Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Кібербезпеки*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка системи моніторингу сервісів комп'ютерної мережі та мережного обладнання
(тема)

Виконав:
здобувач 2 року навчання,
Владислав ГАВРИШ
(власне ім'я, прізвище)
Спеціальність *172. Електронні комунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *доц. каф. ІМІ, Станіслав ІВАНЕНКО*
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри _____ Микола *МОСКАЛЕЦЬ*
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей, заборонених до відкритого публікування

Здобувач _____ / *Владислав ГАВРИШ* /
(підпис) (власне ім'я, прізвище)

Керівник _____ / *Станіслав ІВАНЕНКО* /
(підпис) (власне ім'я, прізвище)

Харківський національний університет радіоелектроніки

Факультет *Кібербезпеки*
Кафедра *Інформаційно-мережної інженерії*
Рівень вищої освіти *другий (магістерський)*
Спеціальність *172. Електронні комунікації та радіотехніка*
(код і повна назва)
Тип програми *освітньо-наукова*
Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри _____
(підпис)
«___» _____ 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦФЙНУ РОБОТУ

здобувачеві *Гавришу Владиславу Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка системи моніторингу сервісів комп'ютерної мережі та мережного обладнання*
затверджена наказом університету від 23 *березня* 20*26* р. № *232 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *28.05.*2026 р.
3. Вихідні дані до роботи *розробка оптимальної конфігурації системи моніторингу для сервісів комп'ютерної мережі та проведення порівняльного аналізу її роботи з базовою конфігурацією системи моніторингу*
4. Перелік запитань, що необхідно опрацювати в роботі
Вступ
1 Теоретичні основи та порівняльний аналіз систем моніторингу
2 Методика розгортання та оптимізації системи моніторингу
3 Конфігурування та дослідження ефективності розробленої системи
4 Аналіз результатів оптимізації системи моніторингу
Висновки
5. Перелік графічного матеріалу із зазначенням креслень, схем, плакатів, комп'ютерних ілюстрацій *слайди презентації в форматі Power Point (назва роботи, вступ та мета роботи, сервер моніторингу, шаблони для вузлів мережі всередині серверу Zabbix. конфігурація серверу, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз завдання та літературних джерел	24.03.2026-30.03.2026	виконано
2	Аналіз особливостей систем моніторингу	31.03.2026-06.04.2026	виконано
3	Дослідження проблем систем моніторингу	07.04.2026-13.04.2026	виконано
4	Дослідження можливостей системи моніторингу Zabbix	14.04.2026-20.04.2026	виконано
5	Встановлення та налаштування системи моніторингу Zabbix	21.04.2026-27.04.2026	виконано
6	Вирішення проблем системи моніторингу	28.04.2026-04.05.2026	виконано
7	Аналіз результатів оптимізації системи моніторингу	05.05.2026-11.05.2026	виконано
8	Висновки	12.05.2026	виконано
9	Оформлення пояснювальної записки	13.05.2026-18.05.2026	виконано

Дата видачі завдання *23 березня* 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. каф. ІМІ, Станіслав Іваненко
(підпис) (посада, прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 61 с., 10 рис., 3 табл., 2 додатки, 12 джерел.

Об'єкт роботи – віддалені сервери та система моніторингу Zabbix

Мета роботи – проєктування та реалізація комплексної системи моніторингу мережних сервісів та телекомунікаційного обладнання на базі платформи Zabbix із розробкою методів підвищення її ефективності та надійності.

Актуальність теми зумовлена потребою підприємств у забезпеченні безперервності й стабільності функціонування корпоративних інформаційно-мережних систем, що критично залежить від ефективності інструментів автоматизованого контролю.

У кваліфікаційній роботі здійснено системний аналіз функціональних можливостей та архітектурних обмежень платформи Zabbix. Проведено порівняльний аналіз методів розгортання системи моніторингу та визначено оптимальну конфігурацію серверного середовища для її стабільного функціонування.

Виявлено та формалізовано ключові проблеми, що виникають у процесі адміністрування системи, зокрема: часові витрати на технічне обслуговування, складність масштабування конфігурацій, повторне використання уніфікованих шаблонів та надмірність збереження метрик. Запропоновано комплексні технічні рішення щодо оптимізації процесів налаштування, зниження апаратного навантаження та мінімізації надлишковості даних.

СИСТЕМА МОНІТОРИНГУ, ZABBIX, IT-ІНФРАСТРУКТУРА, МЕРЕЖНИЙ ВУЗОЛ, СЕРВЕРНЕ СЕРЕДОВИЩЕ, ОПТИМІЗАЦІЯ ДАНИХ

THE ABSTRACT

Explanatory note: 61 p., 10 fig., 3 tab., 2 apps, 12 sources.

Object of research – remote servers and the Zabbix monitoring system.

Purpose of the work – to design and implement an integrated monitoring system for network services and telecommunications equipment based on the Zabbix platform, developing methods to enhance its efficiency and reliability.

Relevance of the topic is driven by the need of enterprises to ensure the continuity and stability of corporate information and network systems, which critically depends on the efficiency of automated control tools.

The qualification work provides a systematic analysis of the functional capabilities and architectural limitations of the Zabbix platform. A comparative analysis of monitoring system deployment methods was conducted, determining the optimal configuration of the server environment for its stable operation.

Key challenges arising during system administration were identified and formalized, specifically: time costs for technical maintenance, configuration scaling complexity, reuse of unified templates, and data redundancy of metrics. Integrated technical solutions aimed at optimizing configuration processes, reducing hardware load, and minimizing data redundancy were proposed.

MONITORING SYSTEM, ZABBIX, IT INFRASTRUCTURE, NETWORK NODE, SERVER ENVIRONMENT, DATA OPTIMIZATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9
1 ТЕОРЕТИЧНІ ОСНОВИ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ СИСТЕМ МОНІТОРИНГУ 10
1.1 Критерії оцінювання та класифікація сучасних систем моніторингу мережної інфраструктури 11
1.2 Порівняльний аналіз архітектурних особливостей систем Zabbix, Prometheus та Nagios 11
1.3 Обґрунтування вибору платформи Zabbix для проєктування системи контролю мережних сервісів 12
1.4 Аналіз функціональних можливостей та архітектурних особливостей платформи Zabbix 14
1.5 Принципи функціонування та механізми взаємодії компонентів системи Zabbix 15
2 МЕТОДИКА РОЗГОРТАННЯ ТА ОПТИМІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ 24
2.1 Методика та етапи розгортання системи моніторингу Zabbix 25
2.2 Аналіз результатів розгортання та оцінювання початкового стану системи моніторингу Zabbix 31
3 КОНФІГУРУВАННЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ 33
3.1 Управління інцидентами та оперативне інформування персоналу 33
3.2 Аналітичний функціонал системи моніторингу в задачах оптимізації ресурсів та прогнозування 38
4 АНАЛІЗ РЕЗУЛЬТАТІВ ОПТИМІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ 40
4.1 Метрики оцінювання ефективності та методика тестування 40
4.2 Порівняльний аналіз споживання ресурсів до та після оптимізації 41
4.3 Оцінювання масштабованості системи моніторингу 43
ВИСНОВКИ 44
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 46
ДОДАТОК А – СЛАЙДИ ПРЕЗЕНТАЦІЇ 48
ДОДАТОК Б – ТЕЗИ ДОПОВІДІ НА ММФ 59

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CPU	Central Processing Unit	Центральний процесор
CLI	Command Line Interface	Інтерфейс командного рядка
DBMS	Database Management System	Система управління базами даних (СУБД)
DNS	Domain Name System	Система доменних імен
HDD	Hard Disk Drive	Жорсткий диск
HTTP	Hyper Text Transfer Protocol	Протокол передачі гіпертексту
HTTPS	Hyper Text Transfer Protocol Secure	Захищений протокол передачі гіпертексту
ICMP	Internet Control Message Protocol	Протокол міжмережних керуючих повідомлень
OS	Operating System	Операційна система (ОС)
RAM	Random Access Memory	Оперативний запам'ятовувальний пристрій (ОЗП)
SNMP	Simple Network Management Protocol	Простий протокол керування мережею
SSH	Secure Shell	Мережний протокол для безпечного доступу до системи
БД		База даних
ВМ		Віртуальна машина
ІТ		Інформаційні технології

ВСТУП

Ефективне функціонування сучасної корпоративної комп'ютерної мережі критично залежить від організації безперервного контролю параметрів її роботи. Навіть за умови високого рівня первинного конфігурування оперативне керування інфраструктурою не може покладатися виключно на людський фактор. Це зумовлює необхідність інтеграції автоматизованих засобів моніторингу стану мережі та систем сповіщення про інциденти.

Апаратні збої та програмні помилки у роботі критично важливих мережних вузлів, які залишаються неідентифікованими протягом тривалого часу, здатні дестабілізувати бізнес-процеси підприємства та призвести до матеріальних і репутаційних збитків. Раціональним науково-технічним рішенням є виявлення аномалій на ранніх стадіях їх виникнення. Спеціалізоване програмне забезпечення автоматизованого контролю дозволяє не лише оперативно інформувати персонал, а й здійснювати довгостроковий статистичний аналіз метрик продуктивності систем.

Паралельно з масштабуванням телекомунікаційного середовища виникає потреба в оптимізації систем моніторингу. Наявність даних про поточний стан інфраструктури є підґрунтям для планування модернізації, контролю, своєчасного оновлення системного оточення та мінімізації нераціональних капітальних витрат. Аналіз відповідності наявних апаратно-програмних засобів операційним завданням дозволяє ефективно координувати інвестиційні стратегії у сфері розвитку інформаційно-мережних технологій.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ СИСТЕМ МОНІТОРИНГУ

Сучасний етап розвитку інфокомунікаційних технологій характеризується стрімким масштабуванням мережної інфраструктури підприємств, ускладненням архітектури корпоративних мереж та підвищенням вимог до якості надання телекомунікаційних сервісів. У цих умовах забезпечення високого рівня доступності, надійності та відмовостійкості мережних елементів стає критично важливим завданням для інженерів та системних адміністраторів. Ефективне вирішення цієї проблеми неможливе без інтеграції автоматизованих систем моніторингу та аналізу мережного трафіку й стану обладнання [1].

Метою першого розділу кваліфікаційної роботи є теоретичне дослідження принципів побудови сучасних систем моніторингу та формування науково обґрунтованого базису для подальшого дослідження.

Для досягнення поставленої мети у розділі вирішуються такі завдання:

– визначення ключових класифікаційних ознак, архітектурних моделей (агентних, безагентних, Push та Pull моделей) та критеріїв оцінювання ефективності систем контролю IT-інфраструктури;

– проведення системного порівняльного аналізу найбільш поширених програмних рішень з відкритим вихідним кодом (Open Source), які застосовуються в сучасній практиці адміністрування;

– виявлення їхніх архітектурних обмежень, переваг та недоліків у контексті моніторингу телекомунікаційного обладнання та віддалених серверних систем;

– комплексне обґрунтування вибору програмної платформи, яка буде використана як базовий інструментарій для практичної реалізації дослідницьких завдань у наступних розділах роботи.

