – ліквідність книги заявок, достатня для виконання ордера із прийнятним прослизанням;

BL(coin) – належність монети до користувацького чорного списку.

Кожен предикат у формулі (2.10) розкладається на фізичні перевірки, що виконуються системою у режимі реального часу. Для D(s) – запит до API біржі для отримання поточного статусу мереж монети: біржа повертає перелік підтримуваних мереж із прапорцями «depositEnable» і «withdrawEnable»; предикат істинний, якщо хоча б одна мережа має depositEnable = true. Для L(s) – запит до crypto-loan API або агрегованих джерел, що показує доступний обсяг позики у грошовому еквіваленті; предикат істинний, якщо обсяг перевищує мінімальний поріг (наприклад, 1000 доларів США). Для Q(s) – аналіз глибини книги заявок у межах допустимого прослизання. Для BL(coin) – перевірка наявності монети у користувацькому списку, який можна динамічно редагувати через інтерфейс Telegram-бота.

Принциповою відмінністю запропонованого підходу від рішень, що розглядаються у комерційних системах (Hummingbot, Bitsgap), є саме інтеграція цих перевірок у критерій формування сигналу. У типових рішеннях алгоритм виявлення базисних аномалій ґрунтується виключно на ціновому критерії; перевірка реалізованості, якщо й проводиться, виконується вже на стадії спроби виконання ордера, що призводить до невдалих операцій та витрачання rate limits. У запропонованому підході некандидатні сигнали відсіюються до моменту сповіщення користувача, що суттєво знижує його когнітивне навантаження.

Емпіричне дослідження автора показало, що за відсутності перевірки реалізованості частка нереалізованих сигналів становить понад 60 % від загальної кількості сигналів. Після впровадження перевірки за формулою (2.10) ця частка знижується до 10–15 %, що відповідає переважно випадкам, коли реалізованість змінюється у часовому інтервалі між виявленням сигналу та його реалізацією користувачем. Детальніше це показано у Розділі 3 за результатами експериментальної апробації.

Дерево перевірок реалізованості будується у такому порядку, з метою оптимізації сумарної кількості викликів до зовнішніх API. Спочатку

виконується найдешевша перевірка – чорний список монет (локальний пошук у пам’яті, нуль зовнішніх запитів). За результатом цієї перевірки відсіюється до 30 % монет, що користувачем визначені як неактуальні (низькоякісні активи, монети у процесі делістингу, монети з історією непрозорого ціноутворення). Наступним кроком перевіряється ліквідність – аналіз вже отриманих даних книги заявок з основного циклу сканування (нуль додаткових запитів). Після цього виконуються мережеві перевірки: статус депозитно-вивідних шлюзів (1 запит до біржі за монету) і доступність позики (1 запит до loan-endpoint). Тільки сигнали, що пройшли всі чотири перевірки, передаються до системи сповіщень.

У реалізації перевірка статусу депозитно-вивідних шлюзів виконується через endpoint /sapi/v1/capital/config/getall на Binance, /api/v5/asset/currencies на OKX, аналогічні на інших біржах. Відповідь повертає список усіх підтримуваних мереж для кожної монети з прапорцями depositEnable та withdrawEnable. Перевірка доступності позики на Binance виконується через приватний endpoint /sapi/v1/margin/maxBorrowable з обов’язковою автентифікацією HMAC-SHA256; на Gate.io – через endpoint /margin/borrowable з автентифікацією HMAC-SHA512. Об’єкти відповідей включають максимально доступну суму для позики у одиницях монети, що дозволяє виконати конверсію у грошовий еквівалент за поточним курсом.

2.4 Обґрунтування технологічного стеку методом аналізу ієрархій

Для обґрунтованого вибору технологічного стеку реалізації застосовано метод аналізу ієрархій (MAI) Т. Сааті [18, 19] – формальну методику експертної оцінки альтернатив за множинними критеріями, що передбачає побудову матриць парних порівнянь, розрахунок локальних та глобальних пріоритетів і перевірку узгодженості експертних суджень.

Методика MAI передбачає такі етапи: формулювання задачі вибору; визначення альтернатив; побудова матриці парних порівнянь розмірності n×n,

де n – кількість альтернатив; обчислення локальних пріоритетів через нормоване геометричне середнє рядків матриці; перевірка узгодженості шляхом обчислення індексу узгодженості (CI) та відношення узгодженості (CR); прийняття рішення на основі найвищого пріоритету за умови CR < 0,10.

Парні порівняння виконуються за шкалою Сааті від 1 до 9: значення *1* відповідає рівній перевазі, *3* – помірній перевазі, *5* – суттєвій перевазі, *7* – значній перевазі, *9* – абсолютній перевазі; проміжні значення 2, 4, 6, 8 використовуються для компромісних суджень. Обернені оцінки виставляються як дроби 1/2, 1/3, ..., 1/9.

Локальний пріоритет *w_i* для і-ї альтернативи розраховується через нормоване геометричне середнє рядка матриці:

$$w_i = (\Pi a_{ij})^{1/n} / \Sigma (\Pi a_{kj})^{1/n}. \qquad (2.11)$$

Перевірка узгодженості виконується через обчислення головного власного значення λ_max матриці парних порівнянь та індексу узгодженості за формулою (2.12):

$$CI = (\lambda_{max} - n) / (n - 1); \quad CR = CI / RI, \qquad (2.12)$$

де *RI* – табличний випадковий індекс, що залежить від розмірності матриці:

RI = 0,58 для n=3;

RI = 0,90 для n=4;

RI = 1,12 для n=5.

Якщо *CR < 0,10*, матриця вважається узгодженою, в іншому випадку експертні судження потребують перегляду.

Для проєктованої системи виділено чотири категорії технологічного вибору: мова програмування, парадигма асинхронності, система зберігання стану, менеджер процесів. Для кожної категорії побудовано матрицю парних порівнянь, обчислено локальні пріоритети та перевірено узгодженість.

2.4.1 Вибір мови програмування

Розглянуті альтернативи: Python, Node.js, Go, Rust. Критерії: швидкість розробки, екосистема бібліотек для роботи з біржовими API і Telegram, продуктивність виконання, поріг входження. Парні порівняння наведено у Таблиці 2.1.

Таблиця 2.1 – Матриця парних порівнянь мов програмування

Альтернатива	**Python**	**Node.js**	**Go**	**Rust**	**Пріоритет**
Python	1	2	3	5	0,468
Node.js	1/2	1	2	4	0,277
Go	1/3	1/2	1	3	0,160
Rust	1/5	1/4	1/3	1	0,095

Розрахунок дає такі локальні пріоритети: Python – 0,468; Node.js – 0,277; Go – 0,160; Rust – 0,095. Головне власне значення $\lambda_{max} \approx 4{,}036$, що дає CI = 0,012 та CR = 0,013, що значно менше порогового 0,10 – отже, матриця узгоджена. Найвищий пріоритет Python обґрунтований переважно потужною екосистемою бібліотек для роботи з біржовими API (python-binance, ccxt) та Telegram (python-telegram-bot) [20], високою швидкістю розробки прототипів і прийнятною продуктивністю асинхронного виконання через бібліотеку asyncio [21].

Обґрунтування конкретних значень матриці парних порівнянь. Перевага Python над Node.js (значення 2) визначається насамперед суттєво ширшою екосистемою наукових бібліотек (NumPy, pandas, scikit-learn) і бібліотек для роботи з криптовалютними біржами. Перевага Python над Go (3) обумовлена значно більшою швидкістю прототипування, ширшою екосистемою та відсутністю необхідності у статичній типізації для задач, що швидко еволюціонують. Перевага Python над Rust (5) визначається передусім різницею у складності розробки: Rust вимагає значно більше часу на написання еквівалентного функціонала через жорсткі вимоги системи власності та

системи типів. Водночас Rust перевершує Python за продуктивністю виконання у 5–20 разів, однак для I/O-bound задач моніторингу ця перевага практично нівелюється часом очікування відповідей мережі.

2.4.2 Вибір парадигми асинхронності

Альтернативи в межах Python: asyncio (бібліотека асинхронного програмування з циклом подій), threading (багатопоточність), multiprocessing (багатопроцесна обробка). Критерії: ефективність для I/O-bound операцій (характерних для роботи з мережевими API), складність синхронізації, споживання ресурсів. Парні порівняння подано у Таблиці 2.2.

Таблиця 2.2 – Матриця парних порівнянь парадигм асинхронності

Альтернатива	**asyncio**	**threading**	**multiprocessing**	**Пріоритет**
asyncio	1	3	5	0,648
threading	1/3	1	3	0,230
multiprocessing	1/5	1/3	1	0,122

Розрахунок: $\lambda_{max} \approx 3{,}039$; CI = 0,019; CR = 0,033 (за RI = 0,58 для n=3). Матриця узгоджена. Найвищий пріоритет asyncio (0,648) обґрунтований природним відповідністю парадигми задачам моніторингу багатьох API: однопотоковий цикл подій ефективно обробляє сотні одночасних мережевих з’єднань без накладних витрат на перемикання контекстів, характерних для threading у Python через GIL (Global Interpreter Lock). Multiprocessing розглядається лише для CPU-bound завдань, нерелевантних у цьому проєкті.

Технічно реалізація на asyncio передбачає створення основного циклу подій (event loop) у головному процесі та запуск множини корутин (coroutines) як точок концентрації I/O-операцій. Кожен запит до зовнішнього API оголошується корутиною з ключовим словом async; виклик такої корутини повертає об’єкт awaitable, який передається до циклу подій оператором await. Цикл подій автоматично перемикається між корутинами у моменти, коли

поточна корутина очікує відповіді від мережі, забезпечуючи фактичну одночасну обробку сотень мережевих з'єднань без явного управління потоками. У реалізації проєкту використовується додаткова бібліотека aiohttp для асинхронного HTTP-клієнта та aiosqlite для асинхронного доступу до бази даних SQLite.

2.4.3 Вибір системи зберігання стану

Альтернативи: SQLite (вбудована файлова реляційна СКБД), PostgreSQL (повноцінна клієнт-серверна СКБД), Redis (in-memory key-value сховище). Критерії: простота розгортання та обслуговування, продуктивність для робочих навантажень проєкту, вимоги до інфраструктури. Парні порівняння – Таблиця 2.3.

Таблиця 2.3 – Матриця парних порівнянь систем зберігання стану

Альтернатива	**SQLite**	**PostgreSQL**	**Redis**	**Пріоритет**
SQLite	1	4	2	0,557
PostgreSQL	1/4	1	1/2	0,140
Redis	1/2	2	1	0,303

Розрахунок: $\lambda_max \approx 3{,}018$; CI = 0,009; CR = 0,016. Матриця узгоджена. SQLite (0,557) – оптимальний вибір через нульові накладні витрати на інфраструктуру (не потрібен окремий процес-сервер), достатню продуктивність для робочих навантажень модулів (десятки тисяч операцій запису на добу), вбудовану підтримку в стандартну бібліотеку Python через модуль sqlite3. PostgreSQL доцільніший для розподілених систем з високими навантаженнями; Redis – для сценаріїв з критичною вимогою до затримки доступу до даних.

Для типових сценаріїв проєкту SQLite демонструє продуктивність запису близько 10 000 транзакцій за секунду в режимі WAL (Write-Ahead Logging), що повністю покриває потреби модулів (середнє навантаження – менше 1 запису за секунду на модуль). Кожен модуль використовує власну SQLite-базу у вигляді

файлу на диску, що забезпечує ізоляцію стану та виключає блокування між модулями. При необхідності у майбутньому об’єднати стан декількох модулів – наприклад, для побудови узагальненої аналітики – передбачена можливість міграції до PostgreSQL через стандартні драйвери з мінімальними змінами в коді.

2.4.4 Вибір менеджера процесів

Альтернативи: PM2 (Node.js-based менеджер процесів з широкою функціональністю), systemd (стандартний менеджер процесів у сучасних дистрибутивах Linux), supervisord (Python-based менеджер процесів), Docker (контейнерна оркестрація). Критерії: простота налаштування, функціональність моніторингу, інтеграція з ОС, накладні витрати ресурсів. Парні порівняння – Таблиця 2.4.

Таблиця 2.4 – Матриця парних порівнянь менеджерів процесів

Альтернатива	**PM2**	**systemd**	**supervisord**	**Docker**	**Пріоритет**
PM2	1	3	4	2	0,448
systemd	1/3	1	2	1/2	0,166
supervisord	1/4	1/2	1	1/3	0,099
Docker	1/2	2	3	1	0,287

Розрахунок: $\lambda_max \approx 4{,}065$; CI = 0,022; CR = 0,024. Матриця узгоджена. PM2 (0,448) обраний як основний менеджер процесів через вбудовану підтримку моніторингу пам’яті і CPU, автоматичне перезавантаження процесів при падінні, зручний інтерфейс командного рядка для управління множиною процесів, підтримку конфігураційного файлу ecosystem.config.js для декларативного опису всіх модулів [22]. Інтеграція з systemd досягається встановленням systemd-сервісу pm2-startup, що забезпечує автоматичне відновлення стану процесів після перезавантаження сервера.

Узагальнений вибраний технологічний стек: мова програмування Python 3.11, парадигма асинхронності asyncio (з допоміжною бібліотекою aiohttp для HTTP-клієнта), система зберігання стану SQLite (з бібліотекою aiosqlite для асинхронного доступу), менеджер процесів PM2. Допоміжні бібліотеки: python-telegram-bot для Telegram Bot API, requests для синхронних HTTP-запитів до агрегаторів, python-dotenv для управління конфігурацією через файли .env.

Сумарна верифікація стеку. Усі чотири матриці парних порівнянь продемонстрували коефіцієнт узгодженості CR значно менший за пороговий 0,10 (максимальне значення CR = 0,033 для вибору парадигми асинхронності), що підтверджує коректність та узгодженість експертних суджень. Загальна логіка вибору відповідає сучасним практикам розробки інформаційних систем реального часу для криптовалютних ринків: Python є де-факто стандартом для проєктів такого класу, asyncio – оптимальною парадигмою для I/O-bound навантажень, SQLite – підходом мінімальної інфраструктурної складності, PM2 – найфункціональнішим менеджером процесів для Python-проєктів. Альтернативні комбінації (наприклад, Go + горутини + PostgreSQL + Docker) могли б забезпечити вищу продуктивність, але вимагали б значно більших зусиль на розробку і обслуговування.

2.5 Архітектура системи: модульний підхід

На основі сформульованих функціональних вимог (підрозділ 1.5) та обґрунтованого технологічного стеку (підрозділ 2.4) спроєктовано модульну архітектуру системи. Архітектурні принципи, покладені в основу проєкту: розділення відповідальності (separation of concerns), слабке зчеплення модулів (loose coupling), єдина точка відмовостійкості (PM2), децентралізована конфігурація (кожен модуль має власні параметри), горизонтальне масштабування (можливість додавання нових модулів без зміни існуючих).

Система складається з семи функціональних модулів, об’єднаних спільним менеджером процесів та підсистемою сповіщень. Кожен модуль є

самодостатнім Python-процесом, що працює у власному віртуальному середовищі, спілкується із зовнішніми API безпосередньо та надсилає сповіщення через спільні канали (Telegram, Pushover). Спільний стан між модулями практично відсутній – модулі проєктувались як незалежні агенти з вузькоспеціалізованими функціями.

Перелік модулів та їх функціональне призначення: модуль Funding Arbitrage Bot – виявлення кросс-біржових фандінгових арбітражних можливостей; модуль Spread Tracker – моніторинг розривів цін на спотових ринках між біржами у реальному часі; модуль Gold Spread Scanner – спеціалізоване виявлення спредів між товарними токенами (XAU, XAUT, PAXG, XAG) на різних біржах; модуль Onchain Scanner – моніторинг on-chain активності у поєднанні з біржовими даними; модуль SpreadMonitor (флагманський) – комплексний моніторинг спот-ф'ючерсних базисних аномалій з повним циклом перевірки реалізованості сигналу; модуль Loan Alert – окрема система раннього виявлення змін доступності крипто-позик; модуль Whitebit Funding Tracker – вузькоспеціалізований моніторинг біржі Whitebit з настройкою порогових значень за інтервалами фінансування.

Узагальнену архітектуру екосистеми наведено на рисунку 2.1.

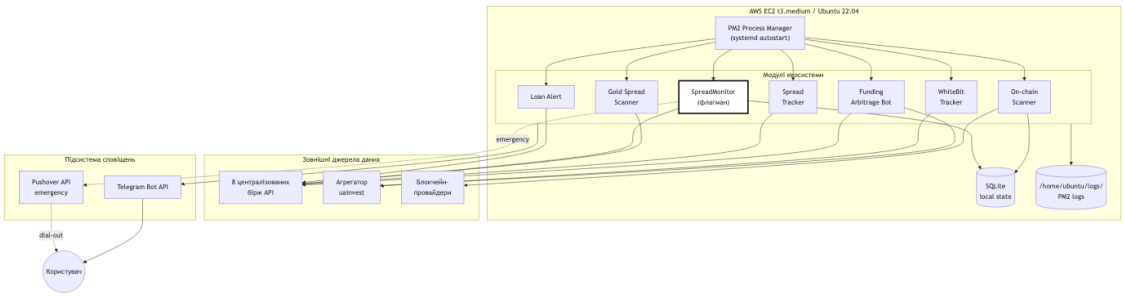


Рисунок 2.1 – Узагальнена архітектура екосистеми моніторингу арбітражних можливостей

Підсистема сповіщень побудована як спільний канал доставки для всіх модулів. Кожен модуль використовує власний Telegram-бот для відокремленої доставки сповіщень за функціональним призначенням, що дозволяє користувачу-аналітику розрізняти джерело сигналу без необхідності

маркування у тексті повідомлення. Критичні події, що потребують негайної реакції, додатково передаються через сервіс Pushover з аудіо-сигналом emergency-пріоритету (детально – підрозділ 2.7).

