Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

СИСТЕМА МОНІТОРИНГУ МЕРЕЖІ ПІДПРИЄМСТВА

Текстова частина до курсової роботи за спеціальністю „Комп'ютерні науки" 122

Керівник курсової роботи
к.т.н., доц. Черкасов Д.І.
_____
(підпис)
"____" _____ 2020 р.

Виконав студент
Шекета К.О.
"____" _____ 2020 р.

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	15.10.2019	
2.	Огляд наявних рішень для системи моніторингу підприємства.	30.10.2019	
3.	Аналіз особливостей наявних рішень.	15.02.2020	
4.	Написання першого розділу	20.03.2020	
5.	Написання другого розділу.	25.03.2020	
6.	Написання останнього розділу.	23.04.2020	
7.	Створення презентації та написання доповіді для захисту роботи.	05.05.2020	
8.	Корегування роботи згідно із зауваженнями керівника	09.05.2020	

Зміст

Вступ

Функціювання сучасного підприємства базується на складній IT інфраструктурі, пов’язане з обробкою значного об’єму даних та великої кількісті запитів до використовуваних інформаційних сервісів. Для цього їхня інформаційна структура перейшла від монолітної архітектури до архітектури мікросервісів, яка розбиває систему на компоненти, кожен з яких вирішує чітко поставлену задачу і взаємодіє з іншими компонентами.

Таким компаніям дуже важливо, щоб усі послуги і можливості системи працювали безвідмовно. Якщо стан системи буде невідомий, то існує можливість не помітити критичний збій, який матиме негативні наслідки для системи і фінансового становища компанії. Тому виникає необхідність слідкувати за справністю компонентів системи і каналів зв’язку між ними, щоб мати можливість завчасно попереджати можливі проблеми.

Для цього існують системи моніторингу, основна задача яких полягає в інформуванні про втрату працездатності мережі та серверів, відхилення ключових метрик від базової норми. Така система також збирає, зберігає та надає візуалізацію даних, що відображають тренди в довготривалих змінах ключових параметрів системи.

На даний момент запропоновано багато систем моніторингу, які мають різний підхід у реалізації базових функцій, наприклад, у якому вигляді будуть зберігатися дані про стан системи. Або пропонують додаткові функції, як автоматичне додавання нового компоненту під моніторинг.

Основна задача курсової роботи полягає в тому, щоб на основі особливостей інших систем запропонувати своє рішення для системи моніторингу.

Ця задача поділяється на підзадачі:

1. Оглянути існуючі рішення, виконати порівняльний аналіз, виділити особливості кожного рішення.
2. Розробити структурну архітектуру свого рішення
3. Детально розробити один з компонентів системи

Розділ 1: Порівняльний аналіз систем моніторингу

Перед тим, як порівнювати наявні рішення системи моніторингу, перелічимо основні її функції:

- Сповіщення («alerting») про будь-які проблеми чи відхилення ключових метрик від заданої норми. Має здійснюватися різними способами, наприклад, через e-mail.
- Здатність до моніторингу різних типів об’єктів і подій. Система має бути здатна надавати необхідну інформацію про стан потрібних об’єктів. Наприклад, сервери, компоненти мережі тощо.
- Візуалізація даних через інтерфейс користувача (“dashboards”).
- Звітування (“reporting”) результатів моніторингу. Відбувається із заданою періодичністю, або по запиту.
- Ключова задача системи моніторингу є отримання, збереження та аналіз інформації про стан об’єктів, які знаходяться під моніторингом, визначення трендів в змінах значень ключових метрик.

Наразі існує низка рішень систем моніторингу, кожна з яких пропонує свій підхід у реалізації вище наведених функцій.

Очевидно, що популярною і ефективною систему моніторингу робить її гнучкість: здатність розширюватися за допомогою плагінів, автоматичне виявлення нових об’єктів в інфраструктурі, сумісність з різною кількістю серверів та застосунків, а також надання можливості для модифікацій та вибір між агентним та безагентним моніторингом.

Процес конфігурації системи моніторингу повинен бути простим.

Розглядатися будуть найпопулярніші на ринку системи моніторингу: Nagios, Zabbix та PandoraFMS.

Аналізуватися буде реалізація основних функцій кожною системою, архітектура систем, зручність інтерфейсу та простота конфігурації.

1.1. Огляд системи моніторингу Nagios

Nagios, також відомий як Nagios Core, безкоштовне програмне забезпечення з відкритим вихідним кодом, який розроблений під Unix системи. До нього також існує платне розширення Nagios XI.

На ринку з 2002 року і за весь час здобув широку популярність як проста і потужна система моніторингу.

Архітектура

Nagios має клієнт-серверну архітектуру. Принцип його роботи полягає в тому, що Nagios сервер працює на гості, а плагіни - на віддалених гостах, над якими здійснюється моніторинг.

Плагіни виконують тести та обробляють результат. Статус цих тестів може мати один з чотирьох станів: OK, WARNING, CRITICAL або UNKNOWN.

Планувальник процесів (“Process scheduler”) надсилає сигнал на виконання плагінів на віддаленому гості. Плагін отримує статус з віддаленого гості і надсилає її планувальнику процесів, який оновлює веб-інтерфейс.