Проведений аналіз дозволить сформувати вимоги до проектованої системи та забезпечити мінімізацію ризиків, пов'язаних із надмірністю даних, складністю

масштабування та високими часовими витратами на технічне обслуговування корпоративної мережі.

1.1 Критерії оцінювання та класифікація сучасних систем моніторингу мережної інфраструктури

Для об’єктивного аналізу програмних рішень у сфері контролю IT-інфраструктури необхідно визначити ключові критерії оцінювання. Сучасні системи моніторингу класифікуються за методами збору даних (агентні та безагентні), архітектурою (централізовані, розподілені) та моделями обробки метрик (Push або Pull моделі).

Основними критеріями ефективності для корпоративних мереж підприємств є:

– Масштабованість та архітектурна гнучкість – здатність системи обробляти великі потоки даних без суттєвої деградації продуктивності.

– Універсальність збору даних – підтримка різноманітних протоколів (SNMP, ICMP, JMX, IPMI) та можливість безагентного моніторингу телекомунікаційного обладнання.

– Автоматизація та шаблонізація – наявність механізмів динамічного виявлення вузлів (Auto-discovery) та повторного використання конфігурацій.

– Економічна доцільність – вартість ліцензування, підтримки та вимог до апаратних ресурсів.

1.2 Порівняльний аналіз архітектурних особливостей систем Zabbix, Prometheus та Nagios

На сучасному ринку програмного забезпечення виділяються три фундаментальні архітектурні рішення: Nagios, Prometheus та Zabbix.

Nagios є однією з найстаріших систем. Вона базується на регулярному запуску зовнішніх скриптів (плагінів). Проте її архітектура має суттєві обмеження: високе навантаження на процесор через постійне створення нових

процесів (fork-процеси), складність конфігурування, виключно через текстові файли, та відсутність вбудованої глибокої аналітики історичних даних без сторонніх модулів.

Prometheus орієнтований на хмарні середовища (Cloud-Native) та роботу з контейнеризованими додатками (Kubernetes, Docker) за допомогою Pull-моделі (сервер сам опитує точки збору). Його слабким місцем у контексті класичних корпоративних мереж є орієнтація на короткострокове збереження метрик (потребує додаткових баз даних наприклад TimescaleDB для довготривалого аналізу), відсутність зручного вбудованого графічного інтерфейсу для адміністрування та складність налаштування моніторингу традиційного мережного обладнання через SNMP.

Zabbix представляє собою універсальну цілісну систему (All-in-One), що поєднує можливості збору даних (як агентним, так і безагентним методами), потужний вбудований механізм веб-інтерфейсу для керування, та оптимізовану схему збереження історичних даних у реляційних або тайм-серійних СУБД [2].

1.3 Обґрунтування вибору платформи Zabbix для проєктування системи контролю мережних сервісів

На основі проведеного системно-порівняльного аналізу (див. табл. 1.1) було визначено, що для вирішення завдань кваліфікаційної роботи – а саме розробки комплексної системи моніторингу сервісів комп'ютерної мережі та телекомунікаційного обладнання – платформа Zabbix є найбільш оптимальним та ефективним рішенням.

На відміну від Nagios, Zabbix має низькі вимоги до апаратних ресурсів при масштабуванні завдяки використанню проксі-серверів (Zabbix Proxy), що дозволяє будувати географічно розподілені архітектури моніторингу. У порівнянні з Prometheus, Zabbix «з коробки» володіє потужним інструментарієм для роботи з мережним обладнанням за протоколом SNMP, гнучкою системою шаблонізації та механізмом низькорівневого виявлення (LLD), що критично

важливо для динамічних корпоративних мереж. Повністю відкритий вихідний код (Open Source) за ліцензією GPLv2 нівелює фінансові витрати на придбання ліцензій, що суттєво знижує вартість впровадження системи.

Таблиця 1.1 – Порівняльна характеристика систем моніторингу

Критерій порівняння	Nagios	Prometheus	Zabbix
Основна орієнтація	Статус хостів (Up/Down)	Хмарні сервіси, метрики	Універсальний моніторинг інфраструктури
Метод збору даних	Скрипти / Агенти	Pull-модель (HTTP)	Агентний, безагентний (SNMP, IPMI, ICMP)
Розподілений моніторинг	Складно реалізувати	Потребує федеративної структури	Вбудований (Zabbix Proxy)
Веб-інтерфейс налаштування	Відсутній (текстові файли)	Базовий (потребує Grafana)	Повнофункціональний вбудований
Автовиявлення (Discovery)	Слабке	Добре (для хмар)	Відмінне (Вбудований LLD)
Вартість ліцензії	Безкоштовно (обмежено Core)	Open Source	Повністю Open Source (GPLv2)

Таким чином, архітектурні переваги, функціональна повнота, висока відмовостійкість та гнучкість конфігурування визначили вибір платформи

Zabbix як базового інструменту для подальшого проєктування та практичної реалізації системи моніторингу в межах цього дослідження.

1.4 Аналіз функціональних можливостей та архітектурних особливостей платформи Zabbix

Zabbix підтримує розподілений моніторинг аж до 1000 і більше вузлів. Конфігурація молодших вузлів повністю контролюється старшими вузлами, що знаходяться на більш високому рівні ієрархії. Основні можливості:

- сценарії на основі моніторингу;
- автоматичне виявлення;
- централізований моніторинг лог-файлів;
- веб-інтерфейс для адміністрування і настройки;
- звітність і тенденції;
- підтримка високопродуктивних агентів (zabbix-agent) практично для всіх платформ;
- комплексна реакція на події;
- підтримка SNMPv1,2,3;
- розширення за рахунок виконання зовнішніх програм;
- гнучка система шаблонів і груп;
- можливість створювати карти мереж.

Автоматичне виявлення:

- автоматичне виявлення за діапазоном IP-адрес, доступних сервісів і SNMP перевірка;
- автоматичний моніторинг виявлених пристроїв;
- автоматичне видалення відсутніх хостів;
- розподіл за групами і шаблонами в залежності від того що повертається в результаті [3].

1.5 Принципи функціонування та механізми взаємодії компонентів системи Zabbix

Zabbix складається з:

– сервера моніторингу, який виконує періодичне отримання даних, обробку, аналіз і запуск скриптів оповіщення;

– бази даних (MySQL, PostgreSQL, SQLite або Oracle);

– веб-інтерфейсу на PHP;

– агента-демона, який запускається на об'єктах і надає дані для сервера.

Використання спеціалізованого програмного агента (Zabbix Agent) є опціональним, оскільки архітектура платформи підтримує широкий спектр альтернативних методів збору даних. Інформаційне забезпечення процесу контролю може здійснюватися за допомогою мережного протоколу керування SNMP (версій v1, v2c, v3), шляхом ініціалізації зовнішніх користувацьких скриптів, а також за допомогою вбудованих зумовлених перевірок (Simple checks). Останні дозволяють верифікувати доступність та вимірювати час відгуку мережних сервісів за протоколами ICMP (ping), HTTP/HTTPS, SSH, FTP тощо.

Фундаментальною логічною одиницею архітектури системи моніторингу є вузол мережі (host) (рис. 1.1), який представляє собою окремий пристрій, телекомунікаційне обладнання або програмно-апаратний комплекс, що є об'єктом автоматизованого контролю.

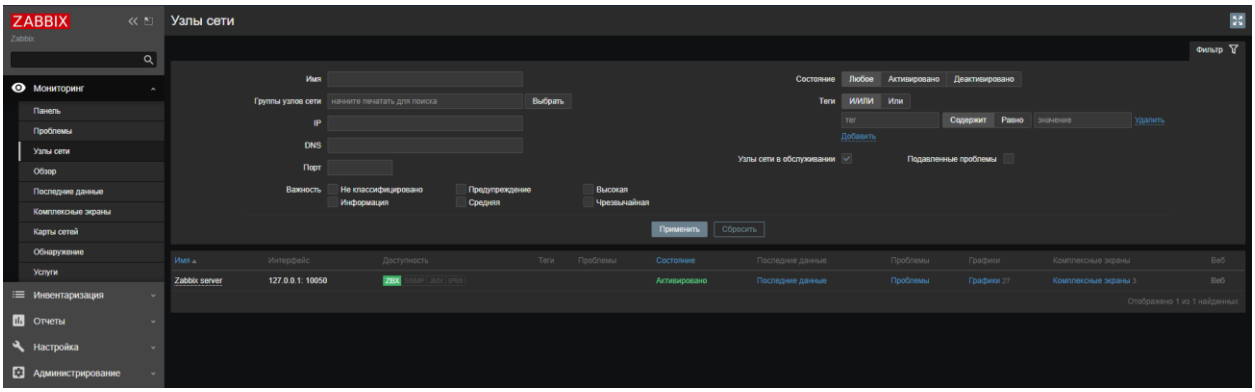


Рисунок 1.1 – Інтерфейс із вузлами мережі

Кожному вузлу присвоюється опис і адреса (DNS або IP, можна обидва, причому з можливістю вибирати, що використовувати для з'єднання) (рис. 1.2).

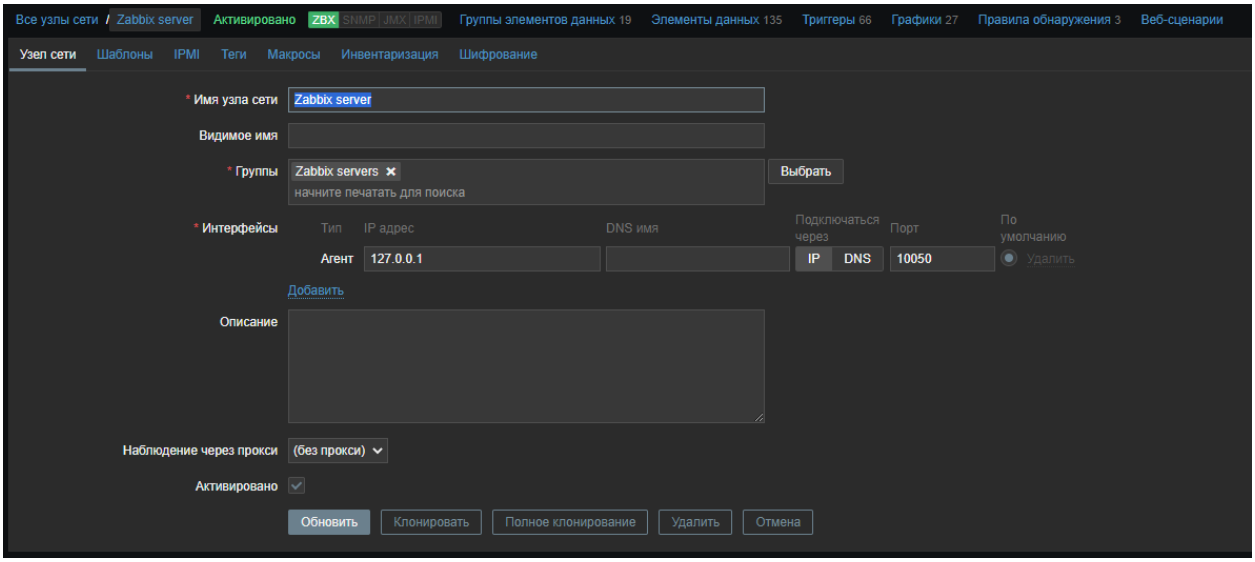


Рисунок 1.2 – Параметри підключення вузла мережі

Вузли об’єднуються в групи, наприклад веб-сервера або сервера баз даних. Групи служать для виведення тільки певних серверів при спостереженні (рис. 1.3).