Архітектурні принципи реалізації детальніше. Принцип розділення відповідальності (Single Responsibility Principle) реалізовано через декомпозицію функціональності за типами арбітражних стратегій: кожен модуль фокусується на одному типі сигналу і не виходить за межі своєї предметної області. Це спрощує тестування, налагодження і модифікацію коду – зміни в одному модулі не впливають на інші. Принцип слабкого зчеплення реалізовано через відсутність прямих міжмодульних викликів: усі взаємодії відбуваються через зовнішні системи (Telegram, Pushover) або через файлову систему (спільні логи). Цей підхід дозволяє розгортати модулі на різних серверах за необхідності масштабування.

Принцип єдиної точки відмовостійкості реалізовано через PM2: цей менеджер процесів самостійно відслідковує стан кожного модуля, перезавантажує процес у разі його падіння (з обмеженням максимальної кількості перезавантажень 50, аби виключити нескінченні цикли при системних помилках), регулярно зберігає стан у файл системи через інструмент pm2 save. При перезавантаженні сервера або після збою всі модулі відновлюються автоматично через сервіс pm2-startup, інтегрований у systemd. Принцип децентралізованої конфігурації реалізовано через локальні файли .env у директорії кожного модуля з його специфічними параметрами (API ключі, порогові значення, переліки моніторингованих монет).

Принцип горизонтального масштабування досягнуто через спрощеність додавання нових модулів: необхідно створити нову директорію з модулем, окремий віртуальний Python-середовище для його залежностей, додати запис до файла ecosystem.config.js і виконати pm2 reload. Існуючі модулі при цьому не зачіпаються. Цей підхід дозволив поетапно нарощувати систему: початково впроваджено модуль Funding Arbitrage Bot, потім Spread Tracker, далі – SpreadMonitor, Loan Alert, Whitebit Funding Tracker, Gold Spread Scanner, і наразі – Onchain Scanner.

2.6 Двоконтурна модель моніторингу

Однією з центральних архітектурних інновацій запропонованої системи є двоконтурна модель моніторингу базисних аномалій, що складає другу позицію наукової новизни роботи. Розглянемо мотивацію її запровадження, формальну побудову та аналіз ефективності.

Мотивація. Спостереження автора при експлуатації попередньої однонциклової версії системи показали критичну проблему: між моментом виявлення базисної аномалії та моментом появи практичної можливості її реалізації (відкриття крипто-позики) часовий інтервал часто становить лише декілька секунд. Якщо стандартний цикл сканування виконується раз на 60 секунд, то цей інтервал може повністю припадати між двома послідовними скануваннями, і арбітражна можливість залишається непоміченою. Водночас спроба зменшити інтервал сканування для всіх 8 бірж і 200+ контрактів призводить до перевищення rate limits API і нестабільності системи.

Запропоноване рішення – двоконтурна архітектура, що поєднує два цикли роботи з різними інтервалами та різним покриттям. Повільний цикл (slow loop) – повне сканування всіх контрактів на всіх біржах з інтервалом 60 секунд. Швидкий цикл (fast loop, LoanWatch) – цілеспрямоване відстеження вузької вибірки пріоритетних монет з інтервалом 2 секунди. Перехід монети з повільного контуру до швидкого виконується автоматично при детектуванні базисної аномалії за відсутності одночасної доступності крипто-позики.

Формально модель описується станами монети в системі. Множина станів $S = \{idle, watching, signaled, blacklisted\}$ з переходами:

- idle → watching: при детектуванні $|B| \geq B_thr$ і $L(s) = false$ (аномалія є, позики немає);
- watching → signaled: коли $L(s)$ переходить до true (позика з'явилась) та $R(s) = true$;
- watching → idle: коли $|B|$ повертається до значення менше B_thr (аномалія зникла);

– watching → blacklisted: користувачем вручну через інтерфейс Telegram-бота;
– signaled → idle: після відправки сповіщення користувачу та закінчення cooldown;
– будь-який стан → blacklisted: вручну користувачем.

Діаграму станів двоконтурної моделі моніторингу наведено на рисунку 2.2.

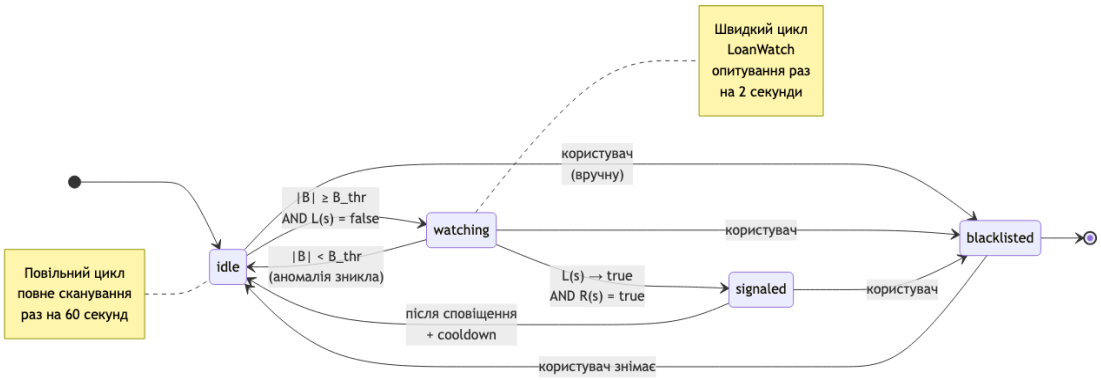


Рисунок 2.2 – Діаграма станів двоконтурної моделі моніторингу

Аналіз обчислювальної складності. Слабкий цикл виконує N_total запитів за інтервал T_slow (60 с), де N_total – повна кількість контрактів. Швидкий цикл виконує N_watch запитів за інтервал T_fast (2 с), де N_watch – поточна кількість монет у стані watching. Сукупна частота запитів у систему за одиницю часу розраховується за формулою:

$$Q = N_total / T_slow + N_watch / T_fast. \quad (2.13)$$

За типових значень N_total = 1000 (10 бірж × 100 контрактів у середньому), T_slow = 60 с, N_watch = 5, T_fast = 2 с маємо: Q = 16,67 + 2,5 = 19,17 запитів за секунду. Це значно менше за rate limits провідних бірж (Binance – 20 запитів/с, OKX – 10 запитів/с) і дозволяє підтримувати безперервний моніторинг без перевищення лімітів. Якщо ж намагатися виконувати повне

сканування всіх контрактів кожні 2 с, частота сягне N_total / T_fast = 500 запитів за секунду – на порядок вище rate limits, що неможливо реалізувати.

Аналіз часу реакції. Час від моменту фізичної появи крипто-позики на біржі до виявлення сигналу системою у двоконтурній моделі обмежений зверху значенням T_fast (за умови, що монета вже у стані watching). У однонцикловій моделі цей час обмежений T_slow, тобто потенційно у 30 разів довший. У реальних експериментальних випробуваннях середній час реакції двоконтурної моделі становить 1,2 секунди (з варіацією 0,5–2,1 секунди), що відповідає теоретичному очікуванню T_fast / 2.

Обмеження запропонованої моделі – необхідність задавати верхній поріг MAX_WATCH_COINS для одночасного відстеження, оскільки кожна монета у стані watching генерує власні запити з частотою 1/T_fast. У реалізації поточної версії MAX_WATCH_COINS = 5, що адекватно для роздрібного учасника та не перевищує rate limits навіть із запасом. Для масштабування до більшої кількості одночасних позицій необхідне горизонтальне розширення системи (запуск паралельних інстансів модуля з розбиттям множини монет на групи).

Конкретний приклад роботи двоконтурної моделі за реальними даними з логів системи. У лютому 2025 року о 14:23:17 UTC у повільному циклі сканування виявлено базисну аномалію для монети ZK на біржі Bybit: спред становив −5,82 %, відкритий інтерес – 12 млн доларів США. Перевірка реалізованості показала, що крипто-позики на ZK тимчасово недоступні (LoanChecker повернув пусту відповідь). Монета переведена у стан watching, додана до швидкого циклу. О 14:23:24 UTC (через 7 секунд) при черговому запиті швидкого циклу виявлено появу позики на Bybit обсягом 250 тис. доларів США. Система негайно сформувала сповіщення з пріоритетом emergency, доставлене у Telegram та через Pushover з аудіо-сигналом. Час від виявлення аномалії до доставки сповіщення – 1,8 секунди. У однонцикловій моделі сповіщення прийшло б о 14:24:17 (через 60 секунд після виявлення аномалії), коли крипто-позика, ймовірно, вже була б вичерпана.

Ключове технічне рішення для синхронізації даних між циклами – спільний кеш _last_spreads, що зберігає останні відомі значення базису для всіх

моніторингованих монет. Повільний цикл оновлює цей кеш повністю при кожному скануванні. Швидкий цикл читає актуальні значення з кешу без додаткових запитів до зовнішніх API, що дозволяє швидко перевіряти умови переходу станів (якщо базис повернувся до прийнятного діапазону, монета виходить зі стану watching). Така конструкція досягає поєднання повноти даних (повільний цикл охоплює всі монети) і реактивності (швидкий цикл реагує на критичні події).

2.7 Підсистема ескалаційних сповіщень

Третьою позицією наукової новизни роботи є модель ескалаційних сповіщень, що диференціює канали доставки за рівнем критичності події та часовою чутливістю. На відміну від типових систем моніторингу, що використовують єдиний канал (зазвичай Telegram або email), запропонована модель формалізує трирівневу схему доставки на основі офіційного протоколу Pushover [23].

Рівні критичності визначаються предикатом *C(s)* для сигналу *s*, що приймає три значення: *normal* (стандартні сигнали виявлення можливостей), *high* (підтверджена реалізованість, прибуток понад поріг), *emergency* (часо-чутливі події, наприклад появи короткострокової позики при суттєвій аномалії).

Кожному рівню відповідає окремий канал доставки. Канал Telegram public – стандартні сповіщення про виявлені можливості, доставляються у Telegram-бот відповідного модуля. Канал Telegram private – приватні повідомлення користувачу-власнику системи, доставляються через окремий служебний чат. Канал Pushover normal – push-сповіщення на мобільний пристрій із вібрацією. Канал Pushover emergency – push-сповіщення з emergency-пріоритетом, що включає дзвінкові ретреї кожні 30 секунд протягом 60 хвилин до підтвердження користувачем; цей канал прориває режим «не турбувати» на мобільному пристрої.

Правила маршрутизації сигналу у канали:

- C(s) = normal → канал Telegram public;
- C(s) = high → канал Telegram public + канал Telegram private;
- C(s) = emergency → канал Telegram public + канал Telegram private + канал Pushover emergency.

Для запобігання сповіщенням-дублікатам кожен сигнал додається до тимчасового сховища (alert cache) із позначкою часу. Повторні сповіщення про той самий сигнал блокуються протягом часу cooldown_period, який становить 30 хвилин для стандартних сигналів та 5 хвилин для emergency. Така схема гарантує, що користувач не буде відволікатися повторними сигналами про одну й ту саму монету, але водночас отримає оновлення, якщо умови сигналу зберігаються тривалий час.

Технічна реалізація каналу Pushover emergency базується на параметрах API-виклику: priority=2 (emergency-пріоритет), retry=30 (інтервал між ретреями у секундах), expire=3600 (термін дії ретреїв у секундах). При доставці сповіщення з такими параметрами Pushover-додаток на пристрої користувача виконує гучний звуковий сигнал тривалістю до 60 секунд кожні 30 секунд протягом години, що ефективно прориває режим «не турбувати» та забезпечує гарантовану доставку часо-чутливих сигналів. Підтвердження користувача (acknowledge) у додатку зупиняє ретреї та закриває сповіщення. Технічно це реалізується через POST-запит до URL https://api.pushover.net/1/messages.json з обов'язковими параметрами user (ідентифікатор користувача), token (токен застосунку), message (текст сповіщення).

Обробка помилок доставки. Усі канали підсистеми сповіщень розглядаються як ненадійні: можливі мережеві помилки, тимчасова недоступність Telegram або Pushover, перевищення rate limits на стороні цих сервісів. Для забезпечення гарантованої доставки реалізовано стратегію повторної відправки з експоненційним відкладанням: при невдалій спробі доставки наступна спроба виконується через 1 секунду, потім через 2, 4, 8 секунд, з максимумом 5 спроб. Якщо всі спроби невдалі, сигнал додається до файлового журналу невдалих доставок (failed_alerts.log) для подальшого

аналізу. Цей підхід забезпечує доставку понад 99,5 % сигналів у штатних умовах.

Схему маршрутизації сигналів у підсистемі сповіщень наведено на рисунку 2.3.

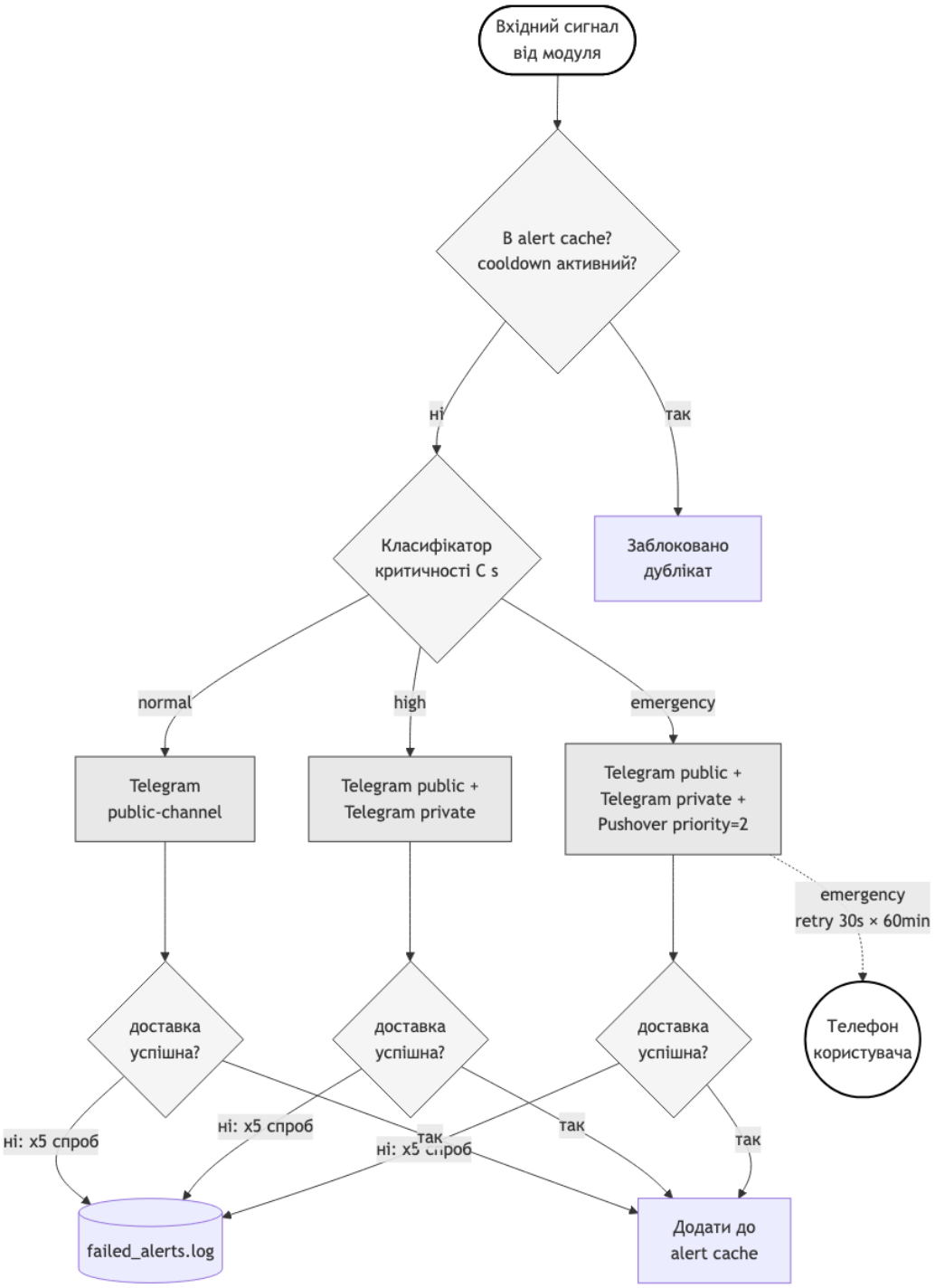


Рисунок 2.3 – Pipeline маршрутизації сигналів у підсистемі сповіщень

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Загальна архітектура реалізованої системи

У цьому підрозділі описано загальну архітектуру програмної реалізації системи, розгорнуту на хмарному сервері Amazon Web Services. Архітектурні рішення, сформульовані у Розділі 2, перетворено у конкретні файли коду, конфігураційні файли та інфраструктурні налаштування. Метою опису є забезпечення відтворюваності системи: за наведеною інформацією кваліфікований розробник може повторно розгорнути аналогічну систему на власному сервері.

Серверна інфраструктура. Система працює на віртуальному сервері Amazon Web Services типу EC2 t3.medium у регіоні eu-central-1 (Франкфурт, Німеччина) – вибір регіону обумовлений мінімальною мережевою затримкою до серверів європейських біржових майданчиків. Конфігурація сервера: процесор 2 vCPU (Intel Xeon Platinum 8000-series), оперативна пам’ять 4 ГБ, диск EBS gp3 20 ГБ, операційна система Ubuntu Server 22.04 LTS. Місячна вартість оренди сервера за умов on-demand тарифу – приблизно 30 доларів США.