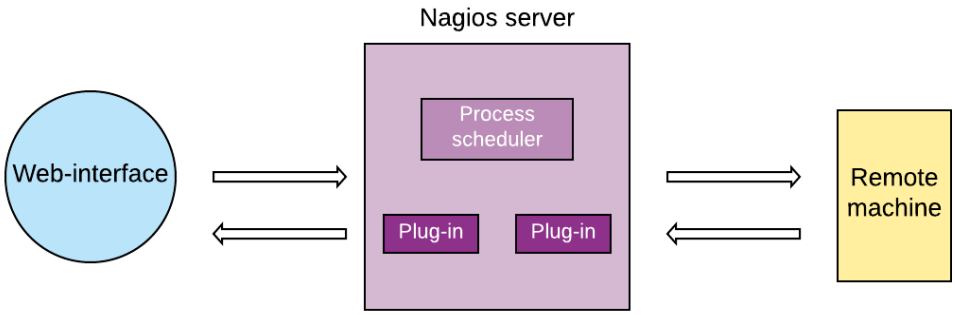

Рис. 1.1.1.1 - Схема архітектури Nagios

Щоб над машиною здійснювався моніторинг, на ній має бути встановлено плагін залежно від її операційної системи: NSClient++ для Windows, NRPE для Linux та NSCA для Mac OS X. До серверу під’єднана база даних, в якій зберігаються лог-файл.

Інтерфейс користувача

Інтерфейс в Nagios XI дає можливість різним користувачам налаштовувати інтерфейс таким чином, щоб була видна лише їм потрібна інформація. Можна також змінювати конфігурацію моніторингу.

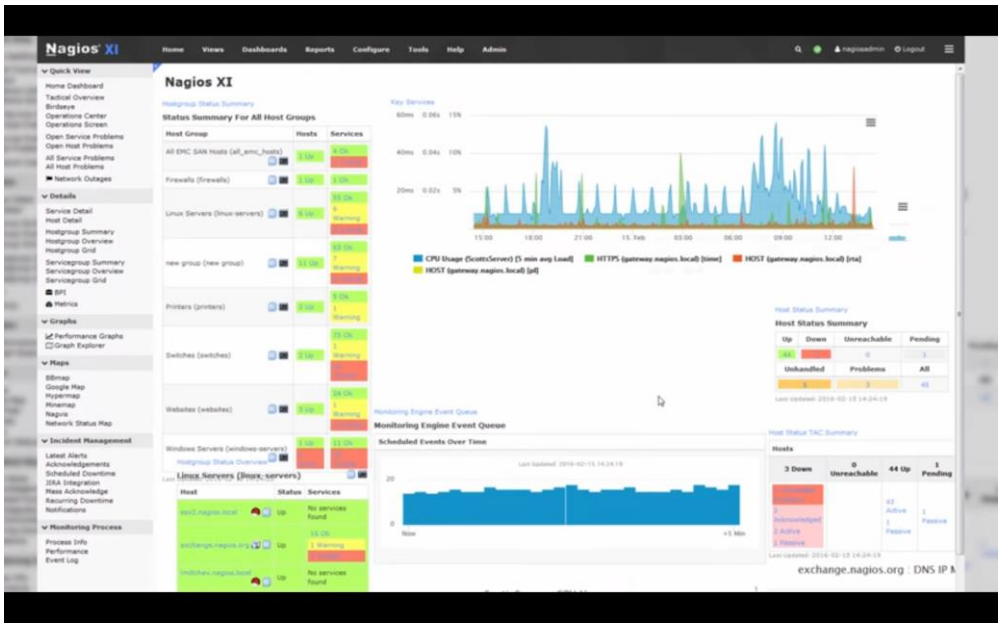

Рис. 1.1.2.1- Інтерфейс Nagios

1.2. Огляд системи моніторингу Pandora FMS

Pandora FMS (*Pandora Flexible Monitoring System*) безкоштовна система моніторингу з відкритим вихідним кодом.

Архітектура

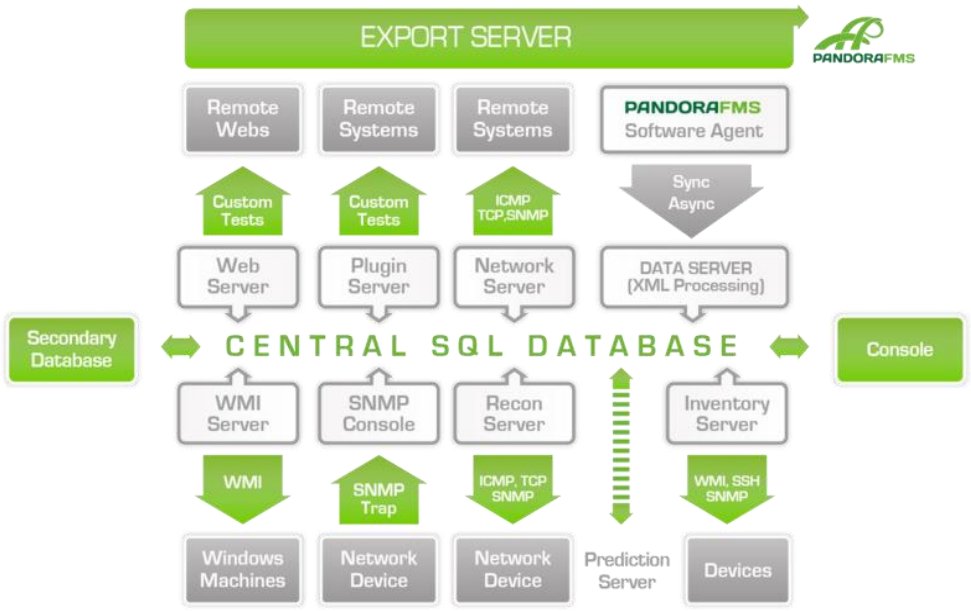

Рис. 1.2.1.1 - Схема архітектури PandoraFMS

Pandora FMS складається з кількох компонентів. За збір і обробку інформації відповідають сервери. Сервери згенеровану ними ж інформацію або ту, що зібрали агенти, заносять у базу даних, яку потім відображає консоль.

Загалом наявні 9 серверів, кожен з яких виконує конкретну задачу.

- Data Server

Обробляє надіслані агентами XML-пакети і зберігає інформацію до БД. На основі цієї інформації генерує сповіщення і системні події.

- Network Server

Відповідає за віддалений моніторинг за допомогою ICMP, TCP та SNMP.