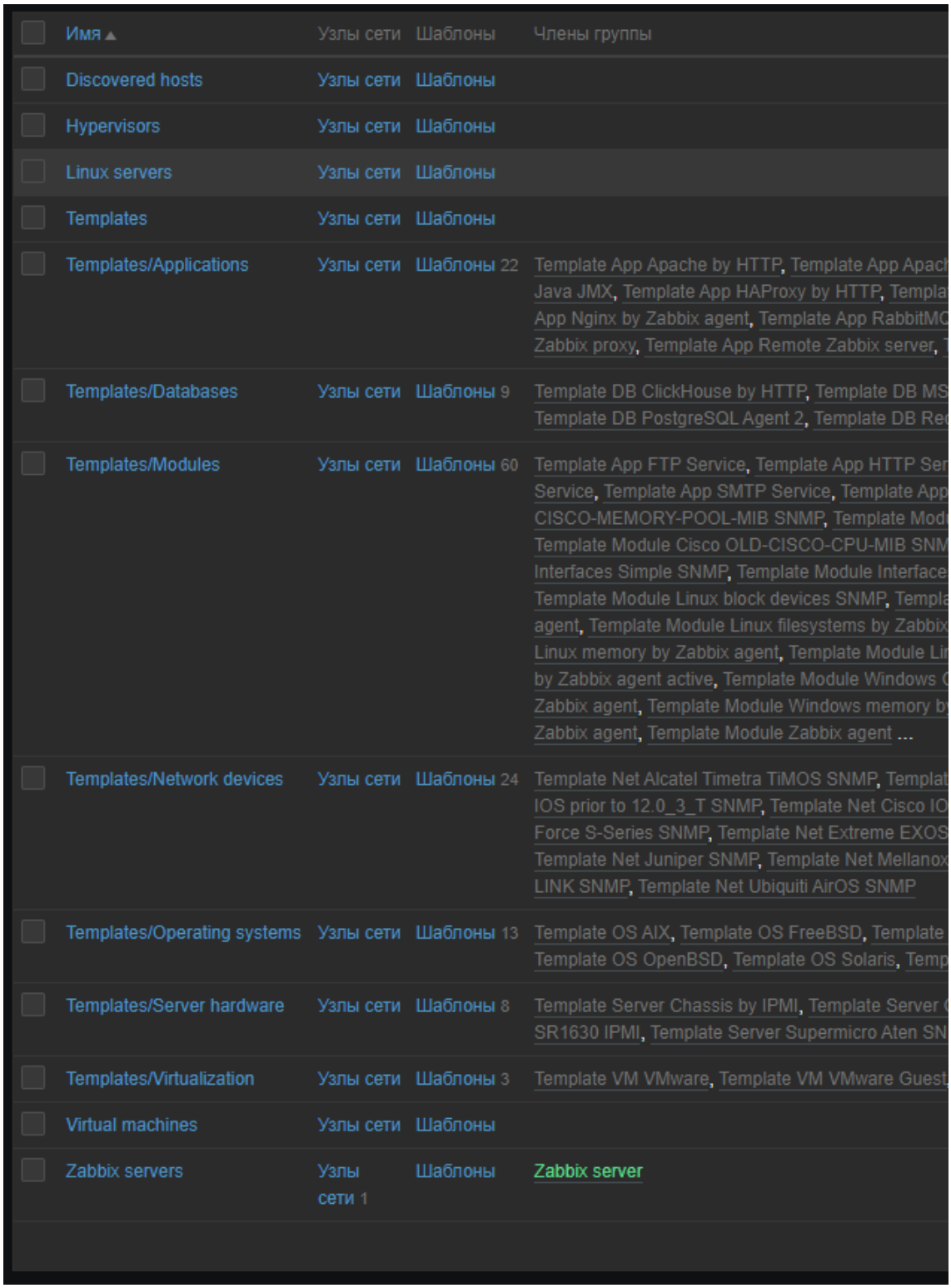


Рисунок 1.3 – Групи вузлів мережі

Кожен вузол має кілька елементів даних (items) – метрики, за якими ведеться моніторинг (рис. 1.4).

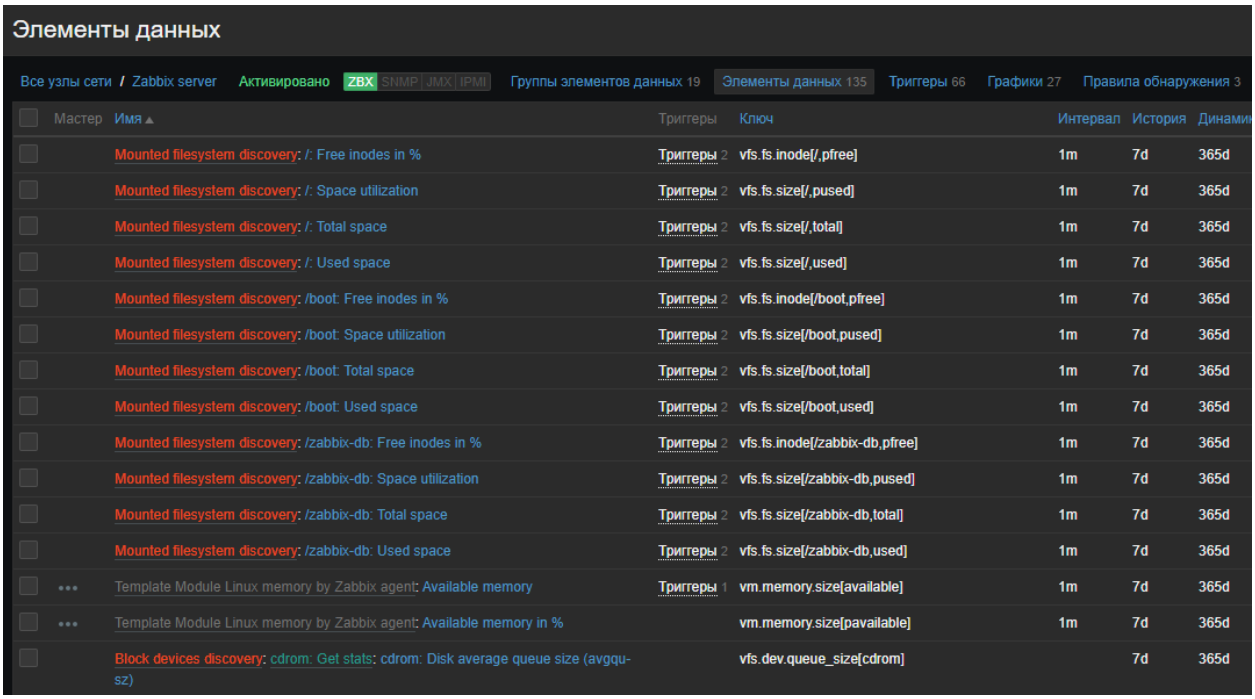


Рисунок 1.4 – Метрики вузла мережі

Наприклад, для оцінювання мережної доступності об'єктів інфраструктури використовується вбудована перевірка, що реєструє параметр доступності за протоколом ICMP. Дана метрика набуває булевого значення: «1» – у разі успішного отримання ехо-відповіді на останній запит, та «0» – за її відсутності. Водночас для специфічних вузлів (наприклад, веб серверів) можлива інтеграція кастомних метрик, як-от «кількість активних сесій користувачів», значення якої формується шляхом виконання користувацького скрипта зчитування даних безпосередньо з веб ресурсу. Для кожного окремого елемента даних (item) існує гнучка можливість конфігурування індивідуальних параметрів обробки: періоду оновлення, методу агрегації та збереження даних (абсолютне значення або дельта швидкості його зміни), застосування користувацьких множників, а також налаштування специфічних часових інтервалів збору метрик (наприклад, виключно в межах робочого часу організації).

З метою раціоналізації часових витрат адміністратора та автоматизації процесу менеджменту великої кількості однотипних об'єктів, в архітектурі Zabbix реалізовано концепцію шаблонізації. Шаблон (template) є уніфікованою

логічною структурою, що містить зумовлені набори елементів даних, тригерів та графіків, які не підлягають безпосередньому моніторингу, а виступають як базовий профіль конфігурації. Прив'язка реального вузла мережі до одного або декількох шаблонів забезпечує автоматичне успадкування хостом усіх задекларованих параметрів. Створення базового елемента даних, наприклад, метрики контролю доступності, здійснюється на рівні абстрактного шаблону (зокрема, «Template Ping»), після чого цей параметр автоматично реплікується на всі асоційовані з цим шаблоном мережні вузли (рис. 1.5).