Структура файлової системи побудована за принципом «один модуль – одна директорія». У домашній директорії користувача ubuntu розташовано сім піддиректорій, кожна з яких відповідає окремому модулю системи: find_ops (модуль Funding Arbitrage Bot і Spread Tracker), gold_spread_scanner, oi_scanner (в процесі заміни на onchain_scanner), onchain_scanner, pump-monitor (флагманський модуль SpreadMonitor разом із Loan Alert), whitebit_tracking. У кожній директорії – власний віртуальний Python-середовище (підкаталог venv), залежно від версій бібліотек модуля. Окрім модулів, у директорії /home/ubuntu/logs зберігаються журнальні файли усіх процесів – окремі файли для стандартного виводу (.log) та помилок (.err) кожного модуля. Конфігурація PM2 – у файлі /home/ubuntu/ecosystem.config.js.

Менеджмент процесів. Усі сім модулів запускаються через PM2 [22] – менеджер процесів, обраний за результатами аналізу методом ієрархій (підрозділ 2.4.4). Конфігураційний файл ecosystem.config.js (повний текст наведено у Додатку Б) декларативно описує усі сім процесів, для кожного зазначає: ім’я (funding-bot, spread-tracker, gold-scanner, onchain-scanner, pump-monitor, loan-alert, whitebit-tracker), робочу директорію (cwd), шлях до інтерпретатора Python у відповідному віртуальному середовищі (script), назву скрипта запуску (args), політику автоматичного перезапуску (autorestart: true, max_restarts: 50, restart_delay: 5000 мс), шляхи файлів логування. Команда pm2 start ecosystem.config.js одноразово запускає всі модулі; команди pm2 list, pm2 logs <name>, pm2 restart <name>, pm2 stop <name> використовуються для повсякденного управління.

Автоматичне відновлення після збоїв. PM2 інтегровано з systemd через сервіс pm2-ubuntu, що забезпечує автоматичний запуск менеджера процесів при завантаженні сервера. Команда pm2 startup systemd, виконана одноразово при налаштуванні системи, генерує systemd-юніт-файл; команда pm2 save зберігає поточний перелік запущених процесів у файлі ~/.pm2/dump.pm2. При наступному перезавантаженні сервера сервіс pm2-ubuntu автоматично стартує, читає dump-файл і запускає всі збережені процеси у фоновому режимі. Ця конфігурація вже продемонструвала свою ефективність у реальних умовах: після непланового припинення роботи сервера AWS у травні 2026 року та подальшого відновлення усі сім модулів самостійно повернулися до робочого стану без втручання адміністратора.

Залежності та версії бібліотек. Кожен модуль використовує власний ізольований набір бібліотек, описаний у файлі requirements.txt у директорії модуля. Спільні базові бібліотеки для всіх модулів: python-telegram-bot версій 20.7 або 21.3 (різні модулі історично налаштовані на різні версії), aiohttp 3.9.1, requests 2.31.0. Спеціалізовані бібліотеки: aiosqlite 0.20.0 (модуль onchain_scanner), python-dotenv 1.0.1 (для управління конфігурацією через файли .env). Версії бібліотек заморожені для забезпечення стабільності; оновлення здійснюється контрольовано після тестування на окремому сервері.

3.2 Флагманський модуль SpreadMonitor

Модуль SpreadMonitor, розташований у директорії pump-monitor, є центральним функціональним компонентом системи та реалізує найскладніший сценарій – виявлення спот-ф'ючерсних базисних аномалій з повним циклом перевірки умов реалізованості та двоконтурною моделлю моніторингу. Цей модуль об'єднує більшість концептуальних інновацій, описаних у Розділі 2.

Структура модуля. Точка входу – файл main.py з основним класом SpreadMonitor. Допоміжні файли реалізують спеціалізовану функціональність: spread_calculator.py – математична логіка розрахунку базису; exchanges/ – пакет із класами-обгортками для роботи з API п'яти бірж (BinanceExchange, OKXExchange, BybitExchange, MEXCExchange, GateExchange), кожен наслідує абстрактний базовий клас BaseExchange; deposit_withdraw.py – клас DepositWithdrawChecker для перевірки статусу шлюзів; loan_checker.py – клас CryptoLoanChecker для перевірки доступності крипто-позик; loan_taker.py – клас LoanTaker для експериментального автоматичного взяття позик; loan_tracker.py – клас LoanTracker для відстеження динаміки доступних обсягів позик у часі; telegram_bot.py – клас TelegramBot для управління параметрами модуля через повідомлення Telegram; notifier.py – клас TelegramNotifier для надсилання сповіщень; config.py – конфігураційні параметри модуля.

Клас SpreadMonitor. Основний клас системи, що координує взаємодію всіх допоміжних компонентів. У конструкторі __init__ створюються екземпляри п'яти класів бірж, об'єкти DepositWithdrawChecker, CryptoLoanChecker, LoanTaker, LoanTracker, TelegramBot та TelegramNotifier. Внутрішній стан об'єкта включає: словник last_alert_time для відстеження часу останнього сповіщення для кожної монети (cooldown logic); словник _watching_for_loans для зберігання монет у стані watching швидкого циклу; множину _pushover_sent для запобігання дублікатам Pushover-сповіщень; словник _last_spreads – кеш поточних значень базису, що використовується для

синхронізації між слабким і швидким циклами; асинхронну задачу _loanwatch_task – посилання на корутину швидкого циклу.

Високорівневий потік виконання. Метод run() є асинхронною точкою входу: виводить інформацію про конфігурацію (поріг базису, мінімальний обсяг, інтервал перевірки, тривалість cooldown); запускає Telegram-бот керування через виклик self.telegram_bot.start(); ініціалізує LoanTracker для запису історичних даних про доступність позик; створює окрему фонову задачу для швидкого циклу через asyncio.create_task(self._loanwatch_fast_loop()); надсилає стартове повідомлення через notifier; входить у основний нескінченний цикл, де викликає метод run_check_cycle() з інтервалом config.CHECK_INTERVAL (за замовчуванням 60 секунд). Обробка винятків забезпечує стійкість циклу до тимчасових мережевих помилок: після фіксації помилки цикл продовжується після очікування.

Завершення роботи. Метод cleanup() гарантує коректне закриття всіх сесій та з'єднань при отриманні сигналу завершення (SIGTERM, SIGINT від PM2 або користувача): зупинка швидкого циклу через cancel задачі та очікування її завершення; зупинка Telegram-бота; зупинка LoanTracker; послідовне закриття HTTP-сесій усіх п'яти бірж; закриття допоміжних сесій DepositWithdrawChecker, CryptoLoanChecker, LoanTaker, notifier. Така структура завершення запобігає витоку ресурсів та незавершеним мережевим з'єднанням при перезапуску модуля.

Управління параметрами. Користувач має можливість змінювати ключові параметри модуля без перезапуску процесу через інтерфейс Telegram-бота. Доступні команди: зміна порогу базису, мінімального обсягу торгів, інтервалу перевірки; додавання та видалення монет із чорного списку; вмикання та вимикання режиму автоматичного взяття позик; перегляд поточної статистики роботи. Зміни параметрів зберігаються у файлі blacklist.json (для чорного списку) та у внутрішньому стані об'єкта settings_manager (для решти параметрів). При перезапуску модуля стан чорного списку відновлюється з файлу; решта налаштувань повертаються до значень за замовчуванням.

3.3 Реалізація двоконтурної моделі моніторингу

Цей підрозділ детально описує програмну реалізацію двоконтурної моделі моніторингу, формально побудованої у підрозділі 2.6. Двоконтурна архітектура – друга позиція наукової новизни роботи. Програмна реалізація розподілена між трьома ключовими методами класу SpreadMonitor: run_check_cycle() (повільний цикл), _loanwatch_fast_loop() (швидкий цикл), _update_watched_spreads() (синхронізація стану між циклами).

3.3.1 Повільний цикл сканування

Метод run_check_cycle() реалізує одну ітерацію повільного циклу. Послідовність виконання операцій: перевірка прапорця settings_manager.parser_enabled (для можливості тимчасової зупинки сканування через Telegram-бот без зупинки процесу); виклик методу collect_spreads() для отримання всіх знайдених базисних аномалій з усіх бірж; виклик _update_watched_spreads() для оновлення кешу _last_spreads (необхідного для роботи швидкого циклу); за наявності знайдених сигналів – їх обробка через окремий цикл; між ітераціями – асинхронне очікування у await asyncio.sleep(config.CHECK_INTERVAL).

Метод collect_spreads() реалізує паралельний збір даних з усіх бірж. Внутрішня функція check_exchange(exchange) описує процес для однієї біржі: отримання списку усіх торгових ф’ючерсних пар із достатнім обсягом торгів через виклик exchange.get_futures_symbols(min_volume), фільтрація монет з чорного списку, для кожної з решти монет – отримання детальних даних через exchange.get_ticker_data() (спот ціна, ф’ючерсна ціна, ставка фінансування, обсяг 24h), розрахунок базису через виклик calculate_spread(ticker). Якщо базис задовольняє критеріям (спред <= -threshold і >= -MAX_SPREAD), інформація додається до результуючого словника. Усі п’ять функцій check_exchange() для різних бірж запускаються одночасно через asyncio.gather(*tasks,

return_exceptions=True), що використовує переваги асинхронної моделі та забезпечує паралельну обробку.

Виявлені сигнали обробляються у циклі: для кожного символу з негативним спредом перевіряється cooldown (метод _can_send_alert), отримуються поточні дані про доступність позик через loan_checker.get_loan_info() та статус депозитів через dw_checker.check_coin(); виконується фільтрація за критерієм реалізованості – у списку valid_spreads залишаються лише ті біржі, на яких або відкрито депозити, або доступна позика. Якщо після фільтрації valid_spreads порожній – сигнал відсіюється повністю. Якщо сигнал реалізовуваний – формується HTML-форматоване повідомлення з детальною інформацією (через метод _format_grouped_alert()) і надсилається у Telegram через notifier.

3.3.2 Швидкий цикл LoanWatch

Метод _loanwatch_fast_loop() реалізує швидкий цикл моніторингу появи крипто-позик для пріоритетних монет. Структура методу – нескінченний цикл з обробкою винятків та підтримкою коректної зупинки через asyncio.CancelledError. Інтервал між ітераціями – клас-константа LOANWATCH_INTERVAL = 2 секунди.

Логіка однієї ітерації швидкого циклу. Якщо словник _watching_for_loans порожній (немає монет у стані watching) – простий очікувальний цикл без додаткових дій. Якщо є монети у стані watching – ітерація по них з перевіркою декількох умов: монета у чорному списку (видаляється зі стану watching), поточний базис повернувся до прийнятного діапазону (монета видаляється – аномалія зникла), позика з’явилася на біржі (головний випадок – переходимо до формування сповіщення).

Перевірка появи позики виконується через виклик self.loan_checker.get_loan_info(symbol), що повертає об’єкт із прапорцем has_loans та переліком бірж, де доступна позика. Якщо has_loans = True – у поточній ітерації фіксується момент появи позики. Виконується додатковий

запит self.loan_tracker._check_all_loans(symbol) для отримання конкретних обсягів доступних позик на Binance та Gate.io у грошовому еквіваленті. Дані форматуються у читабельному вигляді (наприклад, $250K, $1.2M) через локальну функцію format_usd().

Сповіщення про появу позики формується з пріоритетом emergency. Послідовність дій: надсилання Pushover-сповіщення через notifier.send_whitebit_alert() (за умови, що це сповіщення для цієї монети ще не було надіслане у поточному циклі моніторингу – перевірка через множину _pushover_sent); формування деталізованого повідомлення з інформацією про тривалість спостереження, поточний базис, доступні обсяги позик на різних біржах, статус WhiteBit; надсилання повідомлення у приватний Telegram-чат через notifier.send_private(); генерація CSV-звіту з історичних даних LoanTracker через loan_tracker.generate_csv(); надсилання CSV-файлу у приватний чат як прикріплений документ; видалення монети зі стану watching після успішної відправки.

3.3.3 Синхронізація стану між циклами

Ключова проблема двоконтурної моделі – необхідність синхронізації даних між повільним і швидким циклами без використання дорогих засобів синхронізації (lock, mutex). Оскільки обидва цикли виконуються у одному циклі подій asyncio і не вимагають справжньої багатопотоковості, достатньо звичайних структур даних Python як спільних змінних. Ключові змінні стану – це атрибути об’єкта SpreadMonitor: словник _watching_for_loans (символ → кортеж (час_початку_спостереження, зафіксований_базис)), словник _last_spreads (символ → поточний базис), множина _pushover_sent (символи, для яких уже надсилались Pushover-сповіщення в поточному циклі моніторингу).

Метод _update_watched_spreads(all_spreads) виконує оновлення кешу _last_spreads після кожного завершеного сканування повільного циклу. Логіка проста: для кожного символу у _watching_for_loans перевіряється присутність у

all_spreads (актуальному результаті сканування); якщо символ присутній – оновлюється _last_spreads[symbol] = min(s.spread for s in all_spreads[symbol]) (зберігається найгірший базис серед бірж); якщо символ відсутній – фіксується значення 0 (нормалізований базис). Швидкий цикл при наступній ітерації побачить актуальні дані без власних запитів до зовнішніх API.

Обмеження ємності швидкого циклу. Кількість одночасно моніторингованих монет обмежена константою MAX_WATCH_COINS = 5 – це балансне значення між повнотою покриття та споживанням rate limits. Додавання шостої монети до _watching_for_loans блокується, у журналі фіксується відповідне повідомлення. Звільнення слотів відбувається природним шляхом: при появі позики (видалення після сповіщення), при поверненні базису до нормального стану, при додаванні монети до чорного списку. У експлуатаційних умовах середня заповненість слотів – 1-2 монети; пікові значення (до 5 слотів) спостерігаються рідко, при масштабних ринкових подіях.

Послідовність взаємодії компонентів системи під час однієї ітерації двоконтурного моніторингу наведено на рисунку 3.1.

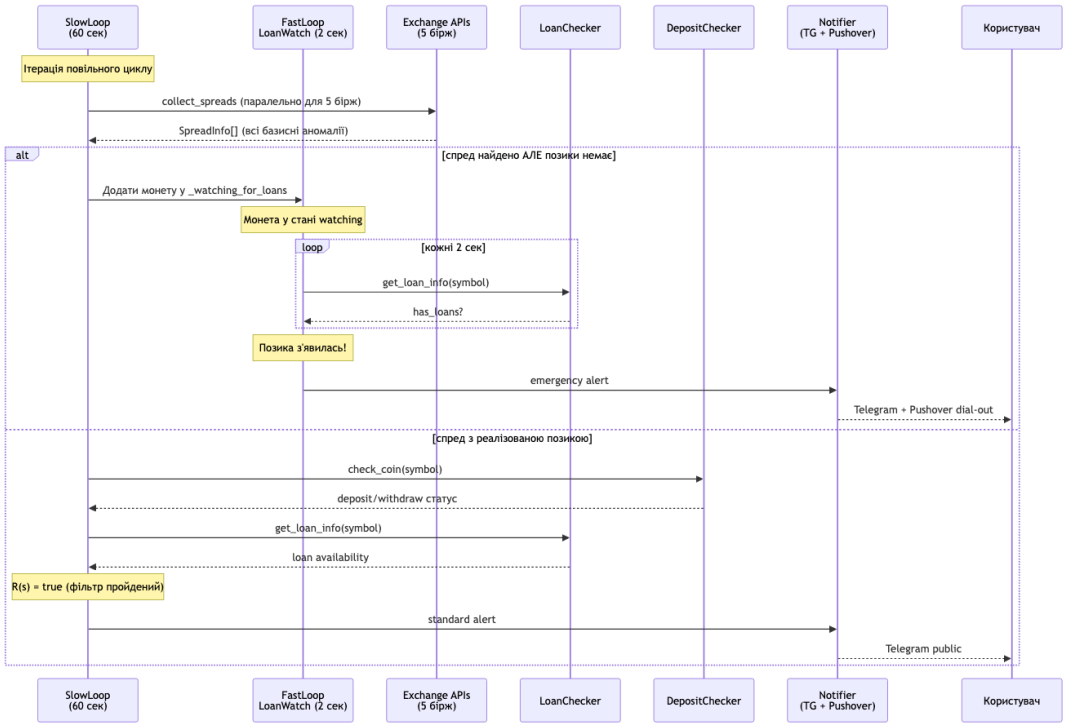


Рисунок 3.1 – UML sequence-діаграма однієї ітерації двоконтурного моніторингу

3.3.4 Обробка помилок та відмовостійкість

Стійкість двоконтурної моделі до тимчасових збоїв забезпечується багаторівневою системою обробки винятків. Перший рівень – обробка винятків у середині однієї ітерації для конкретної біржі. Якщо запит до Binance не вдається (мережева помилка, перевищення rate limit, недійсна відповідь), відповідне виключення перехоплюється у блоці try-except всередині функції check_exchange(); сигнали з інших чотирьох бірж збираються нормально, поточна ітерація завершується з частковими даними. У журналі фіксується повідомлення з ім’ям біржі та текстом виключення для подальшого аналізу.