- SNMP Console та WMI Server

Отримують інформацію відповідно з SNMP Traps та ОС Windows.

- Recon Server

Виявляє нові системи в мережі, а також виявлення ОС за допомогою nmap, xprobe та traceroute.

- Plugin Server

Виконує віддалені перевірки за допомогою власне створених скриптів.

- Prediction Server

Будує передбачення елементу даних через 10-15 хвилинні інтервали, яке будується на основі даних давністю до 30 днів.

- Export Server

Відповідає за перенесення даних з одного об’єкту моніторингу на інший.

- Inventory Server

Збирає та відображає інформацію про встановлене ПЗ, обладнання, пам’ять, жорсткі диски, сервіси, що використовуються в системі тощо. Отримує інформацію від агентів ПЗ.

Інтерфейс користувача

Написаний на PHP. Надає можливість визначати нових агентів, мережевих модулів, сповіщень, а також створення звітів і зміна конфігурації моніторингу. Консоль за замовчуванням показує статус серверу, сповіщень та агентів. Надається можливість кастомізації.

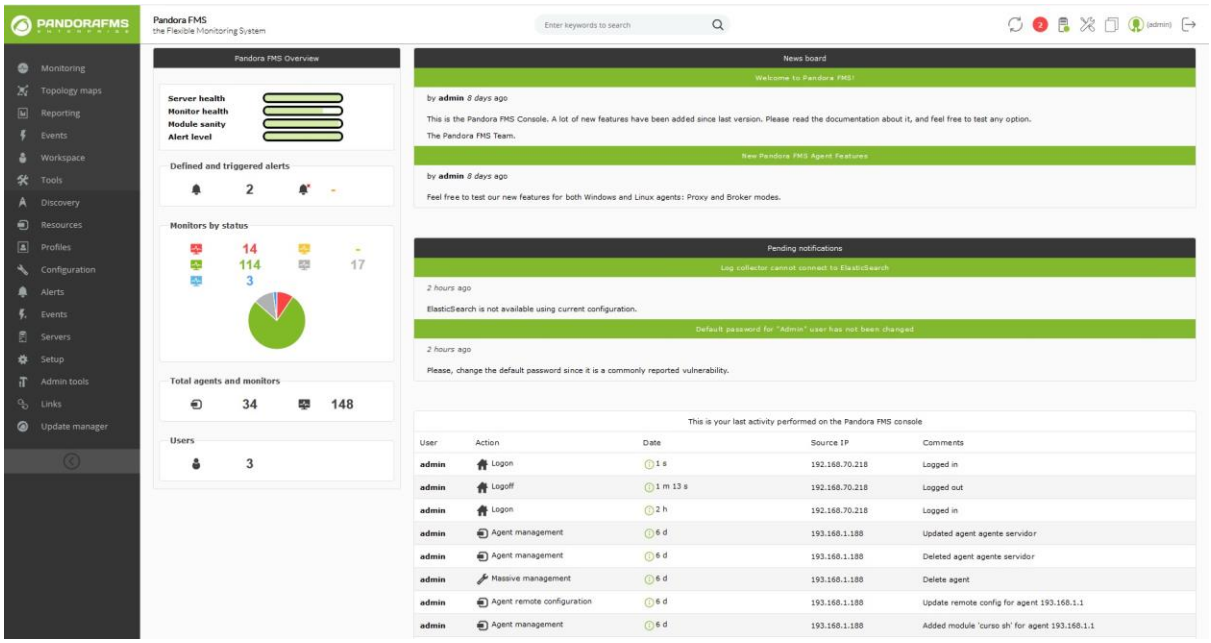

Рис. 1.2.2.1 - Інтерфейс користувача в PandoraFMS

1.3. Огляд системи моніторингу Zabbix

Zabbix - система моніторингу з відкритим вихідним кодом. Є повністю безкоштовна.

Архітектура

Zabbix можна визначити як розподілену систему з централізованим веб-інтерфейсом. Складається з трьох основних компонентів: Zabbix-серверу, RDBMS-серверу та web-серверу. Для великої інфраструктури виділяються також такі компоненти як Zabbix-агенти та Zabbix-проксі.

Zabbix-сервер збирає інформацію від Zabbix-проксі, які у свою чергу збирають інформацію від з'єднаних з ними Zabbix-агентів. Уся зібрана інформація зберігається у відповідній РСКБД.

Веб-інтерфейс написаний на PHP, а сервер, проксі та агенти на C.