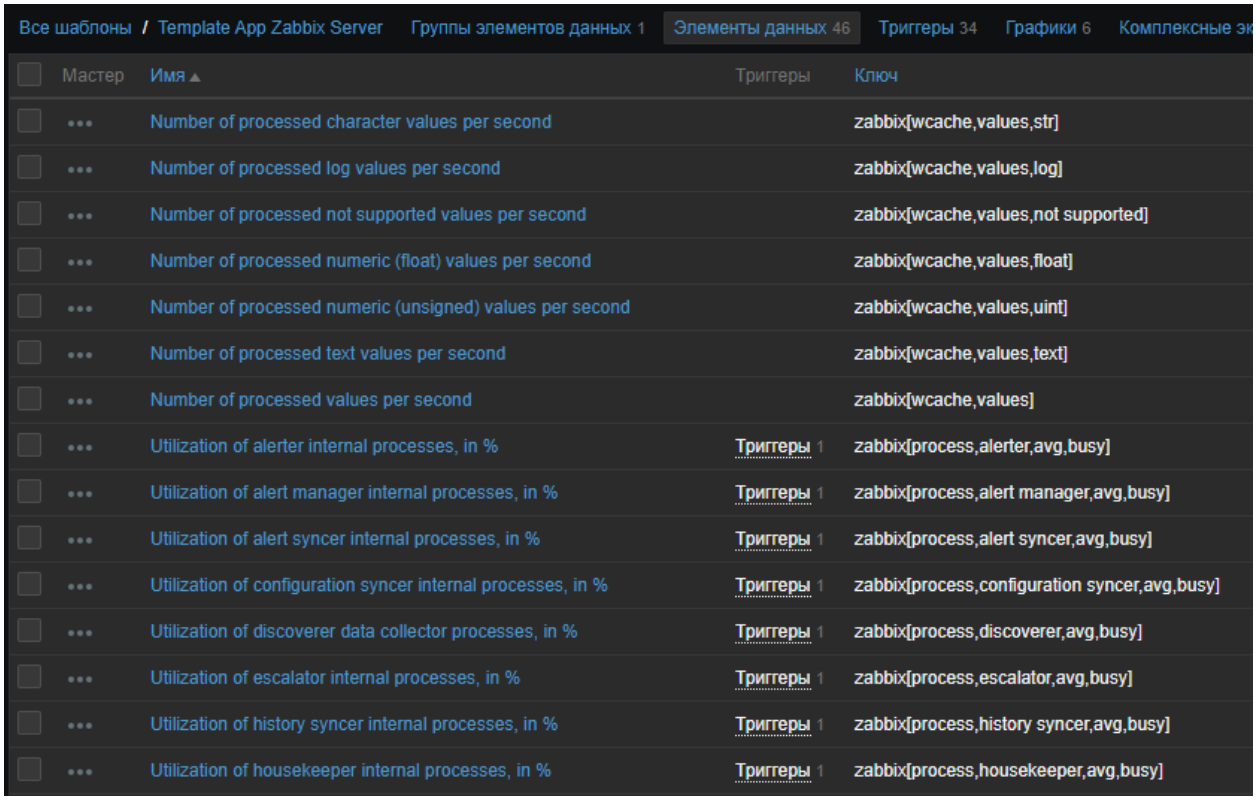


Рисунок 1.5 – Приклад шаблону для вузла мережі

Оскільки безперервний ручний контроль значного масиву телекомунікаційних параметрів є неефективним через обмеження людського фактору, у системі Zabbix реалізовано механізм автоматизації обробки інцидентів за допомогою тригерів. Тригер є логічним виразом, який визначає

критерії переходу системи у стан аварії (аномалії) при досягненні метриками гранично допустимих значень.

У разі ініціалізації тригера система активує систему сповіщення адміністратора, що включає візуальну індикацію на панелі моніторингу (Dashboard) та генерацію звукових сигналів через веб-інтерфейс. Зокрема, в уніфікованому шаблоні «Template Ping» інтегровано базовий тригер, який автоматично фіксує критичне відхилення параметрів доступності об'єкта, якщо час відсутності відповіді на ICMP-запити (ICMP loss) перевищує інтервал у дві хвилини (рис. 1.6).

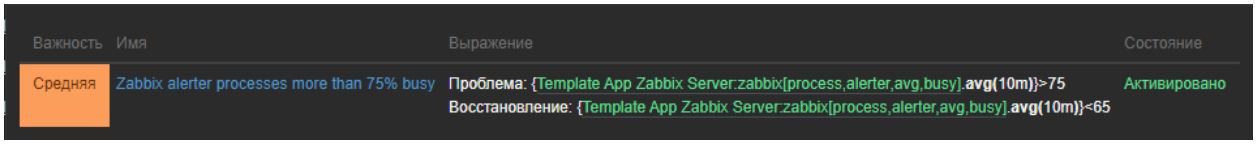
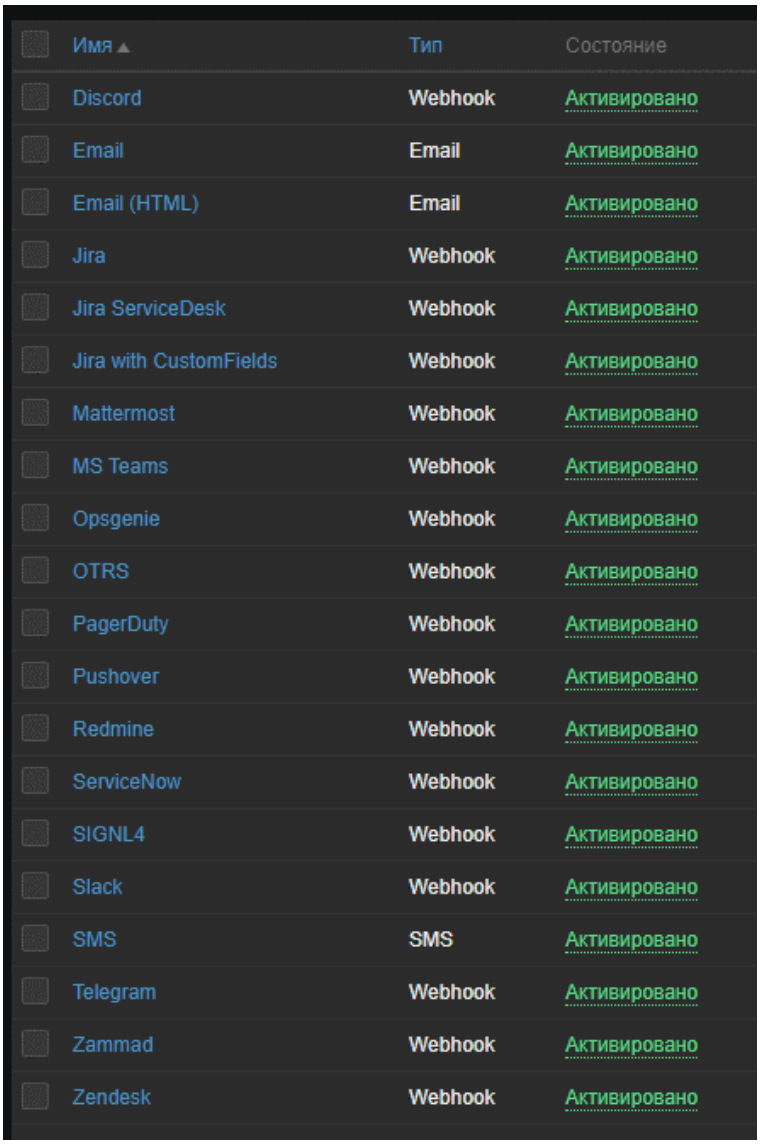


Рисунок 1.6 – Приклад тригера

Для забезпечення безперервності процесів адміністрування за відсутності оператора в системі Zabbix реалізовано механізми автономного функціонування. Програмний комплекс підтримує архітектуру проактивного інформування за допомогою інтеграції з різноманітними каналами зв'язку. Сповіщення про критичні інциденти можуть надсилатися через протокол SMTP, за допомогою API-інтерфейсів сучасних месенджерів та корпоративних платформ (Telegram, Slack), а також у вигляді SMS-повідомлень або голосових викликів через підключений GSM-модем (рис. 1.7).

Крім того, функціонал системи не обмежується пасивним моніторингом і сповіщенням: в архітектурі Zabbix передбачено підсистему автоматичного реагування на інциденти (Remote commands). При переході тригера в стан аварії система здатна ініціалізувати заздалегідь визначені алгоритми відновлення працездатності – наприклад, автоматичний перезапуск деградованого мережного

сервісу або демона, що дозволяє мінімізувати час простою інфраструктури (RTO) без безпосереднього залучення технічного персоналу.



Имя ▲	Тип	Состояние
Discord	Webhook	Активировано
Email	Email	Активировано
Email (HTML)	Email	Активировано
Jira	Webhook	Активировано
Jira ServiceDesk	Webhook	Активировано
Jira with CustomFields	Webhook	Активировано
Mattermost	Webhook	Активировано
MS Teams	Webhook	Активировано
Opsgenie	Webhook	Активировано
OTRS	Webhook	Активировано
PagerDuty	Webhook	Активировано
Pushover	Webhook	Активировано
Redmine	Webhook	Активировано
ServiceNow	Webhook	Активировано
SIGNL4	Webhook	Активировано
Slack	Webhook	Активировано
SMS	SMS	Активировано
Telegram	Webhook	Активировано
Zammad	Webhook	Активировано
Zendesk	Webhook	Активировано

Рисунок 1.7 – Список способів сповіщень

За даними будь-якого параметра система зможе побудувати графік зміни, причому не за зумовлені і жорстко задані часові інтервали, а за будь-який проміжок часу з максимальною роздільною здатністю. Хочете подивитися в деталях, як змінювалося навантаження на сервер місяць тому? Будь ласка, графік з дозволом в 30 секунд (саме такий інтервал опитування за замовчуванням) до ваших послуг. Хочете загальну картину? Виберіть інтервал в місяць і подивіться

на середню величину, і розкид коливань до максимуму і мінімуму. Порівняти? Можна створювати складні графіки, що відображають на одному полі кілька параметрів, і ви відразу побачите, що пікові значення відповідають піків трафіку (рис. 1.8).

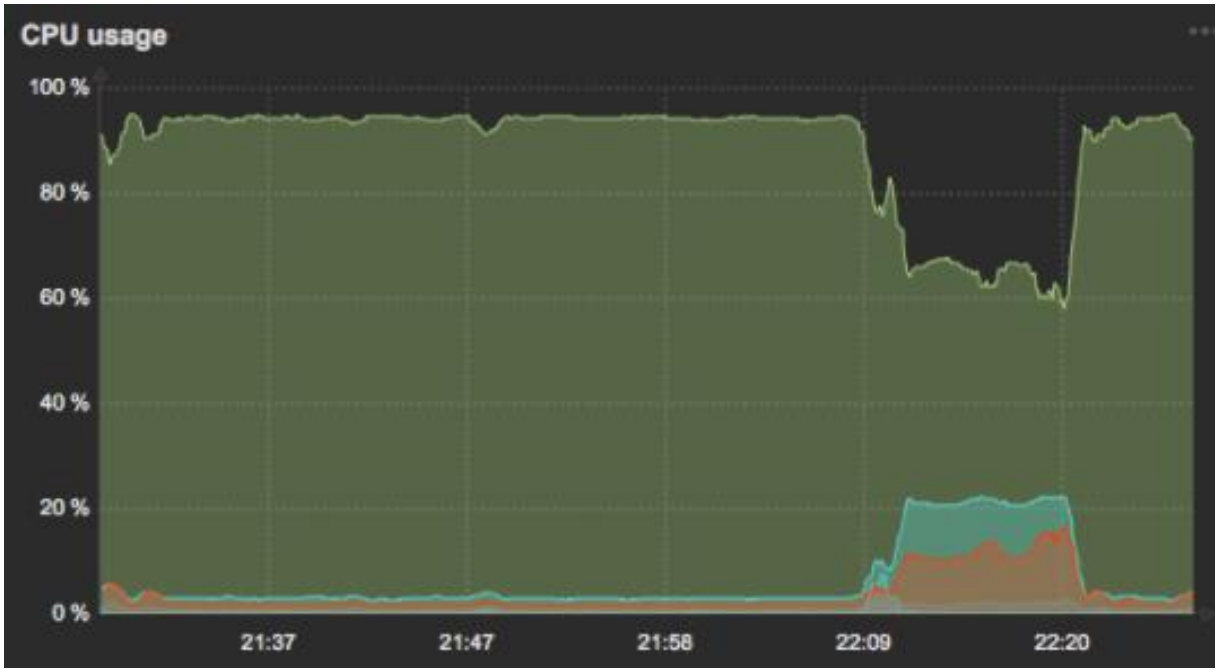


Рисунок 1.8 – Приклад графіку

Для відображення логічної структури мережі можна створювати карти мережі, що відображають саме розташування вузлів мережі і зв'язків між ними. Природно, стан вузлів (доступний чи ні) відображається і на карті.