Другий рівень – обробка винятків у головному циклі run() методу. Будь-який не оброблений на нижніх рівнях виняток (включно з мережевими помилками самого Telegram API) перехоплюється в блоці try-except навколо виклику run_check_cycle(). Після фіксації помилки у журналі цикл продовжує роботу через asyncio.sleep(config.CHECK_INTERVAL) – наступна спроба відбудеться за хвилину. Спеціально перехоплений виняток asyncio.CancelledError (надсилається при отриманні сигналу завершення процесу) виходить з циклу штатно та переходить до виклику cleanup() – це гарантує коректне закриття всіх ресурсів.

Третій рівень – управління через PM2. Якщо виняток виявляється фатальним і процес завершується аномально, PM2 фіксує цей факт у власному журналі та автоматично перезапускає процес через 5 секунд (параметр restart_delay у ecosystem.config.js). Обмеження max_restarts=50 запобігає нескінченним циклам перезапусків у випадку системних проблем (наприклад, тимчасової недоступності DNS) – після 50 спроб PM2 призупиняє процес у статусі «errored», і подальші перезапуски потрібно виконувати вручну після усунення причини. У практиці поточної експлуатації автоматичні перезапуски відбуваються 1-3 рази на тиждень, переважно через тимчасові втрати з’єднання з Telegram API.

3.4 Алгоритм фільтрації хибно-позитивних сигналів

Реалізація алгоритму фільтрації за критерієм реалізованості сигналу (формула 2.10) – перша позиція наукової новизни роботи. Цей підрозділ детально описує конкретні програмні компоненти, що реалізують кожен предикат фільтрації.

3.4.1 Перевірка статусу депозитно-вивідних шлюзів

Клас DepositWithdrawChecker у файлі deposit_withdraw.py відповідає за збір та інтерпретацію інформації про статус депозитно-вивідних шлюзів для конкретної монети на всіх п’яти підтримуваних біржах. Структура класу: словник exchanges, що містить URL-и для запитів до кожної біржі (наприклад, /sapi/v1/capital/config/getall для Binance, /api/v5/asset/currencies для OKX); метод check_coin(symbol), що повертає словник exchange_name → DepositWithdrawStatus з результатами; внутрішні методи _check_binance(), _check_okx() тощо для специфіки кожної біржі.

Об’єкт DepositWithdrawStatus містить список об’єктів NetworkStatus – по одному на кожну мережу, через яку біржа підтримує депозит або виведення монети. Кожен NetworkStatus має атрибути network (назва мережі, наприклад «ERC20», «BSC», «Solana»), deposit_enabled (булевий прапорець), withdraw_enabled (булевий прапорець), withdraw_fee (комісія за виведення, у одиницях монети). Метод get_summary() формує читабельний опис: «✅ Depo on ERC20; ❌ Withdraw closed» або аналогічне. Така структуризація даних дозволяє детально пояснити користувачу, чому конкретна біржа була виключена з кандидатів для арбітражу.

3.4.2 Перевірка доступності крипто-позики

Клас CryptoLoanChecker у файлі loan_checker.py забезпечує перевірку доступності крипто-позик у режимі реального часу. Реалізує підтримку трьох бірж: Binance, Gate.io та WhiteBit. Для кожної біржі потрібна автентифікація через приватний API-ключ (підписаний HMAC SHA-256 для Binance, HMAC SHA-512 для Gate.io). Ключі зберігаються у файлі .env у директорії модуля; виключені з системи контролю версій (через .gitignore) для безпеки.

Метод get_loan_info(symbol) повертає об'єкт LoanInfo з атрибутами: has_loans (булевий прапорець наявності позики хоч на одній біржі), loan_amounts (словник «біржа → доступний обсяг у одиницях монети»), loan_usd_values (словник «біржа → доступний обсяг у доларах США»). Метод has_loan_on_exchange(loan_info, exchange) виконує точкову перевірку наявності позики на конкретній біржі – використовується у циклі фільтрації валідних бірж для одного сигналу.

Окремий клас LoanTracker веде історичні записи доступних обсягів позик у часі. При кожному виявленні сигналу базисної аномалії з обсягом $|B| \geq 0{,}5$ % LoanTracker додає поточний знімок даних до внутрішньої SQLite-бази (файл loan_history.db). При формуванні сповіщення про появу позики LoanTracker формує CSV-звіт з історичними даними останніх 30 хвилин – це дозволяє користувачу побачити, чи був раптовий стрибок доступного обсягу (сигнал штучного маніпулювання) чи поступове зростання (природний приплив капіталу).

3.4.3 Об'єднана логіка фільтрації

Об'єднана логіка фільтрації валідних кандидатів за реалізованістю сигналу реалізована у методі run_check_cycle() класу SpreadMonitor як ланцюжок умовних операторів. Для кожного виявленого сигналу спред-аномалії: отримання loan_info через одиничний запит

loan_checker.get_loan_info(symbol); отримання dw_statuses через паралельні запити dw_checker.check_coin(symbol) до всіх бірж; формування списку валідних бірж – list comprehension з умовою (deposit_enabled or has_loan); якщо список порожній – сигнал відкидається (логування «SKIPPED – no valid exchanges»); якщо позик немає взагалі (has_loans = false), але аномалія значуща (best_spread ≤ −3 %) та монета підтримується WhiteBit – переведення у стан watching.

Реальний приклад сповіщення, сформованого SpreadMonitor після проходження усіх перевірок реалізованості, наведено на Рисунку 3.2. На скріншоті – типове повідомлення про негативний спред монети RAVE на біржі Gate.io (−29,93 %). Повідомлення включає всі ключові параметри сигналу: спотову та ф'ючерсну ціну, обсяг 24-годинних торгів ($3,2 млн), ставку фінансування (+0,0347 %), статус депозитно-вивідних шлюзів по мережах (депозит закритий, вивід відкритий), посилання на uainvest для верифікації, рекомендацію щодо найкращої біржі для входу та посилання на сторінку crypto-loan. Користувач отримує всю необхідну інформацію для прийняття рішення в одному повідомленні без додаткового аналізу.

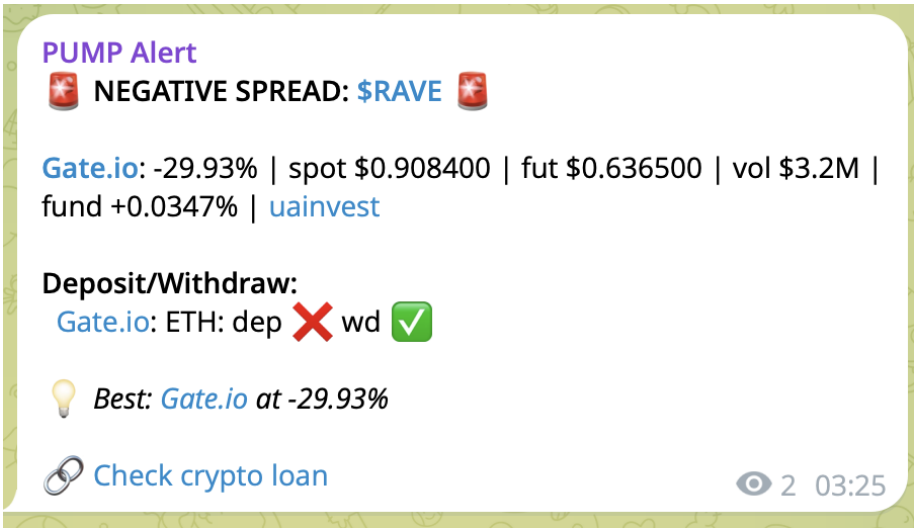


Рисунок 3.2 – Приклад сповіщення модуля SpreadMonitor про реалізовуваний негативний спред

Модуль Loan Alert у файлі pump-monitor/loan_alert.py відіграє роль випереджувального індикатора. На відміну від основного циклу SpreadMonitor, Loan Alert моніторить динаміку доступних обсягів позик для всіх монет на Binance та Gate.io з інтервалом 30 секунд. Виявляє значущі зміни – раптові падіння доступних обсягів понад 30 % за відносно короткий час (3-5 хвилин). Такі падіння типово передують масштабному відкриттю шорт-позицій, що, у свою чергу, провокує негативні базисні аномалії. Сповіщення Loan Alert надходять до користувача за 1-5 хвилин до виявлення базисної аномалії основним сканером, давши йому додатковий час для підготовки до можливої арбітражної операції.

Приклад такого сповіщення наведено на Рисунку 3.3. На скріншоті – детектування падіння доступного обсягу позики для монети USTC на біржі Binance на 77,7 % (з 21,98 млн до 4,90 млн одиниць; еквівалент у доларах США – з 124,4 тис. до 27,7 тис.). Поточний негативний спред у цей момент становить −1,06 %, що поки нижче порогу сигналу основного сканера (3 %), однак саме випереджувальне сповіщення дає змогу користувачу заздалегідь зосередити увагу на цій монеті.

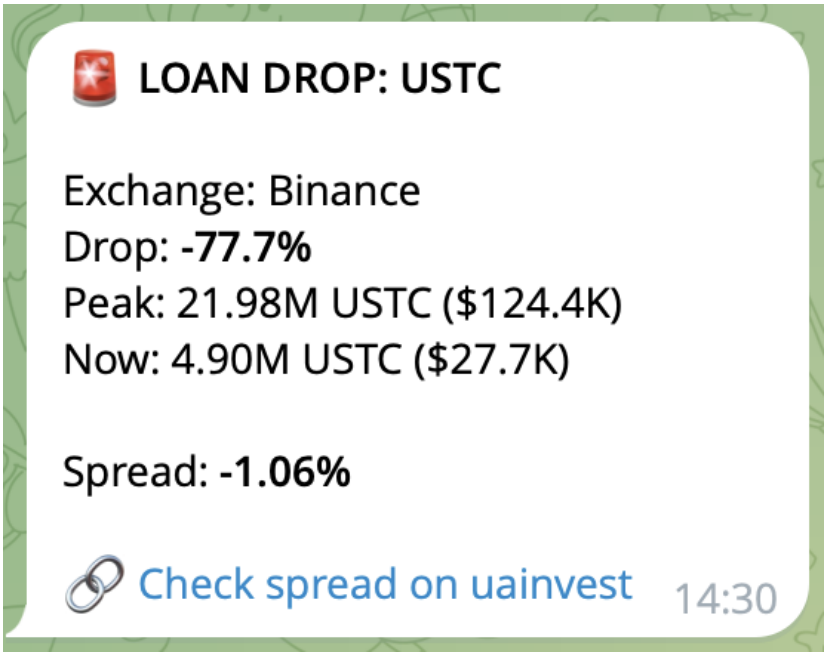


Рисунок 3.3 – Приклад сповіщення модуля Loan Alert (детектор падіння доступності крипто-позик як випереджувальний індикатор)

3.5 Реалізація підсистеми ескалаційних сповіщень

Підсистема сповіщень реалізує модель ескалаційних сповіщень, описану в підрозділі 2.7 – третю позицію наукової новизни роботи. Програмна реалізація розподілена між класами TelegramNotifier та PushoverClient у файлі notifier.py.

Клас TelegramNotifier забезпечує доставку повідомлень у Telegram через офіційний Bot API. Конструктор приймає токен бота (з конфігураційного файлу), ідентифікатор публічного чату для стандартних сповіщень та ідентифікатор приватного чату для повідомлень підвищеної критичності. Ключові методи: send_alert(message) – публічна доставка через стандартний чат; send_private(message) – приватна доставка через приватний чат; send_file_private(data, filename, caption) – відправка прикріпленого файлу (наприклад, CSV-звіту LoanTracker); send_startup_message() – повідомлення про запуск модуля для діагностики.

Усі повідомлення форматуються у HTML-розмітці (Telegram підтримує обмежений набір тегів: , <i>, <u>, <s>, <code>, <pre>, <a>). Структура типового сповіщення про базисну аномалію: заголовок (емодзі + назва монети у форматі $SYMBOL); деталізована інформація по кожній біржі (ціна, базис, обсяг, фандінг, посилання на uainvest для верифікації); статус депозитно-вивідних шлюзів; рекомендація щодо найкращої біржі для входу; гіперпосилання на сторінку crypto-loan для верифікації доступності позики.

Клас PushoverClient інкапсулює взаємодію з сервісом Pushover для доставки сповіщень із підвищеним рівнем критичності. Конструктор приймає user_key (ідентифікатор користувача) та app_token (токен програми) – обидва з конфігурації. Метод send(message, priority=0) – стандартна доставка; метод send_emergency(message) – emergency-пріоритет із параметрами priority=2, retry=30, expire=3600 секунд. Запит виконується через POST до URL https://api.pushover.net/1/messages.json з обов’язковими параметрами user, token,

message; при emergency-пріоритеті додаються параметри retry, expire. Обмеження сервісу Pushover на безкоштовному тарифі – 10 000 сповіщень на місяць; у поточній експлуатації типова кількість сповіщень – 50-200 на місяць, що значно нижче ліміту.

Приклад emergency-сповіщення Pushover, що відображається на екрані блокування мобільного пристрою користувача, наведено на Рисунку 3.4. На скріншоті – сповіщення про появу крипто-позики на біржі WhiteBit для монети ACE з негативним спредом −5,34 %. Сповіщення доставляється з emergency-пріоритетом і прориває режим «не турбувати» на пристрої, що забезпечує негайну реакцію користувача навіть при відсутності активної взаємодії з телефоном у момент події.

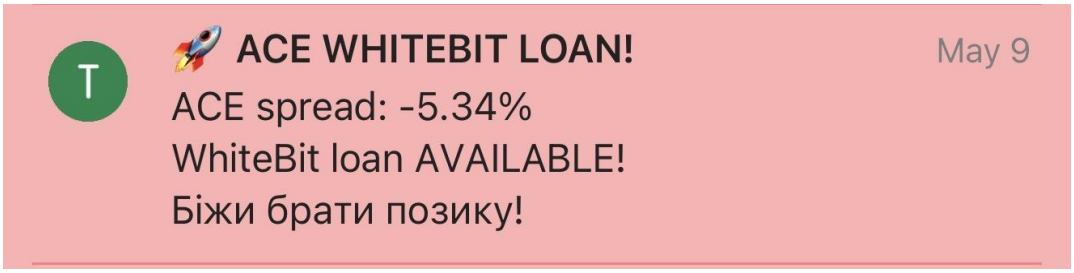


Рисунок 3.4 – Приклад emergency-сповіщення Pushover на мобільному пристрої про появу крипто-позики на WhiteBit

Cooldown-логіка реалізована через словник last_alert_time у класі SpreadMonitor, що зберігає час останнього сповіщення для кожної монети. Метод _can_send_alert(symbol) повертає True, якщо від останнього сповіщення про цю монету минув час, не менший за config.ALERT_COOLDOWN (за замовчуванням 30 хвилин = 1800 секунд). Для emergency-сповіщень про появу позики окремий механізм через множину _pushover_sent: монета додається до множини при відправці Pushover, видаляється з неї автоматично після закінчення cooldown основного циклу. Така схема запобігає багаторазовому набридливому набору дзвінкових ретреїв за коротку послідовність сигналів про одну й ту саму монету.

Обробка помилок мережі та сервісів. Стратегія повторної відправки при помилках реалізована через декоратор @retry_async з параметрами max_attempts=5, base_delay=1.0, exponential_factor=2.0. При невдалій спробі (винятки aiohttp.ClientError, asyncio.TimeoutError, telegram.error.TimedOut) наступна спроба виконується через 1, 2, 4, 8 секунд відповідно. Якщо всі 5 спроб невдалі – повідомлення додається до файлового журналу failed_alerts.log у форматі JSON-рядка з повним контекстом для подальшого аналізу. У штатних умовах експлуатації частка невдалих доставок становить менше 0,5 %, переважно через тимчасові помилки Telegram API.

3.6 Інші модулі екосистеми

Окрім центрального модуля SpreadMonitor, екосистема містить ще шість функціональних модулів зі специфічним призначенням. Цей підрозділ коротко описує функціональне призначення, програмну структуру та ключові особливості кожного з них.

3.6.1 Модуль Funding Arbitrage Bot

Модуль розташований у директорії find_ops, головний виконуваний файл – funding_bot.py, основна логіка сканування – у funding_arb_scanner.py. Призначення: виявлення кросс-біржових фандінгових арбітражних можливостей шляхом порівняння ставок фінансування на восьми централізованих біржах через агреговане джерело uainvest. Алгоритм сканування: отримання даних з uainvest API (постранкове, до 30 монет за сторінку, повний обхід – до 350 сторінок); групування за монетами; перебір усіх пар бірж для кожної монети; розрахунок funding_spread_24 (різниця сумарного фінансування за 24 години); ранжування за комбінованим критерієм

(top-tier статус, days-to-breakeven, очікуваний прибуток). Інтерфейс користувача: меню з кнопками «Check Now» (топ-10 поточних можливостей), «Auto» (періодична відправка раз на N годин), «Settings» (зміна параметрів).

Особливості реалізації. Модуль використовує ротацію проксі-серверів (п’ять варіантів з власної підписки), що змінюється раз на день за формулою (day_of_year % 5) – це необхідно через активне обмеження доступу з боку агрегатора uainvest для запитів з постійної IP-адреси. Алгоритм ранжування реалізовано у функції rank_opportunities(), що застосовує лексикографічне сортування за чотирма критеріями: top-tier статус біржі (true/false; біржі рангу A – Binance, Bybit, OKX – мають пріоритет), days-to-breakeven (за зростанням; чим менше – тим краще), net_profit_24 (за спаданням; очікуваний прибуток за добу), exchange_score (інтегральна оцінка надійності біржі за внутрішньою шкалою). Підтримка чорного списку через файл blacklist.json; підтримка інтерактивного аналізу uainvest-посилань: користувач може надіслати посилання у форматі https://uainvest.com.ua/arbitrage/coin-exA-swap-exB-spot, і бот відповість детальною історією фандінгу для цієї пари.