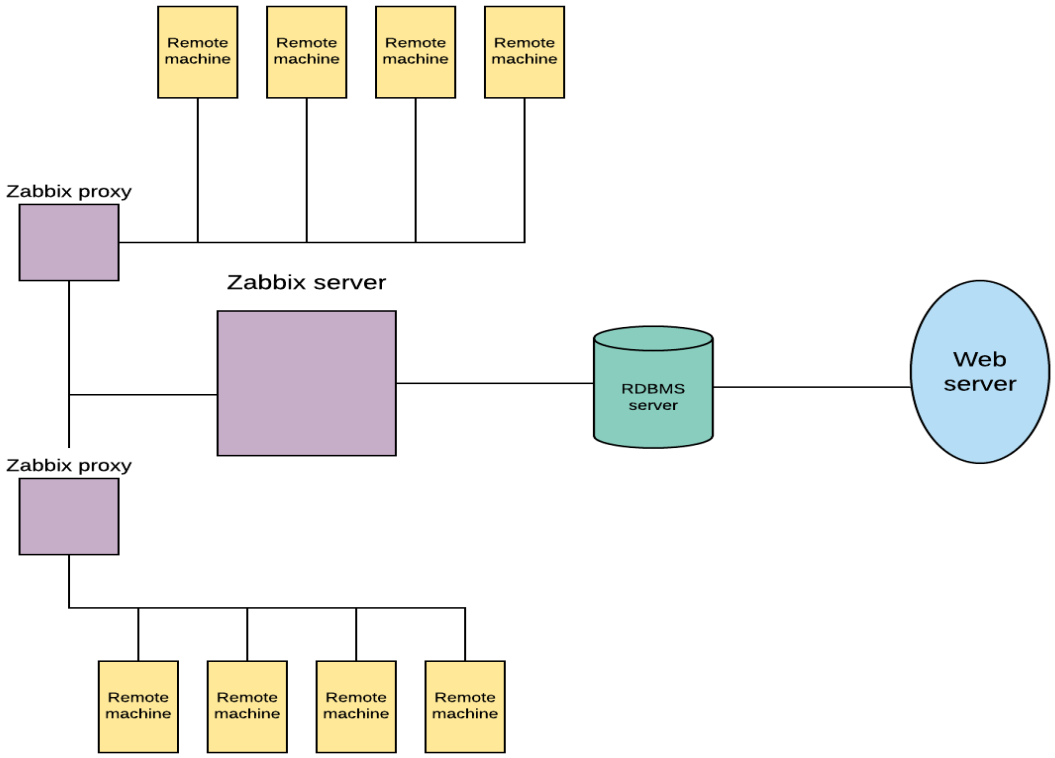

Рис. 1.3.1.1 Схема архітектури Zabbix

Інтерфейс користувача

Інтерфейс Zabbix дає можливість відображати будь-яку кількість графіків, які оновлюються автоматично. Можна задати період оновлень. Перевагою інтерфейсу є можливість змінювати конфігурацію моніторингу та перегляд усіх даних. Є вкладка меню, де можна задати прийнятний поріг для ключових метрик, у разі перевищення якого змінює стан з OK на PROBLEM.

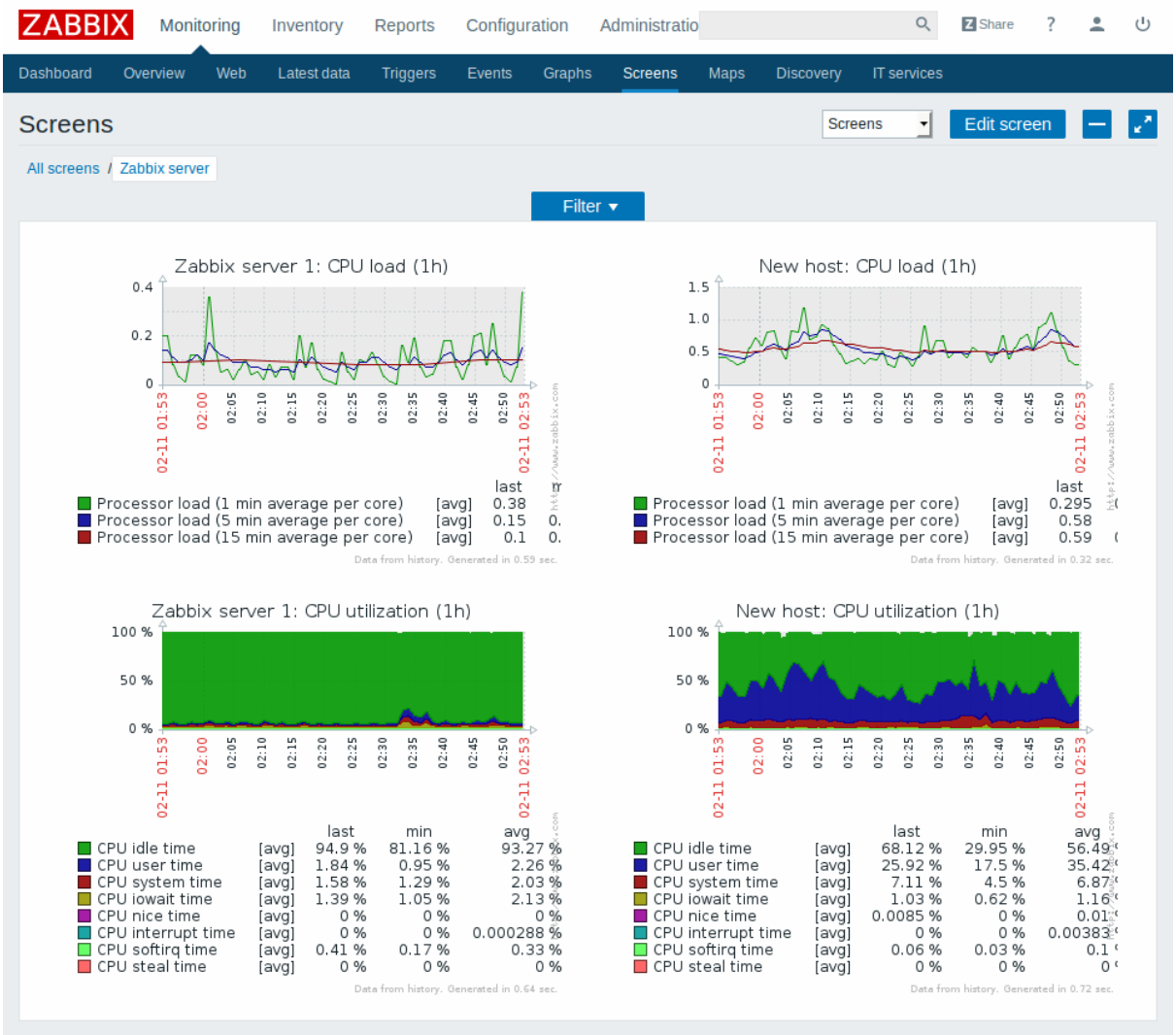

Рис. 1.3.2.1 - Інтерфейс користувача Zabbix

1.4. Порівняльний аналіз

Можливість безагентного моніторингу

Nagios та Zabbix надають можливість для безагентного моніторингу.