Крім того, для більш зручного огляду є комплексні звіти, які дозволяють на одному екрані переглядати відразу кілька сутностей – графіки, дані, тригери (рис. 1.9) [3].

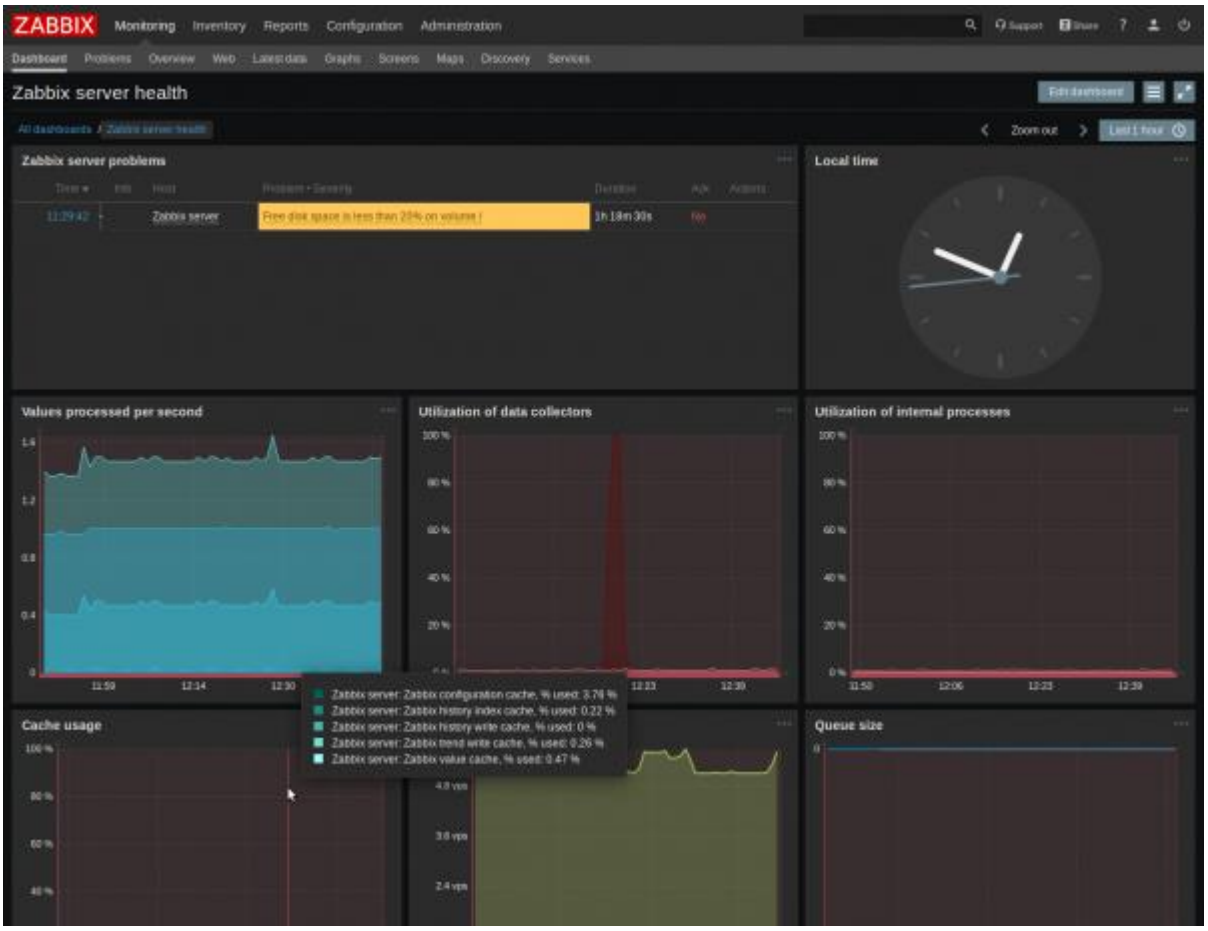


Рисунок 1.9 – Приклад комплексного екрану

2 МЕТОДИКА РОЗГОРТАННЯ ТА ОПТИМІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ

У Zabbix є три способи встановлення:

– встановлення з пакетів – Zabbix поставляє офіційні RPM і DEB пакети для Red Hat Enterprise Linux, Debian і Ubuntu LTS. Також доступні репозиторії yumi apt;

– завантаження архіву з вихідними кодами і самостійна їх складання;

– завантаження готового рішення – повністю встановлена і налаштована система на віртуальному диску або інсталяційний образ CD; готове рішення Zabbix присутнє в наступних форматах: vmdk (VMware/Virtualbox); OVF (Open Virtualisation Format); KVM; HDD/flash image, USB stick; Live CD/DVD; Xen guest; Microsoft VHD (Azure); Microsoft VHD (Hyper-V).

Під час проєктування та розгортання серверної частини системи моніторингу особливу увагу приділено вибору методу інсталяції програмного забезпечення та компонентів інфраструктури. Використання зумовлених репозиторіїв та готових пакетних рішень (дистрибутивів), попри високу швидкість розгортання, характеризується значним рівнем надлишковості, оскільки призводить до інсталяції невикористовуваних залежностей та сервісів, що потенційно розширює вектор атак і знижує загальну безпеку системи. З метою забезпечення максимальної детермінованості конфігурації, оптимізації продуктивності та повного контролю над архітектурою середовища, було прийнято рішення щодо відмови від пакетних менеджерів на користь ручної компіляції та інсталяції платформи Zabbix безпосередньо з офіційних вихідних кодів (Source codes).

Як систему управління базами даних (СУБД), виходячи з критеріїв високої поширеності, відкритого вихідного коду, високої швидкодії при обробці реляційних структур та простоти адміністрування, було обрано MySQL.

Базовою операційною системою для розгортання компонентів Zabbix Server виступив дистрибутив Ubuntu Server (останньої стабільної версії з довготривалою підтримкою – LTS). Вибір серверної модифікації (Server Edition) обумовлений відсутністю необхідності у функціонуванні графічного інтерфейсу користувача (GUI) на стороні сервера, що дозволяє раціонально перерозподілити апаратні ресурси на користь обчислювальних процесів ядра моніторингу та взаємодії через консольний інтерфейс командного рядка (CLI).

Для забезпечення гнучкості масштабування, ізоляції середовища та спрощення процедур резервного копіювання, розгортання операційної системи та всього програмного комплексу реалізовано на базі технологій віртуалізації шляхом створення виділеної віртуальної машини (ВМ) з оптимізованим виділенням апаратних потужностей.

2.1 Методика та етапи розгортання системи моніторингу Zabbix

Весь алгоритм встановлення описаний в табл. 2.1 [4].

Таблиця 2.1 – Алгоритм встановлення Zabbix

Команда [параметр]
VM: Zabbix OS: ubuntu-20.04.3-live-server-amd64 RAM: 8192 M CPU: 4 GPU: 64 M CD-ROM: SATA port 0 HDD: SATA port 1 HDD: SATA port 2 USB: 3.0 Install OS Defaults Your name: Your server's name: Pick a username Password Install OpenSSH server

Продовження таблиці 2.1

```
Defaults
Reboot > Login > Shutdown > Snapshot

Install Guest Additions
Insert Guest Additions CD Image
sudo mkdir /media/cdrom
sudo mount /dev/cdrom /media/cdrom
sudo apt-get install -y dkms build-essential linux-headers-generic linux-
headers-$(uname -r)
sudo su
cd /media/cdrom
./VBoxLinuxAdditions.run
exit
sudo reboot
lsmod | grep -io vboxguest
sudo rmdir /media/cdrom
sudo shutdown
Snapshot

Always connect via SSH
sudo apt-get update
sudo apt-get upgrade -y
sudo reboot
sudo shutdown
Snapshot

sudo apt-get update
sudo apt-get install mysql-client mysql-server -y
sudo mysql_secure_installation
root
everywhere y
Disallow root login remotely?
everywhere y
Server version: 8.0.20-0ubuntu0.20.04.1 (Ubuntu)
sudo mysql
SELECT user,authentication_string,plugin,host FROM mysql.user;
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'i@cHfMWkrKSb^Th8J';
FLUSH PRIVILEGES;
SELECT user,authentication_string,plugin,host FROM mysql.user;
exit
systemctl status mysql.service
mysql -u root -p
select @@datadir;
exit
sudo systemctl stop mysql
```

Продовження таблиці 2.1

```
sudo systemctl status mysql
sudo rsync -av /var/lib/mysql /zabbix-db
sudo mv /var/lib/mysql /var/lib/mysql.bak
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
datadir=/zabbix-db/mysql
sudo nano /etc/apparmor.d/tunables/alias
alias /var/lib/mysql/ -> /zabbix-db/mysql/,
sudo systemctl restart apparmor
sudo mkdir /var/lib/mysql/mysql -p
sudo systemctl start mysql
sudo systemctl status mysql
mysql -u root -p
select @@datadir;
exit

sudo rm -Rf /var/lib/mysql.bak
sudo systemctl restart mysql
sudo systemctl status mysql
sudo reboot
sudo systemctl status mysql
sudo shutdown
Snapshot

mysql -u root -p
CREATE DATABASE zabbix CHARACTER SET UTF8 COLLATE UTF8_BIN;
CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'itw2XtQHAJCw%eiHo';
SELECT user,authentication_string,plugin,host FROM mysql.user;
GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
FLUSH PRIVILEGES;
exit

cd ~
wget https://cdn.zabbix.com/zabbix/sources/stable/5.0/zabbix-5.0.2.tar.gz
tar -zxvf zabbix-5.0.2.tar.gz
cd zabbix-5.0.2/database/mysql/
mysql -u zabbix -p zabbix < schema.sql
mysql -u zabbix -p zabbix < images.sql
mysql -u zabbix -p zabbix < data.sql
sudo nano /etc/mysql/mysql.conf.d/zabbix.cnf
"[mysqld]
innodb_io_capacity=650
innodb_buffer_pool_size = 6G
innodb_buffer_pool_instances=16
innodb_log_file_size = 512M
innodb_flush_method=O_DSYNC"
```