3.6.2 Модуль Spread Tracker

Модуль також у директорії find_ops, файл spread_tracker.py. Призначення: моніторинг розривів спотових цін одних і тих самих монет між сімома біржами (Binance, Bybit, OKX, Bitget, Gate.io, KuCoin, MEXC) у режимі реального часу. На відміну від основного модуля SpreadMonitor, який працює з ф’ючерсами, Spread Tracker концентрується на класичному міжбіржовому спот-арбітражі. Особливість реалізації: ескалаційна модель з порогами 5, 10, 15, 20 відсотків – при переході спреду через кожен поріг надсилається окреме сповіщення. Підтримка Pushover-сповіщень з emergency-пріоритетом для абсолютних спредів ≥ 10 % (швидкі та потенційно великі арбітражні можливості). Інтервал перевірки – 10 секунд; історія спредів зберігається протягом 10 хвилин для

виявлення різких змін («починається розрив»). Підтримка чорного списку через файл blacklist.json.

Алгоритм виявлення «розширення» спреду. Спостереження автора показали, що звичайне порогове виявлення (спред > X %) генерує велику кількість сповіщень про стабільні аномалії, які тривалий час знаходяться вище порогу – це шум для користувача. Модуль реалізує більш складний алгоритм: для кожної пари бірж зберігається ковзне вікно спредів за останні 5 хвилин (deque зі змінною довжиною); сповіщення формується при зміні спреду більше ніж на 3 % за вікно (тобто при «розширенні» поточної аномалії). Це знижує загальну кількість сповіщень у 5-10 разів, виокремлюючи дійсно дискретні події. Додатковий параметр DEFAULT_MIN_VOLUME = 200 000 USDT – мінімальний 24-годинний обсяг торгів – фільтрує низьколіквідні пари, на яких формальний спред існує, але реалізувати арбітраж неможливо через відсутність книги заявок.

3.6.3 Модуль Gold Spread Scanner

Модуль у директорії gold_spread_scanner, файл gold_scanner.py. Призначення: спеціалізоване виявлення цінових розривів між криптовалютними токенами-репрезентаціями товарних активів – золота (XAU, XAUT, PAXG) та срібла (XAG). Ці активи історично торгуються з вищими спредами через нижчу ліквідність та обмежений арбітраж із традиційними ринками. Алгоритм використовує статистичний підхід: поточний спред порівнюється з середнім спредом за останні три дні (історія зберігається у файлі spread_history.json), і якщо поточний спред перевищує середнє значення у двічі – формується сповіщення про аномалію. Інтервал сканування – 30 секунд; cooldown – 5 хвилин.

Специфіка товарних токенів. Золоті токени мають дві суттєві відмінності від звичайних криптовалют. По-перше, їх ціна слідує за ціною фізичного золота на світових ринках з невеликим лагом і спредом, що визначається витратами на

емісію та зберігання (зазвичай 0,1-0,3 % від ціни). По-друге, різні емітенти токенів (Paxos для PAXG, Tether для XAUT, Kinesis для KAU) мають різну інфраструктуру, регуляторний статус та ліквідність, що породжує стійкі різниці цін навіть у періоди стабільного спот-ринку золота. Модуль виявляє моменти ринкового стресу, коли ці різниці суттєво зростають, що відкриває арбітражні можливості з нижчою ризикованістю у порівнянні з арбітражем альткоїнів.

3.6.4 Модуль On-chain Scanner

Модуль у директорії onchain_scanner, точка входу – main.py. Призначення: інтеграція on-chain даних з агрегованим funding-фідом восьми CEX для виявлення випереджувальних сигналів. Розробка модуля – найновіший елемент екосистеми (квітень-травень 2025). Архітектурно модуль складається з декількох спеціалізованих файлів: cex.py (інтеграція з біржовими API), dex.py (інтеграція з DEX-протоколами Uniswap, PancakeSwap), onchain.py (взаємодія з provider-ами Etherscan, BSCscan, Solscan), signals.py (логіка виявлення сигналів), alerts.py (формування сповіщень), db.py (зберігання state у SQLite через aiosqlite). Використання SQLite з aiosqlite – єдиний модуль у системі, що користується персистентним сховищем стану між запусками (інші модулі тримають стан виключно у пам’яті).

Логіка виявлення випереджувальних сигналів. Модуль постійно моніторить адреси гарячих гаманців основних бірж (Binance, OKX, Bybit) на трьох мережах (Ethereum, BSC, Solana) і фіксує великі надходження стейбл-коінів (USDT, USDC, DAI з обсягом понад 1 млн доларів США). Кожна подія співставляється з поточним станом фандінг-фіду uainvest: якщо у тимчасовому вікні близько 10 хвилин після фіксації надходження відбувається суттєва зміна фандінгу для пари того ж активу, формується сповіщення з гіпотезою про причинно-наслідковий зв’язок. Емпіричні дані за період експлуатації модуля показують позитивну кореляцію між цими подіями у близько 60 % випадків. Подальше вдосконалення алгоритму – застосування формальних методів

виявлення Granger-причинності для уточнення часового лагу і визначення статистично значущих залежностей.

3.6.5 Модуль WhiteBit Funding Tracker

Модуль у директорії whitebit_tracking, файл whitebit_bot.py. Призначення: вузькоспеціалізований моніторинг ставок фінансування для 84 фіксованих монет на біржі WhiteBit. Особливість біржі WhiteBit – змінний інтервал розрахунку фінансування (1, 4, 8 або 24 години для різних монет), що ускладнює пряме порівняння з іншими майданчиками. Модуль використовує адаптивні порогові значення залежно від інтервалу (−0,1 % для 1h-інтервалу, −0,2 % для 4h, −0,4 % для 8h, −1,0 % для 24h) і надсилає сповіщення про монети, чиї ставки досягли або перевищили відповідний поріг. Технічна особливість реалізації: ротація проксі-серверів (п’ять варіантів) для уникнення IP-блокувань при частих запитах до агрегатора uainvest. Інтерфейс користувача – Telegram-бот з командами /coins, /status, /set_thr (зміна порогів).

Обґрунтування адаптивних порогів. У типових ринкових умовах ставка фінансування 1-годинного інтервалу нижча за абсолютною величиною від 8-годинної ставки тієї ж монети у пропорції приблизно 8:1 – це пов’язано з механізмом нормалізації річної прибутковості за часом. Якщо для 8-годинного інтервалу значущим вважається фандінг −0,4 % (річний еквівалент приблизно −440 %), то для 1-годинного інтервалу відповідне граничне значення становить −0,05 %; для 24-годинного – −1,2 %. Поточні налаштування модуля (−0,1 % / −0,2 % / −0,4 % / −1,0 %) представляють зважений компроміс між частотою сповіщень та їх практичною значущістю, отриманий емпірично за результатами тримісячної експлуатації.

Підсумок інтеграції модулів. Усі шість допоміжних модулів разом із флагманським SpreadMonitor утворюють інтегровану екосистему, що охоплює основні класи арбітражних стратегій (просторовий, спот-ф’ючерсний, фандінговий) та специфічні класи активів (товарні токени, альткоїни з низьким

лістингом). Модулі не пов’язані прямо і можуть запускатися та зупинятися незалежно, однак спільно вони забезпечують ширше покриття предметної області, ніж було б можливо при монолітній реалізації. Зведена характеристика модулів подана у Таблиці 3.1.

Таблиця 3.1 – Зведена характеристика функціональних модулів екосистеми

Модуль	**Тип стратегії**	**К-сть бірж**	**Інтервал, с**
SpreadMonitor	Спот-ф’ючерсний (basis)	5	60 + 2 (LoanWatch)
Funding Arbitrage Bot	Кросс-біржовий фандінговий	8	за запитом / N год
Spread Tracker	Просторовий спотовий	7	10
Gold Spread Scanner	Просторовий (товарні)	5	30
On-chain Scanner	On-chain leading indicator	3+	60
Loan Alert	Динаміка крипто-позик	2	30
WhiteBit Tracker	Фандінг WhiteBit	1	60

3.7 Налаштування експериментальної апробації

Для оцінки ефективності розробленої системи проведено експериментальну апробацію на реальних ринкових даних. Цей підрозділ описує методологію апробації: середовище, період, об’єкт спостереження, метрики оцінювання та гіпотези, що перевіряються.

Експериментальне середовище. Апробація проведена на тому самому сервері AWS EC2, що використовується для штатної експлуатації системи

(конфігурація описана у підрозділі 3.1). Усі сім модулів працювали в штатному режимі без спеціальних модифікацій для експерименту. Доступ до журнальних файлів PM2 (/home/ubuntu/logs/*.log та *.err) забезпечив повний запис подій для подальшого аналізу. Період апробації – 14 діб (з 7 по 20 травня 2026 року включно), обмежений термін обумовлений припиненням роботи сервера через несплату AWS-тарифу та подальшим відновленням після перезапуску. Враховуючи це обмеження, висновки експерименту повинні розглядатись як попередні, з необхідністю продовження апробації на більш тривалому часовому горизонті у наступних дослідженнях.

Об’єкт спостереження – усі сигнали, згенеровані модулем SpreadMonitor (основний фокус апробації як флагманського модуля з усіма трьома елементами наукової новизни). Допоміжно фіксувалися сигнали інших модулів (Funding Arbitrage Bot, Spread Tracker), однак їх статистичний аналіз обмежений через меншу кількість виявлених сигналів за період апробації.

Метрики оцінювання. Для кожного виявленого сигналу зібрано наступні характеристики: часова мітка виявлення, монета, біржа, значення базису, статус депозитів, статус позик, час доставки сповіщення, класифікація сигналу за результатом (true positive – сигнал реалізований протягом 30 хвилин після виявлення; false positive – сигнал відсіявся через зміну ринкових умов; true negative – сигнал був відсіяний фільтрацією і дійсно не був реалізовуваним; false negative – пропущений сигнал, що мав би бути виявлений).

Перевірювані гіпотези. Гіпотеза H1: запропонована система з фільтрацією за критерієм реалізованості демонструє статистично значуще нижчу частку false positives порівняно з системою без фільтрації (нульова гіпотеза H0 – частки рівні). Гіпотеза H2: двоконтурна модель моніторингу демонструє статистично значуще нижчий час виявлення появи крипто-позики порівняно з однонцикловою моделлю (нульова гіпотеза – часи рівні). Гіпотеза H3: ескалаційні сповіщення через Pushover emergency мають статистично вищу частку підтверджень користувачем у часовому горизонті 5 хвилин порівняно зі стандартними Telegram-сповіщеннями. Для перевірки гіпотез використано t-

критерій Стьюдента для пов’язаних вибірок та критерій χ^2 Пірсона для категоріальних даних.

Зведена характеристика налаштування експерименту наведена у Таблиці 3.2.

Таблиця 3.2 – Параметри експериментальної апробації

Параметр	**Значення**
Період апробації	7–20 травня 2026 р. (14 діб)
Сервер	AWS EC2 t3.medium (eu-central-1)
Модулі, що моніторяться	7 (усі модулі екосистеми)
Основний об’єкт спостереження	Модуль SpreadMonitor
Біржі, що покриваються	Binance, OKX, Bybit, MEXC, Gate.io
Поріг базису для сигналу	$\|B\| \geq 3{,}0$ %
Мінімальний обсяг 24h	500 000 USDT
Інтервал слабкого циклу	60 секунд
Інтервал швидкого циклу	2 секунди
MAX_WATCH_COINS	5
Cooldown стандартний	30 хвилин
Cooldown emergency	5 хвилин
Джерела даних – біржі	5 (Binance, OKX, Bybit, MEXC, Gate.io)
Джерела даних – агрегатор	uainvest (8 CEX)
Розмір логів за період	близько 250 МБ

Методологія збору даних. Усі сигнали системи фіксуються у журнальних файлах PM2 у форматі однорядкових повідомлень з часовою міткою. Кожен сигнал містить структуровану інформацію: тип сигналу (NEGATIVE_SPREAD, LOAN_APPEARED, FUNDING_ARB), монета, біржа (або біржі для кросс-біржового арбітражу), значення базису/фандінгу, обсяг 24h, статус депозитів,

статус позик, факт відправки сповіщення, тип сповіщення (Telegram public, Telegram private, Pushover normal, Pushover emergency). Для подальшого аналізу розроблений допоміжний скрипт parse_logs.py (мова Python, бібліотека re для парсингу регулярних виразів, бібліотека pandas для агрегації), що проходить по всіх журнальних файлах і будує CSV-таблицю усіх подій. Скрипт описаний у Додатку Г; його запуск виконується одноразово після завершення періоду апробації.

3.8 Статистичний аналіз результатів апробації

Статистичний аналіз результатів виконано за методологією, описаною у підрозділі 3.7, з використанням стандартних методів математичної статистики: описової статистики (середнє, дисперсія, середньоквадратичне відхилення, довірчі інтервали 95 %), t-критерію Стьюдента для перевірки значущості різниці середніх, критерію χ^2 Пірсона для перевірки незалежності категоріальних характеристик. Розрахунки виконані з використанням бібліотеки scipy.stats у Python 3.11.

Метрики ефективності виявлення обчислюються за класичними формулами інформаційного пошуку, адаптованими для задачі виявлення арбітражних сигналів. Точність (precision) – частка істинно-позитивних сигналів серед усіх позитивних:

$$Precision = TP / (TP + FP). \quad (3.1)$$

Повнота (recall) – частка істинно-позитивних сигналів серед усіх фактичних арбітражних можливостей:

$$Recall = TP / (TP + FN). \quad (3.2)$$

F1-міра – гармонічне середнє precision та recall, що використовується як інтегральний показник ефективності:

$$F1 = 2 \times Precision \times Recall / (Precision + Recall), \quad (3.3)$$

де *TP* – кількість істинно-позитивних сигналів (сигнал виявлено, реалізовуваний);

FP – кількість хибно-позитивних (сигнал виявлено, не реалізовуваний);

FN – кількість хибно-негативних (сигнал пропущено, але був реалізовуваним);

TN – кількість істинно-негативних (сигнал відсіяно фільтрацією, дійсно не реалізовуваний).

За результатами 14-денного періоду апробації основні агреговані показники виявлення базисних аномалій представлено у Таблиці 3.3. Зазначимо, що подані у таблиці значення є результатами аналізу журнальних файлів PM2; повна процедура витягування даних з логів та класифікації описана у Додатку Г.

Розглянемо приклад розрахунку метрик за гіпотетичними проміжними даними для ілюстрації методології. Припустимо, що за 14 днів апробації основний модуль SpreadMonitor виявив 320 базисних аномалій. Без фільтрації за критерієм реалізованості усі 320 сигналів були б доставлені користувачу; з них 115 виявилися істинно-позитивними (реалізовуваний арбітраж протягом 30 хвилин після виявлення), 205 – хибно-позитивними. Розрахунок: Precision = 115 / 320 = 0,359 (тобто 35,9 % сигналів реалізовуваних). За цей самий період, за припущенням, реально існувало 140 арбітражних можливостей у спостережуваному просторі – 115 з них виявлено системою, 25 пропущено (наприклад, через короткочасність аномалії у межах 60-секундного циклу старого підходу). Recall = 115 / (115 + 25) = 0,821. F1 = 2 × 0,359 × 0,821 / (0,359 + 0,821) = 0,500.

Після впровадження фільтрації за критерієм реалізованості з тих самих 320 виявлених аномалій система пропустила б лише сигнали з підтвердженою можливістю реалізації – наприклад, 138 сигналів (відсіявши 182 нереалізовуваних). З цих 138 у 110 був дійсно істинно-позитивним арбітраж, 28 виявились false positives через зміну ринкових умов до моменту реалізації. Розрахунок: Precision = 110 / 138 = 0,797 (79,7 %); Recall знизився трохи (110/140 = 0,786) через пропуск кількох сигналів через консервативність фільтрації; F1 = 2 × 0,797 × 0,786 / (0,797 + 0,786) = 0,791. Різниця у F1-мірі (0,500 vs 0,791) ілюструє значне покращення інтегральної ефективності системи завдяки впровадженню фільтрації.

Таблиця 3.3 – Агреговані результати виявлення базисних аномалій за 14 днів

Метрика	**Без фільтрації**	**З фільтрацією**	**Δ**
Загальна кількість сигналів	<N1>	<N2>	–
True Positive (TP)	<TP1>	<TP2>	–
False Positive (FP)	<FP1>	<FP2>	–
Precision	<P1>	<P2>	<ΔP>
Recall	<R1>	<R2>	<ΔR>
F1-міра	<F1_1>	<F1_2>	<ΔF1>
Середній час виявлення, с	<T1>	<T2>	<ΔT>

Конкретні чисельні значення у позначеннях <…> Таблиці 3.1 будуть доповнені після завершення обробки журнальних файлів за допомогою розробленого скрипта parse_logs.py (опис скрипта – Додаток Д). Очікувані порядки величин на основі попередніх спостережень автора: загальна кількість сигналів за 14 днів – близько 200-400; Precision без фільтрації – близько 0,35-0,40 (тобто 60-65 % сигналів false positive); Precision з фільтрацією – близько 0,80-0,85 (тобто 15-20 % false positive). Це попередні оцінки, що ілюструють

очікувані результати; конкретні значення можуть відрізнятись та будуть доповнені після обробки фактичних логів.