Nagios реалізовує це за допомогою технологій WMI та SNMP. SNMP підтримується багатьма різними мережевими пристроями та операційними системами, а WMI відповідає за моніторинг операційної системи Windows – метрики операційної системи, стан сервісу, використання файлової системи тощо. Для інтеграції цих технологій з Nagios надається документація на офіційному сайті. Можливо розширити можливості безагентного моніторингу за допомогою плагінів.

Zabbix так само підтримує безагентний моніторинг за допомогою SNMP traps, а також IPMI (працює лише з пристроями, які його

підтримують), SSH перевірки (вимагається конфігурація Zabbix серверу з SSH2), Telnet, ICMP пінгів.

PandoraFMS надає можливості для безагентного моніторингу з використанням SNMP, мережевих запитів і WMI.

Процес автовиявлення нових пристроїв

Усі три системи моніторингу надають автовиявлення нових об’єктів.

В Nagios це складається з двох компонентів: Auto-Discovery Jobs та Auto-Discovery Configuration Wizard. Перший компонент відповідає за пошук нових пристроїв в заданій адресі мережі і з заданою періодичністю пошуку нових пристроїв. У другому компоненті користувач обирає, які із знайдених пристроїв моніторити.

В Zabbix автовиявлення в мережі відбувається теж у два етапи: пошук пристроїв згідно із заданими правилами виявлення в мережі та ручного додавання пристрою під моніторинг. Можна задати автоматичні дії при виявленні нового пристрою. Zabbix сервер може автоматично почати моніторинг пристрою, на якому встановлений Zabbix агент.

У Pandora FMS її Recon Server виявляє пристрої в мережі за допомогою ICMP (Ping) згідно із наперед заданим Recon-task. Нові системи ідентифікуються своїми IP адресами, і потім вони відслідковуються та агрегуються як агенти. Згідно з "Plugin Templates", автоматично встановлюються необхідні мережеві модулі (TCP, SNMP, ICMP тощо). Таким чином нова виявлена система потрапляє під моніторинг.

Розширення плагінами

Nagios відомий у першу чергу своєю гнучкістю в питаннях встановлення плагінів. При звичайній установці на ньому є стандартний набір плагінів написаних на Perl для збирання основної інформації (навантаження CPU, обсяг вільної оперативної пам’яті, температура в корпусі тощо) як з windows-серверів так і unix.

Стандартні плагіни можна розширити самостійно або використати свої власне створені плагіни, або інших користувачів. Власне створені плагіни можуть бути написані будь-якою мовою програмування. Встановлення плагіну реалізовано через інтерфейс користувача. Після, необхідно здійснити конфігурацію нової команди. Таким чином

різноманіття плагінів і простота їхнього встановлення дозволяють Nagios здійснювати моніторинг будь-чого.

Розширення можливостей агента в Zabbix можливе трьома шляхами: написанням нових модулів на C, створення скриптів та за допомогою system.run**.** Будь-які зміни в конфігурації агента вимагає його перезавантаження, а мова C хоч відома своєю швидкістю, складна для вивчення. З виходом Zabbix 4.4 презентували нове покоління Zabbix-агенту написаного на Go, і який спеціально розроблений таким чином, щоб його можна було легко розширювати плагінами написаних на Go. Також надаються офіційні фреймворки для створення плагінів.

Pandora FMS має бібліотеку з плагінами для інтеграції з іншими застосунками і модулі для моніторингу застосунків, мережі, ОС тощо. Плагіни можна написати будь-якою мовою програмування. Є обов'язкова вимога до плагінів для агентів: вони повинні надавати інформацію в XML-форматі.

Інтеграція з іншими застосунками

Усі три системи моніторингу дозволяють інтеграцію з іншими застосунками. Офіційний список сервісів з якими може інтегруватися Zabbix більший, аніж у Nagios, але гнучкість Nagios дозволяє йому інтегруватися з будь-яким сервісом, якщо написати плагін. У Nagios вже інтегрований AWS моніторинг, тоді як на Zabbix його потрібно встановити окремо. У Pandora FMS теж можливо здійснити інтеграцію з іншими сервісами шляхом встановлення плагінів.

Загалом, усі три системи надають можливість для інтеграції з популярними сервісами, такими як AWS, Oracle VM, Docker тощо.

Способи сповіщень

У Nagios XI пропонується два способи сповіщень: через email або SMS-повідомлення. У Pandora FMS так само, а Zabbix, крім цих стандартних способів, також має сповіщення за допомогою Jabber та Ez texting. Але ці списки можна розширити за допомогою плагінів. Наприклад, Pandora FMS має плагін для інтеграції з WhatsApp.

Розподілений моніторинг

Офіційна документація Nagios пропонує три рішення для розподіленого моніторингу: Mod-Gearman, Nagios Fusion, федеративний моніторинг.

Mod-Gearman - аддон для розподіленого моніторингу в Nagios. Реалізація полягає в тому, що один майстер-сервер, на якому міститься конфігурація та визначення перевірок, розподіляє роботу між робочими вузлами (англ. “worker nodes”), на яких відсутня конфігурація, але встановленні Nagios-плагіни.

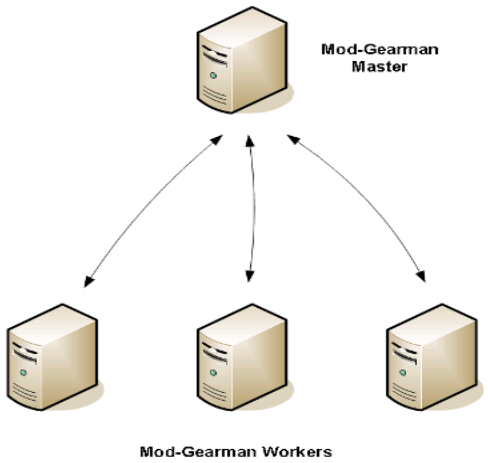

Рис. 1.4.6.1 - Схема принципу роботу Mod-Gearman для розподіленого моніторингу в Nagios XI

Nagios Fusion пропонує централізований інтерфейс для безлічі Nagios-серверів.