Продовження таблиці 2.1

```
wget https://bestmonitoringtools.com/wp-
content/uploads/2019/10/zbx_db_partitiong.tar.gz
tar -zxvf zbx_db_partitiong.tar.gz
nano zbx_db_partitiong.sql

"                CALL partition_maintenance(SCHEMA_NAME, 'history', 180,
24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_log', 180,
24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_str', 180,
24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_text',
180, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_uint',
180, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends', 730, 24,
3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends_uint', 730,
24, 3);"
mysql -u 'zabbix' -p'itw2XtQHAJCw%eiHo' zabbix <  zbx_db_partitiong.sql
sudo crontab -e
30 03 * * * /usr/bin/mysql -u 'zabbix' -p'itw2XtQHAJCw%eiHo' zabbix -e
"CALL partition_maintenance_all('zabbix');" > /tmp/CronDBpartitiong.log
2>&1
mysql -u 'zabbix' -p'itw2XtQHAJCw%eiHo' zabbix -e "CALL
partition_maintenance_all('zabbix');"
sudo dpkg-reconfigure tzdata
sudo apt-get update
sudo apt-get upgrade -y

sudo apt-get install ntpdate ntp -y
sudo ntpdate pool.ntp.br
date

sudo apt-get install nginx -y
sudo service nginx restart
sudo service nginx status
sudo apt-get install php-fpm php-mysql php-mbstring php-xml php-gd php-curl
php-bcmath php-ldap -y
sudo apt-get install locate -y
sudo updatedb
locate php.ini
sudo nano /etc/php/7.4/fpm/php.ini
"max_execution_time = 300
max_input_time = 300
memory_limit = 256M
```

Продовження таблиці 2.1

```
post_max_size = 32M
upload_max_filesize = 32M
date.timezone = Europe/Kiev"
sudo nano /etc/nginx/sites-available/default

"server {
listen 80 default_server;
listen [::]:80 default_server;
root /var/www/html;
index index.php index.html index.htm;
server_name _;
location / {
try_files $uri $uri/ =404;
}
location ~ .php$ {
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
}
}"
sudo chown -R www-data:www-data /var/log/nginx;
sudo chmod -R 755 /var/log/nginx;
sudo nginx -t
sudo service php7.4-fpm restart && sudo service php7.4-fpm status
sudo service nginx restart && sudo service nginx status

sudo add-apt-repository ppa:certbot/certbot
sudo apt-get install python3-certbot-nginx -y
sudo nano /etc/nginx/sites-available/default
server_name zabbix-memolog.sytes.net;
sudo service nginx restart && sudo service nginx status
sudo ufw status
sudo certbot --nginx -d zabbix-memolog.sytes.net
sudo certbot renew --dry-run
sudo service php7.4-fpm restart && sudo service php7.4-fpm status
sudo service nginx restart && sudo service nginx status

mkdir ~/go -p && cd ~/go
wget https://dl.google.com/go/go1.14.6.linux-amd64.tar.gz
sudo tar -C /usr/local -xzf go1.14.6.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin
sudo reboot
sudo shutdown
Snapshot

export PATH=$PATH:/usr/local/go/bin
sudo rm /etc/apt/sources.list.d/certbot-ubuntu-certbot-focal.list
```

Продовження таблиці 2.1

```
sudo apt-get update
sudo apt-get install build-essential libmysqlclient-dev libssl-dev libsnmp-
dev libevent-dev pkg-config -y
sudo apt-get install libopenipmi-dev libcurl4-openssl-dev libxml2-dev
libssh2-1-dev libpcre3-dev -y
sudo apt-get install libldap2-dev libiksemel-dev libcurl4-openssl-dev
libgnutls28-dev -y
cd ~/zabbix-5.0.2
./configure --enable-server --enable-agent --enable-agent2 --with-mysql --
with-openssl --with-net-snmp --with-openipmi --with-libcurl --with-libxml2
--with-ssh2 --with-ldap
sudo su
make install
updatedb
locate zabbix_server.conf
nano /usr/local/etc/zabbix_server.conf

"LogFile=/tmp/zabbix_server.log
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=itw2XtQHAJCW%eiHo
Timeout=4
LogSlowQueries=3000
StatsAllowedIP=127.0.0.1
StartVMwareCollectors=5"
/usr/local/sbin/zabbix_server
/usr/local/sbin/zabbix_agent2
exit
mv ~/zabbix-5.0.2/ui /var/www/html/zabbix
chown www-data.www-data /var/www/html/zabbix/* -R
sudo service php7.4-fpm restart && sudo service php7.4-fpm status
sudo service nginx restart && sudo service nginx status
passwd
Bv$9YQ56@qY5skSA&

cat /usr/share/i18n/SUPPORTED | grep uk_
sudo locale-gen ru_RU.UTF-8
sudo locale-gen uk_UA.UTF-8
sudo dpkg-reconfigure locales
sudo service php7.4-fpm stop
sudo service php7.4-fpm start
sudo service php7.4-fpm status
sudo service nginx stop
sudo service nginx start
```

2.2 Аналіз результатів розгортання та оцінювання початкового стану системи моніторингу Zabbix

Після встановлення системи моніторингу отримано такі дані:

– посилання – memolog.sytes.net/zabbix (якщо хост має «білу» IP-адресу і зареєстровано DNS-ім’я);

– користувач WEB-інтерфейсу – Admin:d5#JoojQZK*ILhE5K;

– доступ до хоста по SSH – memolog.sytes.net:10648, zabbix:Bv$9YQ56@qY5skSA& (мій хост знаходиться за роутером, на якому налаштована переадресація портів з 10648 на 22);

– доступ до користувача СУБД – zabbix: itw2XtQHAJCw%eiHo;

– назва та пароль БД – zabbix: itw2XtQHAJCw%eiHo.

Після авторизації у WEB-інтерфейсі повинна відображатися панель за замовчуванням «Global view» (рис. 2.1).

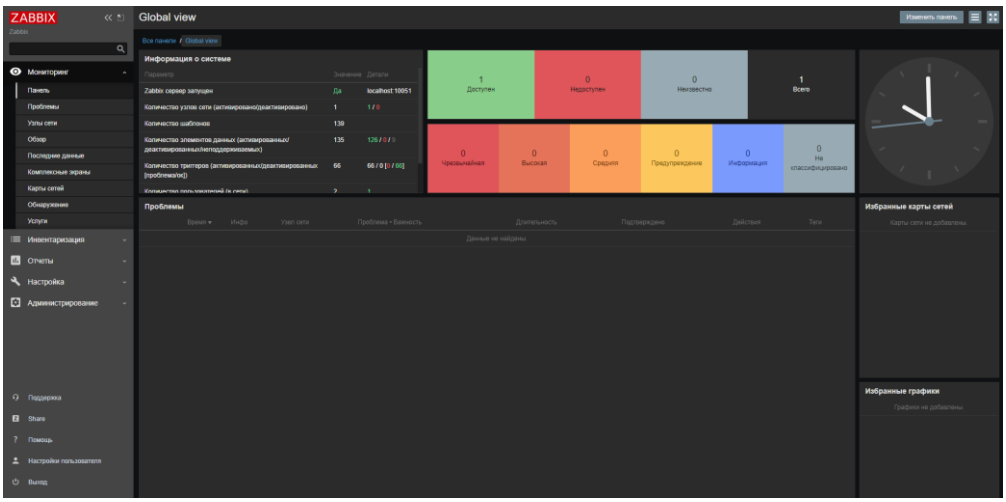


Рисунок 2.1 – приклад панелі «Global view»

Вирішені проблеми на етапі встановлення:

– дані під захистом HTTPS-з’єднання – так як вся інформація переглядається через браузер, то всі дані з вузлів мережі передаються через глобальну мережу (якщо сервер не встановлений локально) [5]

– БД не буде нескінченно збільшуватися з часом за рахунок модифікації файлу «zbx_db_partitiong.sql» – він відповідає за параметри автоматичної очистки БД [6]

– заздалегідь покращена продуктивність серверу від 10 до 50 разів у порівнянні з параметрами за замовчуванням за рахунок правильно налаштованого WEB-серверу та MySQL, параметри яких цілком відповідають фізичній конфігурації хоста

3 КОНФІГУРУВАННЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

Повноцінна система моніторингу повинна грати кілька ролей:

– виступати як система оповіщення про проблеми;

– виступати як засіб аналізу, за допомогою якого можна вивчити стан системи в заданий момент часу;

– виступати як система, що надає необхідні дані для прийняття рішень про конфігурацію інфраструктури, найближчих і стратегічних завданнях.

Для кожної з цих ролей є певні вимоги. Вони обов'язково повинні виконуватися для забезпечення стабільної підтримки.

Уявлення про те, що саме треба буде виконувати, має бути закладено в момент розробки програми та архітектури, і мова навіть не стільки про серверну архітектуру, скільки про архітектуру програми в цілому.

Розробники / архітектори повинні розуміти які частини системи критичні для функціонування проєкту і бізнесу, і заздалегідь думати про те, що їх працездатність треба перевіряти.

Моніторинг повинен бути зручним для адміністратора, і давати уявлення про те, що відбувається. Мета моніторингу – вчасно отримати оповіщення, за графіками швидко зрозуміти, що саме відбувається і що саме потрібно ремонтувати.

3.1 Управління інцидентами та оперативне інформування персоналу

Базова роль моніторингу – це інформування про наближення проєкту до критичних значень, а також про вже трапилися проблеми, аваріях, порушення бізнес-функцій проєкту.

Система моніторингу повинна бути максимально незалежною від самого проєкту. Якщо розмістити її на тому ж майданчику, то система не помітить

недоступність майданчику для зовнішнього світу. А якщо розмістити моніторинг на тих же серверах, де розміщений і сам проєкт, то в разі серверної аварії моніторинг буде недоступний.

Система моніторингу повинна максимально швидко оповіщати про досягнення критичних показників і аваріях. Якщо інформація про проблему не приходить до чергових або приходить занадто пізно, то не буде можливості прийняти ефективні заходи щодо запобігання або вирішення проблеми. Здавалося б, що це очевидна вимога, однак мова йде не просто про швидку доставку сповіщень, а й про впевненість в тому, що чергові адміністратори отримають повідомлення про помилку. Перш за все, оповіщення повинні приходити не тільки по e-mail. Якщо чергових кілька, добре б додати в смс-повідомлення посилання для підтвердження отримання. Якщо черговий адміністратор не перейшов по посиланню протягом декількох хвилин – смс йде наступного адміністратору в ланцюжку ескалації.

До всього іншого система моніторингу повинна не тільки швидко, але і надійно сповіщати про події в системі. Чи не спрацював оповіщення може спричинити за собою годинник простою або досягнення такої ситуації, в якій для виправлення проблеми доведеться затратити величезну кількість часу. Тому якщо метрики проєкту досягли критичних показників – оповіщення про це повинно бути неодмінно зроблено. Для перевірки системи оповіщення рекомендується регулярно примусово створювати «критичні» показники.