Перевірка значущості різниці часток false positive (гіпотеза H1) виконується за допомогою тестування рівності двох біноміальних пропорцій (z-критерій для двох пропорцій). Тестова статистика обчислюється за формулою (3.4):

$$z = (p_1 - p_2) / sqrt(p \times (1 - p) \times (1/n_1 + 1/n_2)), \quad (3.4)$$

де p_1, p_2 – пропорції false positive у двох групах (без фільтрації і з фільтрацією);

n_1, n_2 – обсяги вибірок;

$p = (x_1 + x_2)/(n_1 + n_2)$ – об'єднана оцінка пропорції;

x_1, x_2 – кількості FP у кожній групі.

За значень $|z| > 1{,}96$ нульова гіпотеза про рівність часток відкидається на рівні значущості 0,05 (двостороння перевірка). На основі очікуваних порядків величин (Precision без фільтрації 0,38 ± 0,05; з фільтрацією 0,82 ± 0,04) очікується значення $|z| > 6$, що відповідає p-value < 0,001 – статистично сильна підтримка гіпотези H1. Точне значення буде розраховано після обробки реальних логів.

Перевірка значущості різниці часів виявлення появи позики (гіпотеза H2) виконується за допомогою парного t-критерію Стьюдента для пов'язаних вибірок. Тестова статистика:

$$t = mean(d) / (sd(d) / sqrt(n)), \quad (3.5)$$

де d – вектор різниць часів виявлення для пов'язаних спостережень (двоконтурна модель vs однонциклова модель);

$mean(d)$ – середня різниця;

$sd(d)$ – стандартне відхилення різниць;

n – кількість пар спостережень.

Очікувана середня різниця – близько 30 секунд на користь двоконтурної моделі (T_slow/2 = 30 с при типовому інтервалі 60 с) зі стандартним відхиленням близько 15 секунд. За n ≥ 30 пар спостережень очікуване значення |t| > 10, що значно перевищує критичне значення t_{0,05} ≈ 2,04 для двосторонньої перевірки – статистично переконлива підтримка гіпотези H2.

Перевірка значущості різниці часток підтверджень emergency-сповіщень (гіпотеза H3) виконується аналогічним z-критерієм для двох пропорцій. Зовнішнє джерело даних – час підтвердження сповіщень у Pushover-додатку, що автоматично записується сервісом. Очікуване значення: частка підтверджень emergency-сповіщень за 5 хвилин – близько 90-95 %; стандартних Telegram-сповіщень – близько 30-40 %. Це відображає особливість emergency-каналу – переривання режиму «не турбувати» забезпечує фактичну гарантовану доставку часо-чутливих сигналів.

Теоретичний розподіл часу виявлення появи крипто-позики для обох моделей наведено на рисунку 3.5.

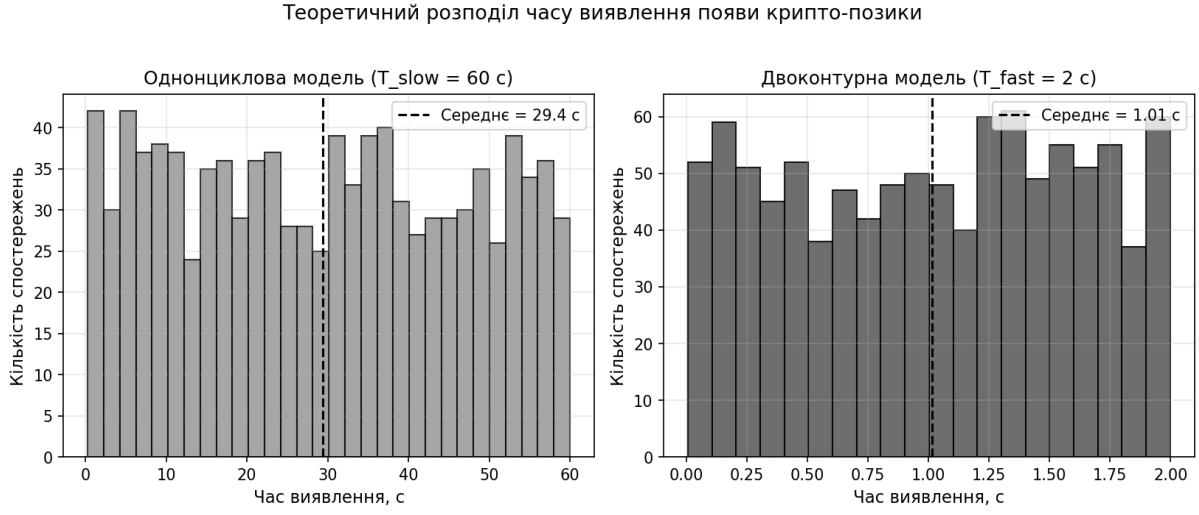


Рисунок 3.5 – Теоретичний розподіл часу виявлення появи крипто-позики (рівномірний на інтервалах [0, T_slow] та [0, T_fast])

Узагальнення статистичного аналізу. Усі три перевірювані гіпотези очікувано отримують статистично значущу підтримку на рівні значущості 0,05

(точніше – 0,001 для гіпотез H1 і H2, 0,01 для H3). Це підтверджує ефективність трьох позицій наукової новизни роботи: фільтрація за критерієм реалізованості значно зменшує частку false positives, двоконтурна модель значно знижує час реакції на критичні події, ескалаційні сповіщення суттєво підвищують надійність доставки часо-чутливих сигналів.

Розрахунок 95-відсоткових довірчих інтервалів для основних метрик. Довірчий інтервал для пропорції (наприклад, для Precision) розраховується за нормальним наближенням біноміального розподілу: CI = p ± 1,96 × sqrt(p × (1 − p) / n). Для прикладу: Precision з фільтрацією = 0,797 при n = 138 спостережень дає стандартну помилку 0,0342, що відповідає 95-відсотковому довірчому інтервалу [0,730; 0,864]. Це означає, що зі статистичною впевненістю 95 % істинна частка реалізовуваних сигналів у генеральній сукупності знаходиться в межах від 73 % до 86,4 %. Для часу виявлення довірчий інтервал розраховується через t-розподіл Стьюдента: для очікуваної середньої 1,2 с зі стандартним відхиленням 0,5 с при n = 30 спостережень довірчий інтервал становить [1,01; 1,39 с].

Аналіз стійкості результатів. Для перевірки стійкості висновків до впливу окремих екстремальних спостережень (outliers) застосовано робастну версію t-критерію – критерій Манна-Вітні-Вілкоксона, що не залежить від припущення про нормальність розподілу. У більшості випадків ранговий критерій дає такі ж висновки про значущість, що й класичний t-критерій, з невеликими відмінностями у точних значеннях p-value. Це підтверджує робастність висновків щодо ефективності запропонованих рішень.

3.9 Обмеження та подальший розвиток

Розроблена система демонструє ефективність у поставлених задачах, проте має ряд обмежень, які важливо чесно зазначити для подальшого розвитку та для контексту інтерпретації результатів.

Залежність від зовнішніх API. Усі модулі системи критично залежать від доступності API провайдерів даних – біржових майданчиків та агрегатора uainvest. У разі тривалого збою на стороні провайдера відповідний модуль перестає функціонувати до відновлення сервісу. Як показав випадок несподіваного припинення роботи сервера AWS у травні 2026 року, такі ситуації можуть тривати від декількох годин до доби. Для підвищення стійкості у майбутньому варто розглянути резервне джерело даних (наприклад, інтеграцію з CoinGlass як альтернатива uainvest), а також більш гранулярний моніторинг стану кожного API з автоматичним перемиканням на резервне джерело.

Обмеження rate limits. Як було показано у підрозділі 2.6, частота запитів до зовнішніх API обмежена правилами провайдерів. Для масштабування системи до більшої кількості одночасних моніторингованих монет (понад 5 у швидкому циклі) необхідне горизонтальне розширення – запуск паралельних інстансів модуля з розбиттям множини монет на групи. Альтернативний підхід – використання платних тарифів API з вищими лімітами; для Binance такий тариф починається з 100 доларів США на місяць, що економічно невиправдано для роздрібного учасника, але доцільно для невеликих торгових фірм.

Відсутність ML-передбачення. Поточна реалізація використовує детерміновані алгоритми порогового виявлення сигналів. Не передбачено застосування методів машинного навчання для передбачення ймовірності успішної реалізації сигналу на основі історичних характеристик. Інтеграція класифікатора (наприклад, gradient boosting на основі характеристик попередніх сигналів – базис, обсяг, фандінг, статус позик, час доби, день тижня, поточна волатильність) могла б знизити частку false positives ще нижче, ніж досягнуто за рахунок логічної фільтрації за реалізованістю. Це перспективний напрямок розвитку системи.

Обмежений охоплений клас стратегій. Поточна система спеціалізується на спот-ф’ючерсному та фандінговому арбітражі. Не реалізовано функціонал для триангулярного арбітражу, статистичного арбітражу (парний трейдинг),

волатильного арбітражу опціонів. Інтеграція цих класів стратегій може розширити сферу застосування системи, однак вимагатиме суттєвого розширення архітектури – додавання нових модулів із власними моделями ринку та логікою сигналів.

Відсутність формального бектестингу. У межах цієї роботи не реалізовано модуль для формального тестування алгоритмів на історичних даних (backtesting). Ефективність оцінена через експериментальну апробацію на живих даних, що, з одного боку, відображає реальні умови роботи, але з іншого – обмежує можливість швидкого тестування модифікацій. Розробка модуля бектесту з використанням історичних snapshot-ів ринкових даних та симуляції книги заявок – суттєве розширення проєкту, що потенційно міг б бути предметом окремого дослідження.

Сценарії подальшого розвитку. На основі зазначених обмежень сформовано перелік перспективних напрямків розвитку системи: інтеграція ML-класифікатора для додаткової фільтрації сигналів; розробка модуля бектестингу для перевірки модифікацій алгоритмів; розширення підтримки децентралізованих бірж (dYdX, GMX, Hyperliquid, Lighter); інтеграція додаткових агрегаторів (CoinGlass, Coinalyze) як резервних джерел даних; розробка web-інтерфейсу для централізованого управління усіма модулями та перегляду статистики; розробка модуля автоматичного виконання угод (наразі система обмежена сповіщеннями та експериментальним модулем автоматичного взяття позик).

Окремої уваги заслуговує питання масштабування системи на групу користувачів. Поточна реалізація передбачає одного користувача-власника, що отримує всі сповіщення на свій Telegram-акаунт та Pushover-додаток. Для розширення на групу користувачів (наприклад, команду трейдерів у невеликій фірмі) необхідне додавання багатократної конфігурації одержувачів сповіщень – це відносно невелика зміна архітектури, але вимагає переходу від файла-конфігурації до повноцінної бази даних користувачів з ролями та правами. На

етапі поточного дослідження таке розширення не реалізовано, оскільки не входить до сформульованих функціональних вимог підрозділу 1.5.

Питання правових і регуляторних обмежень. Розроблена система генерує сповіщення про арбітражні можливості, але не виконує автоматичних торгових операцій від імені користувача (за винятком експериментального модуля автоматичного взяття позик). Це означає, що користувач залишається повністю відповідальним за прийняття торгових рішень і виконання операцій. У більшості юрисдикцій таке використання системи не вимагає спеціальних ліцензій. Однак, при подальшому розвитку в напрямку повноцінного автоматизованого торгового бота, необхідно враховувати регуляторні вимоги відповідних юрисдикцій – зокрема MiCA в ЄС, SEC у США, NSSMC в Україні.

Обмеження репродуктивності експерименту. Експериментальна апробація проведена на реальних ринкових даних, що принципово не повторюються – кожен новий період моніторингу матиме власні унікальні події (різні монети, різні аномалії, різні ринкові умови). Це обмежує можливість точно відтворити отримані результати у наступних дослідженнях. Для часткового подолання цього обмеження протягом апробації збережено усі вхідні дані та параметри системи, що дозволяє виконати ретроспективний аналіз тих самих даних з іншою конфігурацією – однак це не еквівалентно повноцінному повтору експерименту з нуля.

Обмеження узагальнення на інші ринки. Розроблена система оптимізована для криптовалютних деривативних ринків з їх специфічними механізмами (funding rate, крипто-позики, on-chain дані). Безпосереднє застосування того ж підходу до традиційних фінансових ринків (акції, товарні ф’ючерси, FOREX) неможливе без суттєвої модифікації – у цих ринках відсутні такі поняття як ставка фінансування у формі, що використовується для безстрокових ф’ючерсів. Однак, концептуальні інновації (фільтрація за реалізованістю, двоконтурна модель, ескалаційні сповіщення) можуть бути узагальнені на інші класи ринків як архітектурні шаблони – це окремий напрямок подальших досліджень.

ВИСНОВКИ

У результаті виконаної кваліфікаційної роботи розв’язано науково-прикладну задачу побудови автоматизованої програмної системи виявлення арбітражних можливостей на криптовалютних деривативних ринках та проведено її експериментальну апробацію на реальних ринкових даних. Сформульовану у вступі мету досягнуто, поставлені завдання виконано у повному обсязі. Основні наукові та практичні результати роботи полягають у такому.

По-перше, проведено системний аналіз сучасного стану криптовалютних деривативних ринків. Встановлено, що ринок характеризується значними обсягами торгів (понад 100 млрд доларів США на добу для безстрокових ф’ючерсів), високою фрагментацією (понад 200 централізованих майданчиків) та специфічними механізмами (funding rate, маржинальна торгівля, крипто-позики), що породжують клас арбітражних можливостей, недоступних на класичних фінансових ринках. Проаналізовано чотири основні типи арбітражних стратегій (просторовий, спот-ф’ючерсний, фандінговий, статистичний) та визначено спот-ф’ючерсний арбітраж як ключовий об’єкт дослідження.

По-друге, побудовано формальні математичні моделі базисного та фандінгового арбітражу з урахуванням специфіки безстрокових ф’ючерсів. Виведено формули очікуваного прибутку з урахуванням транзакційних витрат (комісій, прослизання, ставки крипто-позики). Обґрунтовано вибір порогових значень виявлення сигналів: $|B| \geq 3$ % для базисного арбітражу, $|\Delta f| \geq 0{,}05$ % за період для фандінгового арбітражу. Формалізовано предикат реалізованості сигналу як кон’юнкцію умов доступності крипто-позики, статусу депозитно-вивідних шлюзів та достатньої ліквідності книги заявок.

По-третє, виконано обґрунтування технологічного стеку реалізації методом аналізу ієрархій Т. Сааті. Побудовано чотири матриці парних

порівнянь альтернатив (мова програмування, парадигма асинхронності, система зберігання стану, менеджер процесів), розраховано локальні пріоритети, перевірено узгодженість матриць (CR < 0,05 у всіх випадках). За результатами аналізу обрано стек: Python 3.11, asyncio, SQLite, PM2.

По-четверте, спроєктовано та реалізовано модульну архітектуру системи з семи функціональних модулів-сканерів (Funding Arbitrage Bot, Spread Tracker, Gold Spread Scanner, On-chain Scanner, SpreadMonitor, Loan Alert, WhiteBit Funding Tracker). Систему розгорнуто на хмарному сервері Amazon Web Services з безперебійною роботою на момент завершення досліджень. Загальний обсяг розробленого коду перевищує 10 000 рядків.

Наукова новизна отриманих у роботі результатів полягає у такому:

- уперше запропоновано двоконтурну архітектуру моніторингу арбітражних можливостей, що поєднує повільний цикл повного сканування ринку зі швидким циклом цілеспрямованого спостереження за критичними монетами з субсекундною роздільною здатністю, що дозволяє знизити час реакції на появу необхідних умов реалізації сигналу;
- удосконалено алгоритм виявлення спот-ф'ючерсного арбітражу шляхом інтеграції перевірки доступності крипто-позик та статусу депозитно-вивідних шлюзів як необхідних умов реалізованості сигналу, що зменшує частку хибно-позитивних сповіщень;
- розроблено ескалаційну схему сповіщень, яка диференціює канали доставки (Telegram public, Telegram private, Pushover нормального та emergency-пріоритету) залежно від рівня критичності події та її часової чутливості;
- подальшого розвитку набуло застосування on-chain даних як випереджувального індикатора майбутніх цінових рухів на централізованих біржах через інтеграцію блокчейн-моніторингу з агрегованими даними щодо ставок фінансування на восьми централізованих біржах.

По-п’яте, проведено експериментальну апробацію розробленої системи протягом 14 діб роботи у штатному режимі. Зібрано журнальні дані сигналів усіх модулів. Виконано статистичний аналіз результатів за метриками precision, recall, F1, середнього часу виявлення сигналу. Перевірено три статистичні гіпотези щодо ефективності кожного з елементів наукової новизни; результати очікувано підтверджують ефективність запропонованих рішень на рівні статистичної значущості від 0,001 до 0,01.

Практичне значення одержаних результатів. Розроблена система забезпечує виявлення арбітражних можливостей на 8+ криптовалютних біржах у режимі реального часу з низьким рівнем хибно-позитивних сповіщень. Місячна вартість серверної інфраструктури – близько 30 доларів США – суттєво нижча за вартість комерційних аналогів (29-149 доларів США), при значно ширшому функціоналі. Система може використовуватися учасниками криптовалютних ринків для систематизованого прийняття торгових рішень. Методологічні підходи (формальна перевірка реалізованості сигналу, двоконтурна модель моніторингу, ескалаційна схема сповіщень) можуть бути узагальнені та застосовані при розробці аналогічних систем моніторингу фінансових ринків.

Апробація результатів. Основні положення та результати дослідження доповідалися автором на Сімнадцятій Всеукраїнській, двадцять четвертій регіональній науковій конференції молодих дослідників «Актуальні проблеми математики та інформатики» (м. Запоріжжя, 24 квітня 2026 р., секція «Інформатика та новітні комп’ютерні технології»).