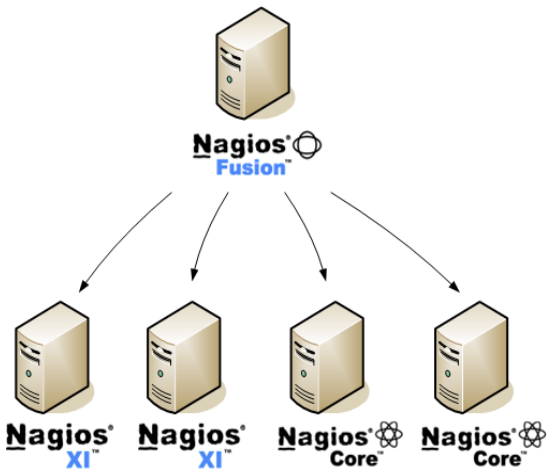

Рис. 1.4.6.2 - Схема принципу роботу Nagios Fusion для розподіленого моніторингу в Nagios

Федеративна архітектура моніторингу призначена для розподіленого моніторингу віддалених мереж. Головний Nagios-сервер в центрі керування мережі (англ. “network operations center”) отримує всі статуси перевірок віддалених Nagios-серверів.

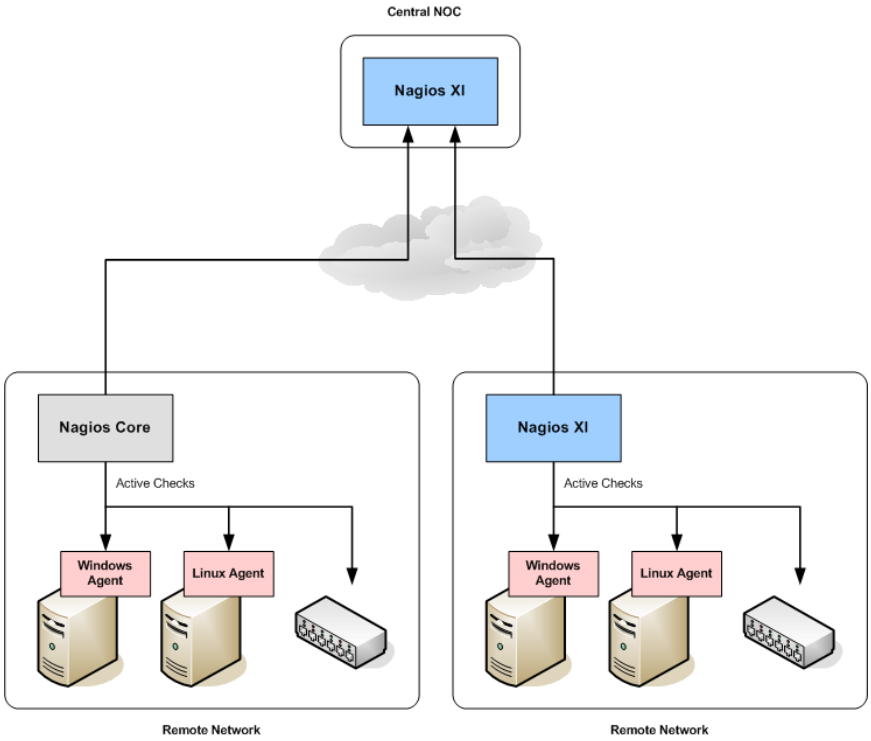

Рис. 1.4.6.3 - Схема федеративного моніторингу для розподіленого моніторингу в Nagios XI

Архітектура Zabbix дозволяє реалізувати розподілений моніторинг за допомогою Zabbix-проксі. Zabbix-проксі керуються головним сервером і спрямовані на збір та збереження інформації. Ними легко розширювати розподілену архітектуру, адже самі вони не виконують складних обчислень і запитів. У разі великої кількості об’єктів моніторингу, є сенс поділити ці об'єкти на групи і кожній групі призначити свій проксі.

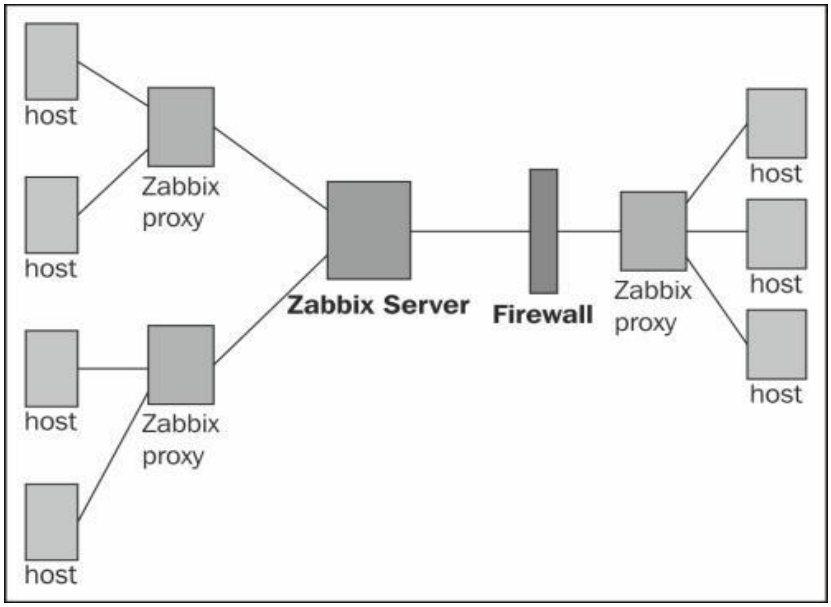

Рис. 1.4.6.4 - Схема розподіленого моніторингу в Zabbix

В PandoraFMS залежно від потреб, існує кілька підходів для реалізації розподіленого моніторингу.