Сповіщення повинні бути максимально зрозумілими: адміністратор, отримавши сповіщення, навіть якщо він не знайомий з системою, повинен зрозуміти, про що це сповіщення, в яку документацію дивитися, чи хоча б кому зателефонувати. Повинні бути чіткі інструкції, що саме потрібно робити, і як саме вирішувати проблему [7].

Коли виникає проблема, дуже хочеться зрозуміти, через що вона виникла. Отримуючи сповіщення про те, що у вас не працює додаток, вам дуже хотілося б знати, які ще суміжні системи ведуть себе не так, які ще є відхилення від норми.

Повинні бути зрозумілі графіки, зібрані в панелі, з яких відразу буде видно відхилення.

Для цього треба точно розуміти, що нормально, а що не є нормальним. Тобто повинна бути достатня історична довідка про стан системи. Завдання полягає в тому, щоб покрити сповіщення всі можливі відхилення від норми.

Коли адміністратор отримує сповіщення, він або повинен знати що з ним робити, або кого запитати. Повинна бути інструкція як саме потрібно реагувати, і її треба регулярно оновлювати. Якщо у вас все працює через систему оркестрування, то, напевно, у вас все добре, якщо всі зміни відбуваються тільки через неї, в тому числі і моніторинг. Система оркестрування дозволяє адекватно стежити за актуальністю моніторингу.

Моніторинг повинен розширюватися після кожної аварії – якщо раптово виникла якась проблема, яка проскочила повз моніторингу, то, очевидно, треба додати метрику, щоб в наступний раз проблема не була раптовою.

Дуже часто в практиці зустрічаються проєкти, на яких моніторинг закриває тільки технічні показники серверної системи. Однак на ділі цим ніяк не можна обмежуватися. У разі, наприклад, помилок в коді в новій версії сайту на сервері не відбуватиметься ніякого навантаження, але це зовсім не означає що з проєктом все добре. Правильна система оповіщення повинна перевіряти працездатність трьох шарів проєкту:

– фіксувати проблеми на рівні сервера;
– перевіряти працездатність системи на рівні додатку;
– розуміти і помічати проблеми на рівні бізнес-логіки додатка.

Ці три шари дозволяють максимально забезпечити впевненість у тому, що з система працює саме так, як повинна [8].

Базовим рівнем є моніторинг серверних показників. Обов'язково варто стежити за навантаженням на CPU (створюючи оповіщення на Load Average, зростаючий більше ніж число ядер в системі, і зниження CPU idle до мінімальних значень), аналізувати статистику по навантаженню на дискову підсистему (збільшення часу avio, зростання дискових операцій iops, зростання «busy»

диска), стежити за використанням оперативної пам'яті, контролюючи значення «used memory», а на великих обсягах пам'яті – значення free. Треба стежити за використанням swap, причому не стільки за зміною swap used (додаток може перебувати в swap-е довгий час, але при цьому не використовуватися), скільки за числом swap free, щоб при вичерпанні процеси системи не знищувалися по out of memory, і, головне, swap in / swap out – безпосереднім використанням swap. Необхідно стежити за показниками ПО, що працює на сервері (статистикою БД, кешей, статистикою веб-сервера). Всі ці показники є базис від якого слід відштовхуватися, але яким не можна обмежуватися.

Наступним рівнем моніторингу є спостереження за технічними даними самого додатка. Сюди потрібно включити контроль за числом помилок додатки (5xx і 40x помилки в access-логах, аналіз логу помилок початковою програмою) – їх різке зростання може свідчити про серйозні проблеми. Практично завжди в access-логах будуть 404-е помилки, але якщо їх число стало занадто великим - можливо щось трапилося зі статичними даними. Це не завжди можна швидко виявити при візуальному перегляді сайту, а оповіщення на таку помилку дозволить зреагувати вчасно. Важливо збирати статистику по числу запитів до підсистем і вузлів системи – різко збільшилася кількість SQL запитів до бази може означати погану викладку коду, падіння запитів до API зовнішнього сервісу (який постійно повинен використовуватися) можливу аварію [9].

Варто також стежити і за часом доступу до зовнішніх сервісів – падіння зовнішнього API є досить частою і важко виявляємою помилкою, яку при моніторингу часу взаємодії стає досить просто локалізувати. Падіння числа відправлених листів може означати проблеми з поштою (перестала відправлятися). І, навпаки, різке зростання відправлених листів може виявитися наслідком того, що з сервера почали розсилати спам.

Систему варто перевіряти і зовні. Необхідно стежити не тільки за часом відповіді і кодом відповіді сторінок сайту, але також і за їх розміром – досить часто в разі помилки сторінки сайту продовжують видавати код відповіді HTTP 200, і це дуже важка ситуація. Пошуковий движок сприйме таку сторінку як

актуальну і проіндексує її без будь-яких даних. Це не єдиний випадок, коли така метрика може стати в нагоді. Дуже важливо збирати метрики і за статистикою резервного копіювання. Таких метрик можна придумати ще досить багато, але для повного контролю необхідний третій шар захисту.

Навіть якщо додаток буде працювати добре, навіть якщо на сервері не буде проблем – падіння зовнішнього білінгу призведе до неможливості провести платіж, відсутності замовлень і втрати грошей. Контроль бізнес-метрик є найважливішою і часто не береться частиною повноцінного моніторингу. Почати такий контроль можна з найпростішого збору часу, що пройшов з останнього замовлення. Якщо замовлень не було дуже давно - це не означає проблему на самому проєкті, проте може означати, наприклад, зупинену рекламну кампанію або проблеми в SEO-трафіку. Подібні проблеми часто можна виявити вже через дуже довгий час, коли виправити їх майже неможливо. Для запобігання проблем рекомендується періодично емулювати дії користувача на сайті (наприклад, вхід на сайт, виконання певних дій на сайті), щоб мати додаткову впевненість. Навіть якщо помилки проєкту будуть пропущені на перших двох рівнях моніторингу, то спрацюють оповіщення про зміни в бізнес-процесах проєкту, які можна буде розслідувати.

Корисно моніторити час з останнього продажу, кількість продажів за період. Якщо виклали реліз, то що змінилося: чи є просадки по бізнес-показниками? На це відповідає, звичайно, A / B тестування, але графіки теж хотілося б мати. І треба моніторити дії кінцевого користувача: писати скрипти на phantomjs, які повторюють покупку, проходять по всіх етапах основного бізнес-процесу [10].

Також вам, напевно, цікаво знати, чи працює сервіс логістики, або не впав чи в черговий раз IpGeoBase.

Потрібно розуміти, залежить чи просадка по бізнес-показниками від вашої системи, або від зовнішньої.

Сама система моніторингу повинна також відстежуватися. Повинен бути якийсь зовнішній кастомний скрипт, який буде перевіряти, що моніторинг

працює нормально. Нікому не хочеться прокинутися від дзвінка, тому що ваша система моніторингу впала разом з усім дата-центром, і вам про це ніхто не сказав [11].

3.2 Аналітичний функціонал системи моніторингу в задачах оптимізації ресурсів та прогнозування

Система моніторингу не повинна обмежуватися тільки інформуванням про проблеми. При досягненні критичних значень важливо зрозуміти – в перший чи раз така проблема трапилася або вона системна? Чи може система продовжити працювати в такому стані, як скоро відбудеться сама аварія? Для цього система моніторингу повинна містити достатню кількість даних, які можна проаналізувати, адекватно і швидко відобразити з максимальною деталізацією і мати можливість вибрати потрібний проміжок часу.

Добре побудована система моніторингу повинна давати можливість порівняти однотипні дані по різних серверах системи, мати можливість провести порівняння поточних метрик з історичними (день назад, тиждень тому, місяць тому тощо), для того щоб зрозуміти наскільки поточні метрики відхиляються від норми.

В ідеальному випадку система моніторингу як засіб аналізу повинна надавати можливість складної статистичної агрегації метрик – як мінімум обчислення ковзної середньої, перцентилю, угруповання серії даних по їх мінімуму / максимуму і інші значення.

Дані в моніторингу слід використовувати не тільки для аналізу причин поточних аварій, а й для розуміння розвитку проєкту. Регулярний аналіз зростання навантаження, відвідуваності, взаємодії систем дозволить зрозуміти межа зростання проєкту і, виходячи з цього, спланувати його подальший розвиток: плани на розробку, плани зі зміни архітектури тощо. Таким чином в моніторинг не варто заходити лише в разі аварії. Регулярний аудит збираються

метрик повинен стати основою для регулярного аудиту всієї системи і оцінки перспектив її розвитку [12].

4 АНАЛІЗ РЕЗУЛЬТАТІВ ОПТИМІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ

Практичне впровадження системи моніторингу в інфраструктуру підприємства з великою кількістю контрольованих об’єктів вимагає оцінювання її ефективності та навантаження на обчислювальні ресурси сервера. Метою цього розділу є проведення порівняльного аналізу функціонування розгорнутої системи Zabbix у двох конфігураціях: базовій (з використанням стандартних шаблонів та налаштувань за замовчуванням) та оптимізованій (із застосуванням кастомних шаблонів, механізмів фільтрації низькорівневого виявлення, оптимізації періодів опитування тощо).

Експериментальне дослідження проводилося для мережної інфраструктури, що складається зі 150 вузлів мережі: 100 одиниць телекомунікаційного обладнання Cisco/MikroTik, 30 серверних систем під управлінням ОС Linux/Windows та 20 віддалених мережних сервісів.

4.1 Метрики оцінювання ефективності та методика тестування

Для аналізу продуктивності та ресурсомісткості системи моніторингу було визначено такі ключові показники:

– NVPS (New Values Per Second) – середня кількість метрик, що надходять і обробляються сервером моніторингу за одну секунду. Цей параметр безпосередньо визначає навантаження на процесор і базу даних.

– Навантаження на CPU (CPU Usage, %) – сумарна утилізація процесорних потужностей процесами ядра Zabbix (history syncer, poller, trapper) та СКБД MySQL.

– Використання оперативної пам'яті (RAM Usage, ГБ) – об'єм ОЗП, що споживається демонами системи та буферами бази даних.

– Швидкість росту бази даних (DB Growth Rate, ГБ/місяць) – обсяг дискового простору, який займають історичні дані та тренди.

– Дискові операції введення-виведення (I/O Wait, %) – відсоток часу, протягом якого процесор очікує завершення операцій запису/читання з накопичувача.

Дослідження проводилося у два етапи, кожен тривалістю 72 години, для забезпечення релевантності накопиченої статистики та виключення випадкових флуктуацій трафіку.