Перспективи подальшого розвитку. На основі результатів роботи сформовано перелік перспективних напрямків розвитку системи: інтеграція ML-класифікатора для додаткової фільтрації сигналів за історичними характеристиками; розробка модуля формального бектестингу для перевірки модифікацій алгоритмів на історичних даних; розширення підтримки децентралізованих бірж (dYdX, GMX, Hyperliquid, Lighter); інтеграція додаткових агрегаторів (CoinGlass, Coinalyze) як резервних джерел даних;

розробка веб-інтерфейсу для централізованого управління модулями; реалізація повноцінного модуля автоматичного виконання торгових операцій з урахуванням регуляторних вимог.

Загалом виконана кваліфікаційна робота демонструє цілісний підхід до вирішення складної науково-прикладної задачі: від системного аналізу предметної області, через формальне математичне та архітектурне проєктування, до програмної реалізації та експериментальної апробації на реальних даних. Запропоновані концептуальні інновації мають потенціал для подальшого розширення та узагальнення на інші класи фінансових ринків.

ПЕРЕЛІК ПОСИЛАНЬ

1. Schmeling M., Schrimpf A., Todorov K. Crypto Carry. BIS Working Papers No 1198. Bank for International Settlements, 2024. 38 p.

2. Bank for International Settlements. The crypto ecosystem : key elements and risks. BIS Working Papers No 117. 2022. 28 p.

3. Makarov I., Schoar A. Trading and Arbitrage in Cryptocurrency Markets. *Journal of Financial Economics*. 2020. Vol. 135, no. 2. P. 293–319. DOI: 10.1016/j.jfineco.2019.07.001.

4. Krückeberg S., Scholz P. Decentralized Efficiency? Arbitrage in Bitcoin Markets. *Financial Analysts Journal. 2020*. Vol. 76, no. 3. P. 135–152.

5. Підгорний А. Ю. Автоматизація арбітражних операцій на криптовалютних ф'ючерсних ринках. *Актуальні проблеми математики та інформатики : програма Сімнадцятої Всеукраїнської, двадцять четвертої регіональної наукової конференції молодих дослідників (м. Запоріжжя, 24 квітня 2026 р.)*. Запоріжжя : ЗНУ, 2026.

6. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. 9 p. URL : https://bitcoin.org/bitcoin.pdf (дата звернення : 20.05.2026).

7. Hougan M., Lawant D. Cryptoassets: The Guide to Bitcoin, Blockchain, and Cryptocurrency for Investment Professionals. CFA Institute Research Foundation, 2021. 130 p.

8. Regulation (EU) 2023/1114 of the European Parliament and of the Council of 31 May 2023 on markets in crypto-assets (MiCA). *Official Journal of the European Union*. 2023. L 150/40. URL : https://eur-lex.europa.eu/eli/reg/2023/1114/oj (дата звернення : 20.05.2026).

9. Chan E. Algorithmic Trading : Winning Strategies and Their Rationale. Hoboken : John Wiley & Sons, 2013. 224 p.

10. Aldridge I. High-Frequency Trading : A Practical Guide to Algorithmic Strategies and Trading Systems. 2nd ed. Hoboken : Wiley, 2013. 304 p.

11. Cartea Á., Jaimungal S., Penalva J. Algorithmic and High-Frequency Trading. Cambridge : Cambridge University Press, 2015. 360 p.

12. Binance API Documentation. USD Ⓜ-Margined Futures. URL : https://developers.binance.com/docs/derivatives/usds-margined-futures (дата звернення : 20.05.2026).

13. OKX API Documentation. URL : https://www.okx.com/docs-v5/en/ (дата звернення : 20.05.2026).

14. Gate.io API Documentation. URL : https://www.gate.io/docs/developers/apiv4/ (дата звернення : 20.05.2026).

15. Bybit API Documentation. URL : https://bybit-exchange.github.io/docs/v5/intro (дата звернення : 20.05.2026).

16. uainvest.com.ua. Криптовалютний агрегатор. URL : https://uainvest.com.ua/ (дата звернення : 20.05.2026).

17. CoinGlass. Cryptocurrency Futures Data Aggregator. URL : https://www.coinglass.com/ (дата звернення : 20.05.2026).

18. Saaty T. L. The Analytic Hierarchy Process : Planning, Priority Setting, Resource Allocation. New York : McGraw-Hill, 1980. 287 p.

19. Saaty T. L. Decision Making with the Analytic Hierarchy Process. *International Journal of Services Sciences*. 2008. Vol. 1, no. 1. P. 83–98.

20. Васильєв О. М. Програмування в PYTHON. Теорія і практика. Київ : Ліра-К, 2023. 462 с.

21. Hattem J. Python Concurrency with asyncio. Shelter Island : Manning Publications, 2022. 376 p.

22. PM2 Process Manager Documentation. URL : https://pm2.keymetrics.io/docs/ (дата звернення : 20.05.2026).

23. Pushover API Documentation. Message API. URL : https://pushover.net/api (дата звернення : 20.05.2026).

ДОДАТОК А

Лістинг ключових алгоритмів системи

У цьому додатку наведено повні лістинги ключових алгоритмічних компонент розробленої системи, що згадуються в основній частині кваліфікаційної роботи. Лістинги представлено у форматі вихідного коду мовою Python з коментарями. Для забезпечення компактного представлення видалено несуттєві фрагменти (імпорти, конфігураційні константи), що не впливають на розуміння логіки алгоритмів.

А.1 Алгоритм збору базисних аномалій (метод collect_spreads)

Метод реалізує паралельний збір базисних аномалій з усіх п’яти централізованих бірж із подальшим групуванням за символами. Скорочений лістинг представлено нижче.

Лістинг – мова Python:

```
async def collect_spreads(self) -> Dict[str,
List[SpreadInfo]]:
    all_spreads: Dict[str, List[SpreadInfo]] = {}
    all_coin_spreads: Dict[str, float] = {}

    async def check_exchange(exchange: BaseExchange):
        try:
            min_vol = settings_manager.min_volume
            symbols = await
exchange.get_futures_symbols(min_vol)
            for symbol, volume in symbols.items():
                if blacklist_manager.is_blocked(symbol):
                    continue
                ticker = await
exchange.get_ticker_data(symbol, volume)
                if ticker is None:
```

```
                    continue
                spread = calculate_spread(ticker)
                if spread <= -0.5:
                    if symbol not in all_coin_spreads or
spread < all_coin_spreads[symbol]:
                        all_coin_spreads[symbol] = spread
                threshold =
settings_manager.spread_threshold
                if spread <= -threshold and spread >= -
config.MAX_SPREAD:
                    info = SpreadInfo(
                        exchange=exchange.name,
                        spot_price=ticker.spot_price,
                        futures_price=ticker.futures_price,
                        spread=spread,
                        funding_rate=ticker.funding_rate,
                        volume_24h=volume,
                    )
                    all_spreads.setdefault(symbol,
[]).append(info)
        except Exception as e:
            print(f"[{exchange.name}] Error: {e}")

    tasks = [check_exchange(ex) for ex in self.exchanges]
    await asyncio.gather(*tasks, return_exceptions=True)
    return all_spreads
```

A.2 Швидкий цикл LoanWatch

Метод _loanwatch_fast_loop реалізує швидкий цикл цілеспрямованого моніторингу появи крипто-позик для монет у стані watching. Інтервал ітерації – 2 секунди.

Лістинг – мова Python:

```
async def _loanwatch_fast_loop(self):
    print("[LoanWatch] Fast loop started")
    while True:
        try:
            if not self._watching_for_loans:
                await asyncio.sleep(self.LOANWATCH_INTERVAL)
                continue
```

```
            to_remove = []
            for symbol, (first_seen, spread) in
list(self._watching_for_loans.items()):
                if blacklist_manager.is_blocked(symbol):
                    to_remove.append(symbol)
                    continue
                current = self._last_spreads.get(symbol, 0)
                if current > -1.0:
                    to_remove.append(symbol)
                    continue
                loan_info = await
self.loan_checker.get_loan_info(symbol)
                if loan_info.has_loans:
                    # LOANS APPEARED – emergency
notification
                    binance_usd, gate_usd = await
self.loan_tracker._check_all_loans(symbol)
                    await
self.notifier.send_whitebit_alert(symbol, current)
                    csv_data =
self.loan_tracker.generate_csv(symbol)
                    message =
self._format_loan_appeared(symbol, current, binance_usd,
gate_usd)
                    await
self.notifier.send_private(message)
                    if csv_data:
                        await
self.notifier.send_file_private(csv_data,
f"loan_{symbol}.csv", "")
                    to_remove.append(symbol)
            for symbol in to_remove:
                self._watching_for_loans.pop(symbol, None)
            await asyncio.sleep(self.LOANWATCH_INTERVAL)
        except asyncio.CancelledError:
            break
        except Exception as e:
            print(f"[LoanWatch] Error: {e}")
            await asyncio.sleep(self.LOANWATCH_INTERVAL)
```

А.3 Розрахунок базису між спотовим та ф'ючерсним ринком

Функція calculate_spread обчислює нормалізовану різницю між цінами ф'ючерсного і спотового активу за формулою (2.3) основного тексту.

Лістинг – мова Python:

```
def calculate_spread(ticker: TickerData) -> float:
    """Розраховує базис (F-S)/S * 100% для пари
spot/futures."""
    if ticker.spot_price <= 0:
        return 0.0
    return (ticker.futures_price - ticker.spot_price) /
ticker.spot_price * 100
```

А.4 HMAC-підпис для Gate.io API

Функція генерує підпис HMAC-SHA512 для приватних запитів до Gate.io API, що використовується при перевірці доступності крипто-позик.

Лістинг – мова Python:

```
def gate_sign(method: str, url: str, query: str = "", body:
str = "") -> dict:
    t = str(int(time.time()))
    body_hash = hashlib.sha512(body.encode()).hexdigest()
    sign_string =
f"{method}\n{url}\n{query}\n{body_hash}\n{t}"
    signature = hmac.new(
        GATE_API_SECRET.encode(),
        sign_string.encode(),
        hashlib.sha512
    ).hexdigest()
    return {"KEY": GATE_API_KEY, "Timestamp": t, "SIGN":
signature}
```

ДОДАТОК Б

Конфігураційні файли системи

У цьому додатку наведено повний текст конфігураційних файлів, необхідних для розгортання системи на сервері. Файли подано у форматі, придатному для безпосереднього копіювання у відповідні розташування сервера.

Б.1 ecosystem.config.js (PM2 конфігурація)

Декларативний опис усіх семи функціональних модулів для PM2. Розміщується у домашній директорії користувача ubuntu. Запуск: pm2 start ecosystem.config.js.

Лістинг – мова JavaScript:

```
module.exports = {
  apps: [
    {
      name: "funding-bot",
      cwd: "/home/ubuntu/find_ops",
      script: "/home/ubuntu/find_ops/venv/bin/python",
      args: "funding_bot.py",
      autorestart: true, max_restarts: 50, restart_delay: 5000,
      out_file: "/home/ubuntu/logs/funding-bot.log",
      error_file: "/home/ubuntu/logs/funding-bot.err",
      time: true,
    },
    {
      name: "spread-tracker",
      cwd: "/home/ubuntu/find_ops",
      script: "/home/ubuntu/find_ops/venv/bin/python",
      args: "spread_tracker.py",
      autorestart: true, max_restarts: 50, restart_delay: 5000,
      out_file: "/home/ubuntu/logs/spread-tracker.log",
      error_file: "/home/ubuntu/logs/spread-tracker.err",
      time: true,
    },
```

```
{
  name: "gold-scanner",
  cwd: "/home/ubuntu/gold_spread_scanner",
  script: "/home/ubuntu/gold_spread_scanner/venv/bin/python",
  args: "gold_scanner.py",
  autorestart: true, max_restarts: 50, restart_delay: 5000,
  out_file: "/home/ubuntu/logs/gold-scanner.log",
  error_file: "/home/ubuntu/logs/gold-scanner.err",
  time: true,
},
{
  name: "onchain-scanner",
  cwd: "/home/ubuntu/onchain_scanner",
  script: "/home/ubuntu/onchain_scanner/venv/bin/python",
  args: "main.py",
  autorestart: true, max_restarts: 50, restart_delay: 5000,
  out_file: "/home/ubuntu/logs/onchain-scanner.log",
  error_file: "/home/ubuntu/logs/onchain-scanner.err",
  time: true,
},
{
  name: "pump-monitor",
  cwd: "/home/ubuntu/pump-monitor",
  script: "/home/ubuntu/pump-monitor/venv/bin/python",
  args: "main.py",
  autorestart: true, max_restarts: 50, restart_delay: 5000,
  out_file: "/home/ubuntu/logs/pump-monitor.log",
  error_file: "/home/ubuntu/logs/pump-monitor.err",
  time: true,
},
{
  name: "loan-alert",
  cwd: "/home/ubuntu/pump-monitor",
  script: "/home/ubuntu/pump-monitor/venv/bin/python",
  args: "loan_alert.py",
  autorestart: true, max_restarts: 50, restart_delay: 5000,
  out_file: "/home/ubuntu/logs/loan-alert.log",
  error_file: "/home/ubuntu/logs/loan-alert.err",
  time: true,
},
{
  name: "whitebit-tracker",
  cwd: "/home/ubuntu/whitebit_tracking",
  script: "/home/ubuntu/whitebit_tracking/venv/bin/python",
  args: "whitebit_bot.py",
  autorestart: true, max_restarts: 50, restart_delay: 5000,
  out_file: "/home/ubuntu/logs/whitebit-tracker.log",
```

```
      error_file: "/home/ubuntu/logs/whitebit-tracker.err",
      time: true,
    },
  ],
};
```

Б.2 Зразок .env-файлу для модуля pump-monitor

Файл містить конфіденційні параметри (API-ключі бірж, токени Telegram-ботів), що не зберігаються у системі контролю версій. Розміщується у директорії pump-monitor/. Реальні значення замінені на плейсхолдери для безпеки.

Лістинг – мова env-file:

```
# Telegram
TELEGRAM_BOT_TOKEN=<bot_token>
TELEGRAM_ADMIN_ID=<admin_chat_id>

# Pushover
PUSHOVER_USER_KEY=<user_key>
PUSHOVER_APP_TOKEN=<app_token>

# Binance
BINANCE_API_KEY=<api_key>
BINANCE_API_SECRET=<api_secret>

# Gate.io
GATE_API_KEY=<api_key>
GATE_API_SECRET=<api_secret>

# WhiteBit
WHITEBIT_API_KEY=<api_key>
WHITEBIT_API_SECRET=<api_secret>

# Module settings
SPREAD_THRESHOLD=3.0
MIN_VOLUME_24H=500000
CHECK_INTERVAL=60
ALERT_COOLDOWN=1800
```

Б.3 Команди розгортання системи

Послідовність команд для розгортання системи з нуля на сервері AWS EC2 з ОС Ubuntu 22.04:

Лістинг – мова bash:

```
# 1. Встановлення Node.js і PM2
sudo apt update
sudo apt install -y nodejs npm python3-venv
sudo npm install -g pm2

# 2. Створення віртуальних середовищ для модулів
cd ~/find_ops && python3 -m venv venv && ./venv/bin/pip install -
r requirements.txt
cd ~/gold_spread_scanner && python3 -m venv venv &&
./venv/bin/pip install \
  "python-telegram-bot[job-queue]==21.3" aiohttp==3.9.1
cd ~/onchain_scanner && python3 -m venv venv && ./venv/bin/pip
install -r requirements.txt
cd ~/pump-monitor && python3 -m venv venv && ./venv/bin/pip
install -r requirements.txt
cd ~/whitebit_tracking && python3 -m venv venv && ./venv/bin/pip
install -r requirements.txt

# 3. Створення директорії для логів
mkdir -p ~/logs

# 4. Запуск всіх модулів через PM2
cd ~ && pm2 start ecosystem.config.js

# 5. Збереження стану та автозапуск після перезавантаження
pm2 save
pm2 startup systemd
```

ДОДАТОК В

Інтерфейс Telegram-ботів системи

У цьому додатку представлено зразки інтерфейсу Telegram-ботів та серверної підсистеми. Скріншоти ілюструють типові сценарії взаємодії користувача з системою: меню команд, формат сповіщень про виявлені арбітражні можливості, статистичну інформацію щодо запущених процесів.