- Broker-режим в агентів

При активації режиму broker, в агента створюється ще один конфігураційний файл, який керує брокером незалежно від цього агента. Це надає можливість надсилати дані так, ніби на машині стоять два різні повноцінні агенти, які здійснюють віддалені перевірки за допомогою мережевих протоколів.

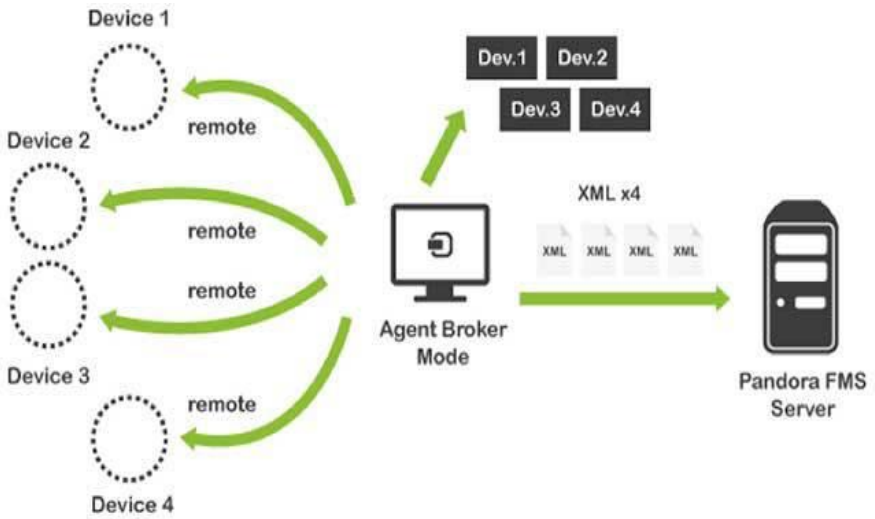

Рис. 1.4.6.5 - Схема принципу роботи broker-mode для розподіленого моніторингу в PandoraFMS

- proxy-режим

Підходить для моніторингу підмереж, до яких головний сервер не має доступу. Агенти, що встановлені на машинах висилають xml-пакети до proxy-агента, який потім пересилає їх до головного сервера. Передача цих файлів відбувається за допомогою Tentacle, протоколу PandoraFMS для передачі даних між агентом та сервером з визначеним режимом (як proxy-режим).

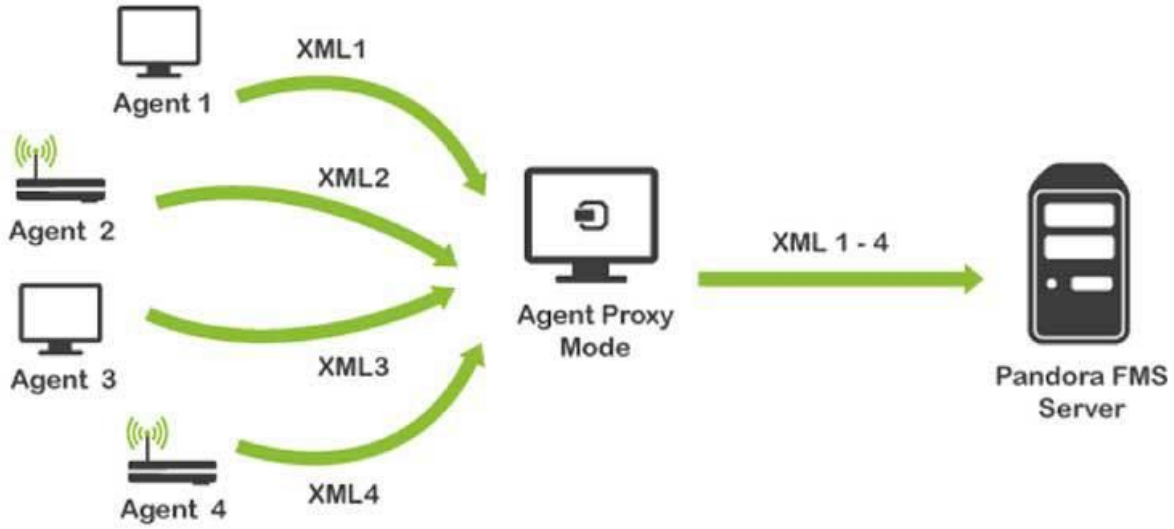

Рис. 1.4.6.6 - Схема принципу роботи proxy-mode для розподіленого моніторингу в PandoraFMS

- Встановлення додаткових PandoraFMS-серверів

Якщо один PandoraFMS-сервер не може впоратися з навантаженням, то можна встановити додатково ще сервери, які будуть під'єднані до спільної бази даних.

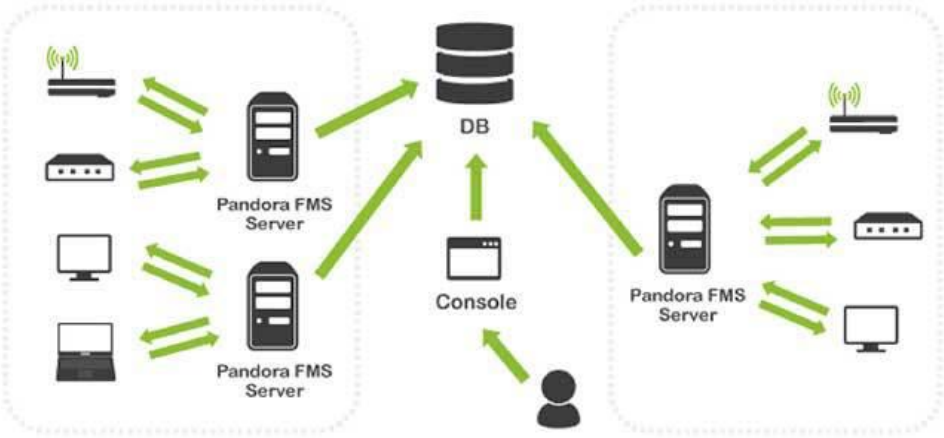

Рис. 1.4.6.8 - Схема для розподіленого моніторингу в PandoraFMS збільшенням кількості серверів

- Satellite server

Здійснює віддалені перевірки за допомогою ICMP, WMI, SNMP, а також отримує xml-пакети від агентів через Tentacle. Так само через Tentacle пересилає ці пакети до головного серверу. На відміну від встановлення звичайних серверів, не вимагає підключення до бази даних, і на відміну від proxy-режиму, самостійно здійснює віддалені перевірки.