4.2 Порівняльний аналіз споживання ресурсів до та після оптимізації

При первинному розгортанні системи моніторингу із застосуванням стандартних шаблонів (Linux by Zabbix agent, Generic SNMP by SNMP) для 150 хостів загальна кількість елементів даних (Items) склала приблизно 18 500 одиниць, а кількість тригерів — 4 200. Така надмірність зумовлена тим, що базові шаблони SNMP опитують усі доступні порти комутаторів (включаючи неактивні або сервісні інтерфейси vlan/loopback), а інтервали перевірок за замовчуванням становлять 60 секунд для більшості метрик.

Після проведення комплексної оптимізації було реалізовано такі кроки:

– модифіковано правила LLD (Low-Level Discovery) із впровадженням регулярних виразів для ігнорування вимкнених портів;

– глибока кастомізація та рефакторинг шаблонів;

– диференціація часових інтервалів опитування;

– перехід на активний режим роботи агентів;

– оптимізація процесів синхронізації історії;

– збільшено інтервали опитування для умовно-статичних параметрів (об'єм дисків, інвентарні дані, версії ПЗ) з 5 хвилин до 4 годин;

– налаштовано внутрішні кеші Zabbix Server;

– реструктуризація та оптимізація підсистеми збереження даних;

– впровадження секціонування таблиць баз даних;

– період зберігання сирої історії (History) зменшено з 90 до 7 днів із паралельним увімкненням агрегованих трендів (Trends) на 365 днів.

Результати вимірювань інтенсивності збору даних та системних ресурсів наведено у таблиці 4.1.

Таблиця 4.1 – Порівняльні технічні показники роботи системи моніторингу

Параметр дослідження	Базова конфігурація	Оптимізована конфігурація	Абсолютна економія	Ефективність оптимізації, %
Загальна кількість елементів даних	18500	6200	12300	66,5%
Показник продуктивності (NVPS)	310	65	245	79%
Завантаження CPU (середнє), %	38,5%	12,2%	26,3%	68,3%
Використання RAM, ГБ	5,4 ГБ	2,8 ГБ	2,6 ГБ	48,1%
Показник дискового I/O Wait, %	14,2%	1,8%	12,4%	87,3%
Швидкість росту БД, ГБ/місяць	42,0 ГБ	6,5 ГБ	35,5 ГБ	84,5%

З отриманих експериментальних даних видно, оптимізація шаблонів та фільтрація непотрібних інтерфейсів дозволили скоротити кількість контрольованих елементів даних на 66,5%. Це безпосередньо вплинуло на метрику NVPS, яка знизилася з 310 до 65 значень за секунду (економія обчислювальної потужності становить 79%).

Найбільш критичний інженерний результат досягнуто у підсистемі збереження даних. Завдяки оптимізації термінів зберігання первинної історії та переходу на роботу з трендами, швидкість заповнення дискового простору базою даних MySQL знизилася з 42 ГБ до 6,5 ГБ на місяць. Це зменшує сукупну вартість володіння системою та вимоги до накопичувачів.

Показник I/O Wait знизився з небезпечних 14,2% (що часто викликає затримки в чергах Zabbix) до стабільних 1,8%, що гарантує високу відмово стійкість системи моніторингу при подальшому масштабуванні мережі.

4.3 Оцінювання масштабованості системи моніторингу

На основі отриманих даних оптимізованого профілю було проведено математичне прогнозування масштабованості системи. Розрахунок граничного навантаження на апаратну платформу (4 ядра CPU, 8 ГБ RAM, SSD-накопичувач) показує, що конфігурація за замовчуванням почне деградувати (черги моніторингу виростуть, показник I/O Wait перевищить 50%) при досягненні ліміту в ~500 хостів.

Оптимізована архітектура, завдяки низькому показнику NVPS (65 одиниць на 150 хостів), дозволяє без капітальної модернізації серверного обладнання розширити мережу моніторингу до 2500–3000 вузлів без втрати швидкодії та затримок у генерації інцидентів.

Експериментально доведено, що перехід від стандартних конфігурацій до кастомних шаблонів та оптимізованих інтервалів опитування дозволив знизити потік вхідних метрик (NVPS) на 79%, що розвантажило центральний процесор сервера з 38,5% до 12,2%.

Оптимізація політик зберігання історичних даних (History/Trends) забезпечила колосальне зниження дискового навантаження (I/O Wait зменшився на 87,3%), а швидкість росту бази даних сповільнилася на 84,5% (з 42 ГБ/міс до 6,5 ГБ/міс).

Впроваджені методи оптимізації підвищили стелю масштабованості системи моніторингу в 5-6 разів, забезпечуючи стабільну детекцію інцидентів у реальному часі без необхідності залучення додаткових апаратних чи фінансових ресурсів підприємства.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, що полягає в дослідженні, проєктуванні, розгортанні та оптимізації комплексної системи моніторингу сервісів комп'ютерної мережі та мережного обладнання для інфраструктури підприємства.

Проведено системний аналіз сучасних архітектурних рішень та критеріїв класифікації систем моніторингу IT-інфраструктури. На основі порівняльного аналізу платформ Nagios, Prometheus та Zabbix науково обґрунтовано вибір системи Zabbix як базового інструментарію проєктування. Доведено, що Zabbix є найбільш універсальним рішенням типу All-in-One, яке «з коробки» забезпечує повноцінний безагентний контроль мережного обладнання за протоколом SNMP, володіє потужним вбудованим графічним інтерфейсом керування та відкритим вихідним кодом, що нівелює ліцензійні витрати та знижує сукупну вартість володіння (TCO).

Розроблено методику та реалізовано розгортання серверної частини системи моніторингу на базі операційної системи Ubuntu Server у віртуалізованому середовищі. З метою мінімізації надлишковості системного оточення, підвищення кібербезпеки та забезпечення повного контролю за залежностями компонентів, інсталяцію ядра Zabbix Server здійснено шляхом ручної компіляції безпосередньо з офіційних вихідних кодів (Source codes). Як підсистему збереження даних інтегровано високопродуктивну реляційну СКБД MySQL.

Виконано глибоке конфігурування розробленої системи для гетерогенного мережного середовища масштабом у 150 вузлів (комутатори, маршрутизатори, сервери та веб сервіси). Сформовано логічну структуру вузлів, впроваджено механізми автоматизації менеджменту за допомогою концепції шаблонізації та спадковості конфігурацій, а також налаштовано логічні вирази тригерів для автоматичної детекції аномалій. Створено підсистему автономного

проактивного сповіщення технічного персоналу про інциденти через сучасні канали зв’язку (API Telegram/Slack, SMTP, GSM) та реалізовано алгоритми автоматичного відновлення сервісів за допомогою віддалених команд (Remote commands).

Розроблено та впроваджено комплексний метод оптимізації системи моніторингу на архітектурному рівні, рівні ядра та СКБД. Основними заходами оптимізації стали: рефакторинг та кастомізація базових шаблонів, інтеграція регулярних виразів для фільтрації інтерфейсів у правилах низькорівневого виявлення (LLD), диференціація інтервалів опитування, перехід на активний режим роботи агентів (Zabbix Agent Active), оптимізація внутрішніх RAM-кешів сервера та впровадження горизонтального секціонування (Table Partitioning) історичних таблиць СКБД із переходом на агреговані тренди (Trends).

Експериментально доведено високу ефективність реалізованих заходів оптимізації й обґрунтовано потенціал масштабованості спроєктованої системи. Завдяки суттєвому зниженню споживання ресурсів, розроблена архітектура здатна забезпечити стабільний моніторинг інфраструктури обсягом до 2500–3000 вузлів мережі в реальному часі на поточному серверному обладнанні без втрати швидкодії чи виникнення затримок (черг) обробки інцидентів.

Таким чином, розроблений та оптимізована система моніторингу повністю відповідає сучасним інженерним вимогам, мінімізує вплив людського фактору на виявлення мережних аварій і може бути безпосередньо впроваджений у практику адміністрування телекомунікаційних мереж або інфраструктур реальних підприємств.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zabbix: краще бути гнучким [Електронний ресурс] – Режим доступу до ресурсу https://itco.blog/zabbix-free-and-flexible.html – 06.03.2018 р. – Загол. з екрану.

2. Порівняння систем моніторингу Zabbix vs Nagios [Електронний ресурс] – Режим доступу до ресурсу http://www.netping.ru/Blog/sravnenie-sistem-monitoringa-zabbix-vs-nagios – 31.10.2018 р. – Загол. з екрану.

3. Zabbix documentation [Zabbix Documentation 5.2] [Електронний ресурс] – Режим доступу до ресурсу https://www.zabbix.com/documentation/current/ru/start – 01.09.2020 р. – Загол. з екрану.

4. Ubuntu 20.04 Server Installation [Електронний ресурс] – Режим доступу до ресурсу https://linuxconfig.org/ubuntu-20-04-server-installation – 16.05.2019 р. – Загол. з екрану.

5. Як підвищити безпеку Nginx за допомогою Let's Encrypt в Ubuntu 18.04 | DigitalOcean [Електронний ресурс] – Режим доступу до ресурсу https://www.digitalocean.com/community/tutorials/nginx-let-s-encrypt-ubuntu-18-04-ru – 06.05.2020 р. – Загол. з екрану.

6. Конфігурація my.cnf для 8 Гб [Електронний ресурс] – Режим доступу до ресурсу https://ruhighload.com/mycnfexample?ram=8 – 02.11.2019 р. – Загол. з екрану.

7. Налаштування оповіщень Zabbix [Електронний ресурс] – Режим доступу до ресурсу https://serveradmin.ru/nastroyka-opoveshheniy-zabbix-v-telegram/ – 06.11.2020 р. – Загол. з екрану.

8. Правильне виявлення проблем за допомогою Zabbix / Блог компанії Конференції Олега Буніна (Онтіко) [Електронний ресурс] – Режим доступу до ресурсу https://habr.com/ru/company/oleg-bunin/blog/320678/ – 28.01.2017 р. – Загол. з екрану.

9. Zabbix для DevOps: як ми впроваджували систему моніторингу у процеси розробки та тестування / Блог компанії Positive Technologies [Електронний ресурс] – Режим доступу до ресурсу https://habr.com/ru/company/pt/blog/325276/ – 30.03.2017 р. – Загол. з екрану.

10. Універсальна система моніторингу Zabbix – введення [Електронний ресурс] – Режим доступу до ресурсу https://habr.com/ru/post/73338/ – 25.10.2009 р. – Загол. з екрану.

11. Організація системи моніторингу / Блог компанії ITSumma [Електронний ресурс] – Режим доступу до ресурсу https://habr.com/ru/company/itsumma/blog/350200/ – 01.03.2018 р. – Загол. з екрану.

12. Моніторинг обладнання. Системи та програми спостереження, моніторингу мережного обладнання. [Електронний ресурс] – Режим доступу до ресурсу https://alp-itsm.ru/interesting/monitoring_setevogo_oborudovaniya/ – 18.05.2016 р. – Загол. з екрану.