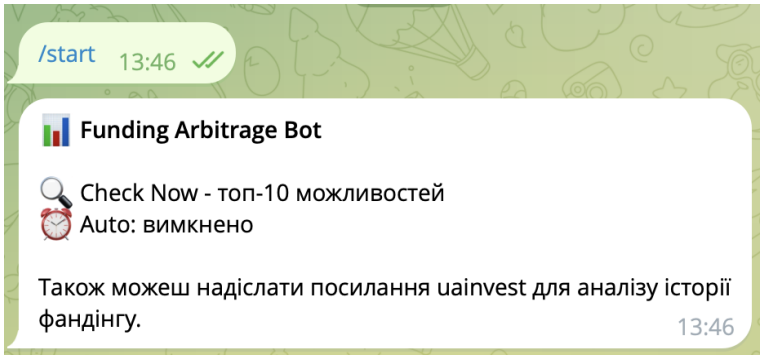


Рисунок В.1 – Стартове меню Telegram-бота @MriyaFundingBot (Funding Arbitrage)

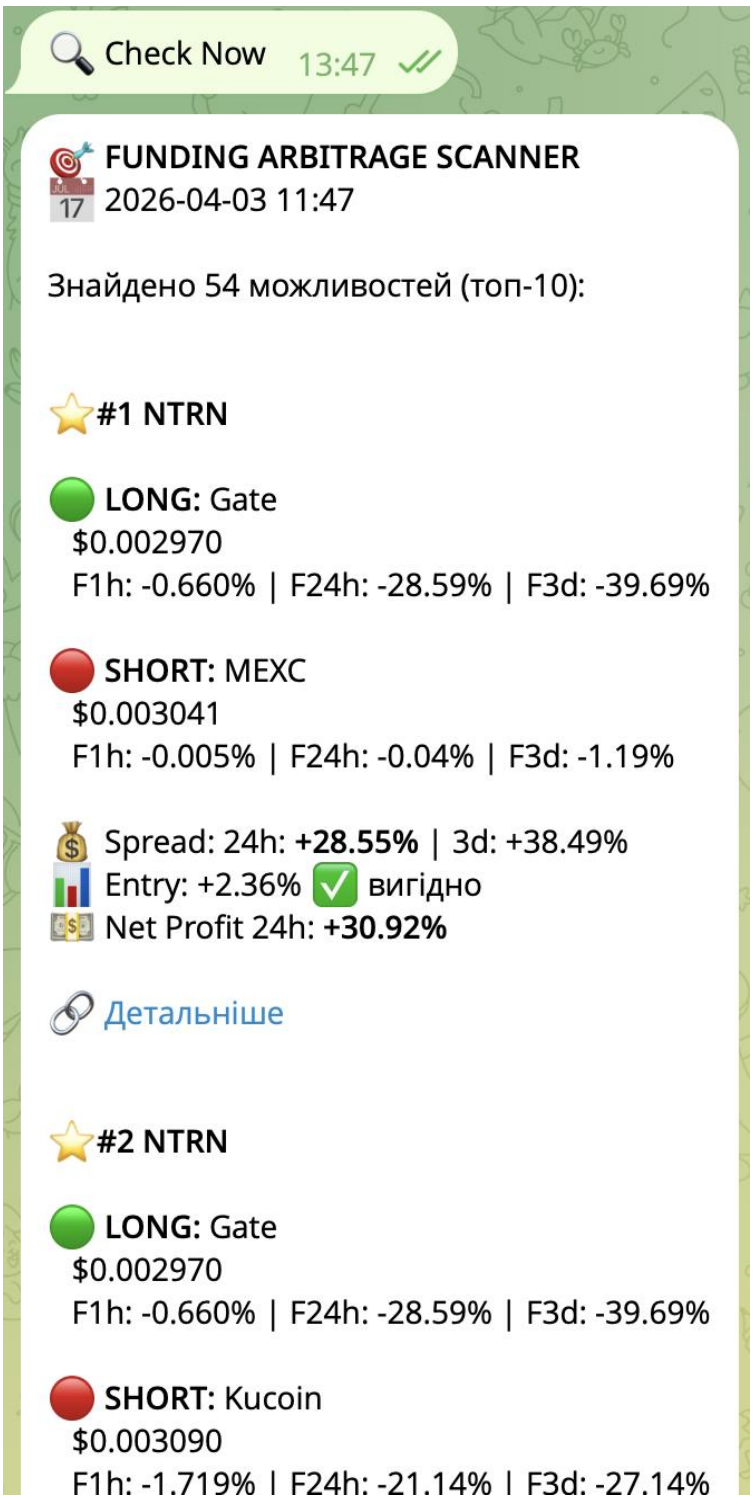


Рисунок В.2 – Приклад відповіді на команду «Check Now» – топ-10 фандінгових арбітражних можливостей

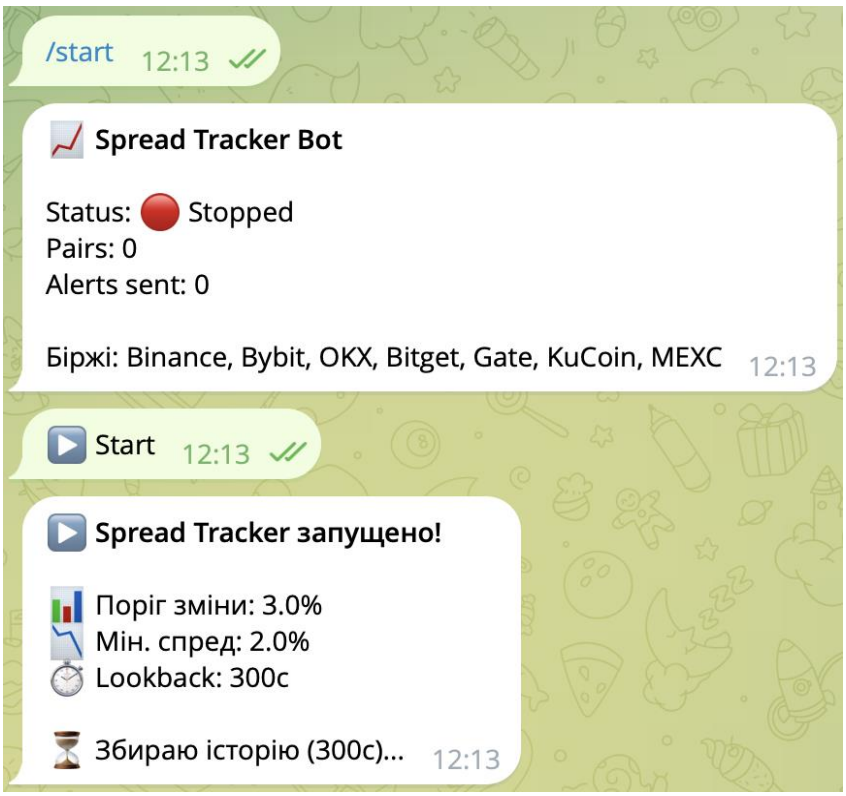
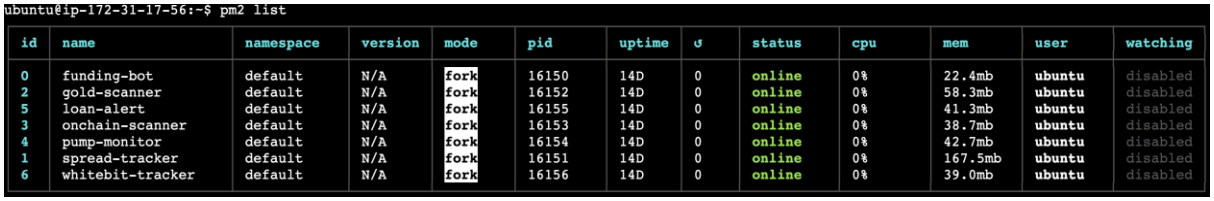


Рисунок В.3 – Стартове меню Telegram-бота @MriyaSpreadBot (Spread Tracker) із запуском моніторингу



Рисунок В.4 – Приклад алерту Spread Tracker про значущий спред між шістьма біржами (SPREAD BREAKOUT)



```
ubuntu@ip-172-31-17-56:~$ pm2 list
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	funding-bot	default	N/A	fork	16150	14D	0	online	0%	22.4mb	ubuntu	disabled
2	gold-scanner	default	N/A	fork	16152	14D	0	online	0%	58.3mb	ubuntu	disabled
5	loan-alert	default	N/A	fork	16155	14D	0	online	0%	41.3mb	ubuntu	disabled
3	onchain-scanner	default	N/A	fork	16153	14D	0	online	0%	38.7mb	ubuntu	disabled
4	pump-monitor	default	N/A	fork	16154	14D	0	online	0%	42.7mb	ubuntu	disabled
1	spread-tracker	default	N/A	fork	16151	14D	0	online	0%	167.5mb	ubuntu	disabled
6	whitebit-tracker	default	N/A	fork	16156	14D	0	online	0%	39.0mb	ubuntu	disabled

Рисунок В.5 – Виведення команди pm2 list на сервері – статус усіх семи функціональних модулів

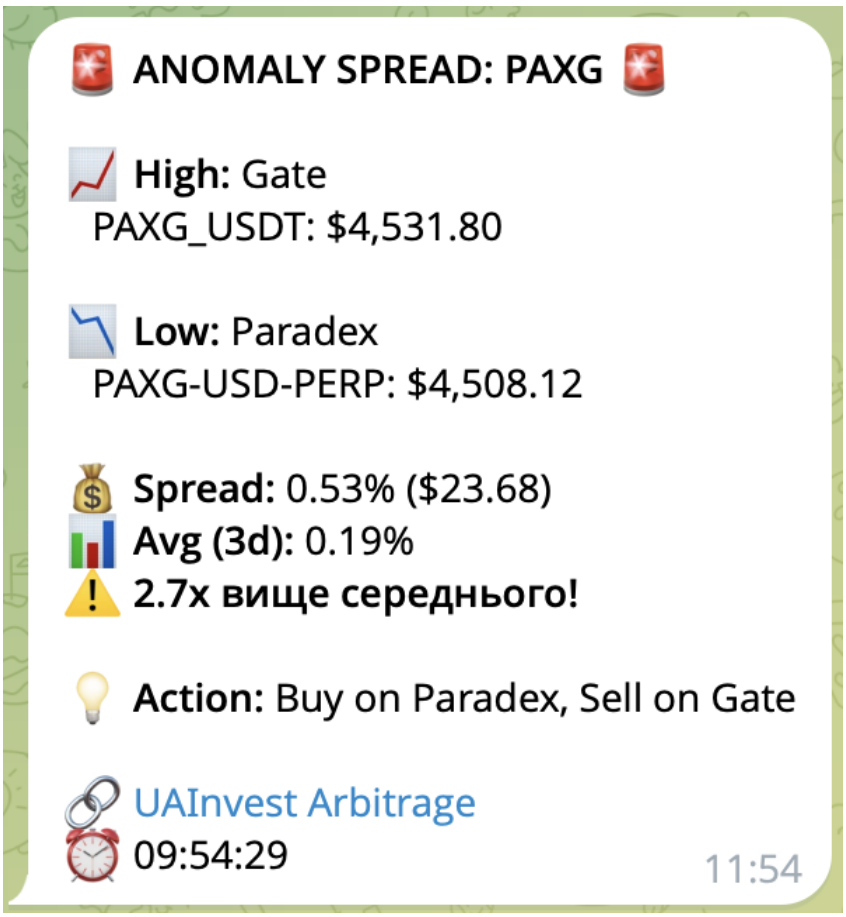


Рисунок В.6 – Приклад сповіщення модуля Gold Spread Scanner про аномальний спред товарних токенів (PAXG)

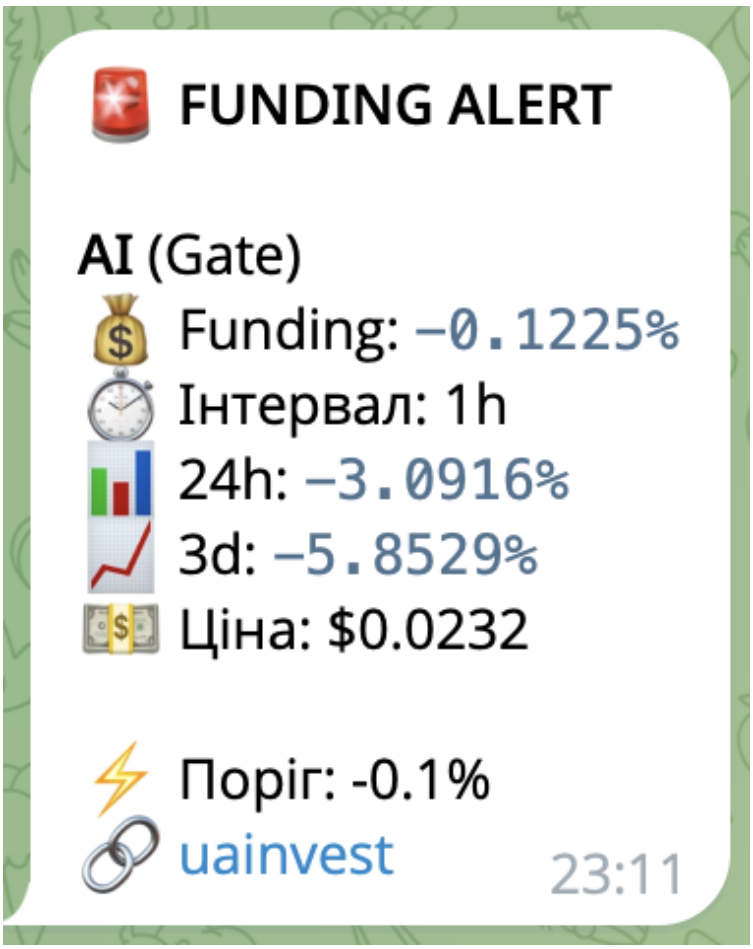


Рисунок В.7 – Приклад сповіщення модуля Funding Arbitrage про екстремальне значення funding rate (монета AI)

ДОДАТОК Г

Скрипт обробки журнальних файлів та шаблони таблиць результатів

Цей додаток містить повний лістинг допоміжного скрипта parse_logs.py, що використовується для витягування структурованих даних з журнальних файлів PM2 для подальшого статистичного аналізу. Також наведено шаблони таблиць результатів апробації, у яких конкретні чисельні значення будуть заповнені після обробки реальних логів за повний період дослідження.

Г.1 Скрипт parse_logs.py

Скрипт проходить по всіх журнальних файлах модулів у директорії /home/ubuntu/logs/, виявляє рядки-сповіщення за регулярними виразами, класифікує події та формує CSV-таблицю усіх подій для подальшого статистичного аналізу засобами pandas.

Лістинг – мова Python:

```
#!/usr/bin/env python3
"""Парсер журнальних файлів PM2 для подальшого статистичного
аналізу."""

import re
import csv
import sys
from pathlib import Path
from datetime import datetime

LOG_DIR = Path("/home/ubuntu/logs")
OUTPUT_CSV = Path("/home/ubuntu/parsed_alerts.csv")

# Регулярні вирази для виявлення подій у логах
RX_NEGATIVE_SPREAD = re.compile(
    r"^(?P<ts>\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.*?)\s+"
    r"\[(?P<exchange>\w+)\] Found: (?P<symbol>\w+) spread =
(?P<spread>-?[\d.]+)%"
)
RX_LOAN_APPEARED = re.compile(
```

```
    r"^(?P<ts>\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.*?)\s+"
    r"\[LoanWatch\] (?P<symbol>\w+) - LOANS APPEARED!\s+"
    r"Spread: (?P<spread>-?[\d.]+)%"
)
RX_ALERT_SENT = re.compile(
    r"^(?P<ts>\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.*?)\s+"
    r"Sent grouped alert for (?P<symbol>\w+)"
)


def parse_log_file(path: Path):
    events = []
    with open(path, 'r', encoding='utf-8') as f:
        for line in f:
            for event_type, rx in (
                ("NEGATIVE_SPREAD", RX_NEGATIVE_SPREAD),
                ("LOAN_APPEARED", RX_LOAN_APPEARED),
                ("ALERT_SENT", RX_ALERT_SENT),
            ):
                m = rx.match(line)
                if m:
                    d = m.groupdict()
                    d["event_type"] = event_type
                    d["module"] = path.stem
                    events.append(d)
                    break
    return events


def main():
    all_events = []
    for log_path in sorted(LOG_DIR.glob("*.log")):
        events = parse_log_file(log_path)
        print(f"  {log_path.name}: {len(events)} events")
        all_events.extend(events)

    if not all_events:
        print("No events parsed!")
        sys.exit(1)

    fieldnames = ["ts", "module", "event_type", "exchange",
"symbol", "spread"]
    with open(OUTPUT_CSV, 'w', encoding='utf-8', newline='') as
f:
        writer = csv.DictWriter(f, fieldnames=fieldnames,
extrasaction='ignore')
        writer.writeheader()
```

```
        for e in all_events:
            writer.writerow(e)

    print(f"\n✅ Saved {len(all_events)} events → {OUTPUT_CSV}")


if __name__ == "__main__":
    main()
```

Г.2 Шаблон таблиці агрегованих результатів за модулями

Шаблон для заповнення після обробки журнальних файлів. Кількість сигналів та інших метрик буде розрахована з parsed_alerts.csv засобами pandas.

Таблиця Г.1 – Кількість сигналів за модулями за період апробації

Модуль	Виявлено	Доставлено	TP	FP	Precision
Funding Arbitrage Bot	<N1>	<D1>	<TP1>	<FP1>	<P1>
Spread Tracker	<N2>	<D2>	<TP2>	<FP2>	<P2>
Gold Spread Scanner	<N3>	<D3>	<TP3>	<FP3>	<P3>
On-chain Scanner	<N4>	<D4>	<TP4>	<FP4>	<P4>
SpreadMonitor	<N5>	<D5>	<TP5>	<FP5>	<P5>
Loan Alert	<N6>	<D6>	<TP6>	<FP6>	<P6>
WhiteBit Tracker	<N7>	<D7>	<TP7>	<FP7>	<P7>
Усього	<NT>	<DT>	<TPT>	<FPT>	<PT>

Г.3 Шаблон таблиці результатів перевірки гіпотез

Таблиця Г.2 – Результати перевірки статистичних гіпотез

Гіпотеза	Тест	Статистика	p-value	Висновок
H1 (фільтрація)	z-критерій 2 пропорції	<z1>	<p1>	відхилити H0
H2 (двоконтурна модель)	парний t-критерій	<t1>	<p2>	відхилити H0
H3 (Pushover emergency)	z-критерій 2 пропорції	<z2>	<p3>	відхилити H0