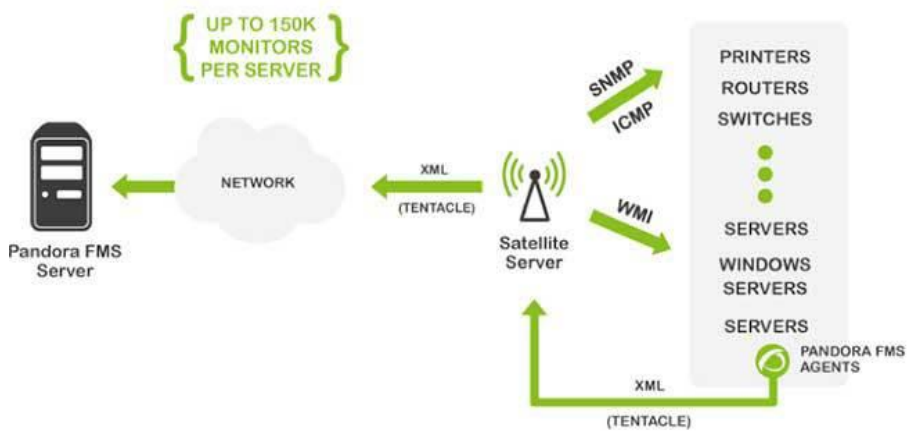

Рис. 1.4.6.9 - Схема для розподіленого моніторингу в PandoraFMS за допомогою Satellite Server

Способи збереження даних

За замовчуванням, в Nagios усі дані (файли із статусом серверів, часу простою (англ. “downtime”), файли з коментарями серверів та гостів) зберігаються в чистих текстових файлах. Nagios підтримує дві бази даних: MySQL та PostgreSQL, які можна інтегрувати за допомогою конфігураційних скриптів.

База даних для Zabbix створюється під час встановлення Zabbix. Zabbix підтримує такі бази даних: MySQL, PostgreSQL, TimescaleDB, Oracle, IBM DB2, SQLite.

PandoraFMS має інтеграції з MySQL та Percona.

Передбачення трендів

Zabbix прогнозує потенційні проблеми на основі історичних даних за допомогою функцій тригерів: forecast і timeleft. Задається період даних, які Zabbix має аналізувати, і поріг значення, який, якщо перетнути зверху або знизу, запускає тригер про майбутню проблему. Функція timeleft передбачає час до досягнення заданого значення. Функція forecast запускає тригер, якщо через заданий проміжок часу передбачене значення перетне заданий поріг.

Для передбачення трендів в PandoraFMS є Prediction Server, який описано в розділі 1.2. PandoraFMS/1.2.1 Архітектура.

Звітування про стан системи і збереження стану

За замовчуванням Nagios збирає два типи даних: дані про стан (англ. “state data”), та дані про продуктивність (англ. “performance data”). Звіти, які генерує Nagios XI засновані на якомусь з цих типів даних. Наприклад, звіти про доступність (англ. “availability reports”) серверів базуються на даних про стан серверів, а звіт про використання трафіку (англ. “bandwidth usage reports”) заснований на даних про продуктивність. Можна переглядати звіти, які датуються вчорашнім днем, минулим тижнем, місяцем або роком.

Zabbix надає звіти засновані на інформації про стан системи (стан Zabbix серверу, кількість гостів та об’єктів моніторингу тощо), тригерів та зібраних даних. Звіти про доступність відображають який відсоток часу тригер був у стані OK/PROBLEM. У Zabbix можна обрати період часу, звіти з якого потрібно показати. Між сповіщеннями також можна налаштувати залежності. Наприклад, якщо робота одного госту обумовлює роботу іншого, і вони обидва припинили роботу через відмову першого госту, то Zabbix дозволяє налаштувати залежність тригерів таким чином, що лише одне сповіщення про головну причину проблеми буде надіслано, а не два.

PandoraFMS генерує SLA звіти про будь-які здійснені нею перевірки, і так само може відображати минулі дані.

Нижче наведена таблиця, в якій порівнюються розглянуті системи моніторингу.

Таблиця:

	Zabbix	Nagios	PandoraFMS
Безагентний моніторинг	Так	Так	Так
Автовиявлення	Так	Так	Так
Розширення плагінами	Так	Так	Так
Інтеграція з іншими застосунками	Так	Так	Так
Способи сповіщень	e-mail,SMS,Jabber,Ez texting	будь-як, залежно від заданої конфігурації	e-mail, але можна розширити список за допомогою інтеграції з іншими застсоунками

Розподілений моніторинг	Так	Так	Так
Способи для збереження даних	Oracle, MySQL, PostgreSQL, IBM DB2, SQLite	Flat file, SQL (via ndoutils), MySql (Via Nconf)	MySQL, Oracle
Передбачення трендів	Так	Ні	Так
Звітування про стан системи і збереження стану	Так	Так	Так

Таблиця 1.1 - Порівняльна таблиця розглянутих систем моніторингу

Розділ 2: Структурна розробка власного рішення

2.1. Структурна схема

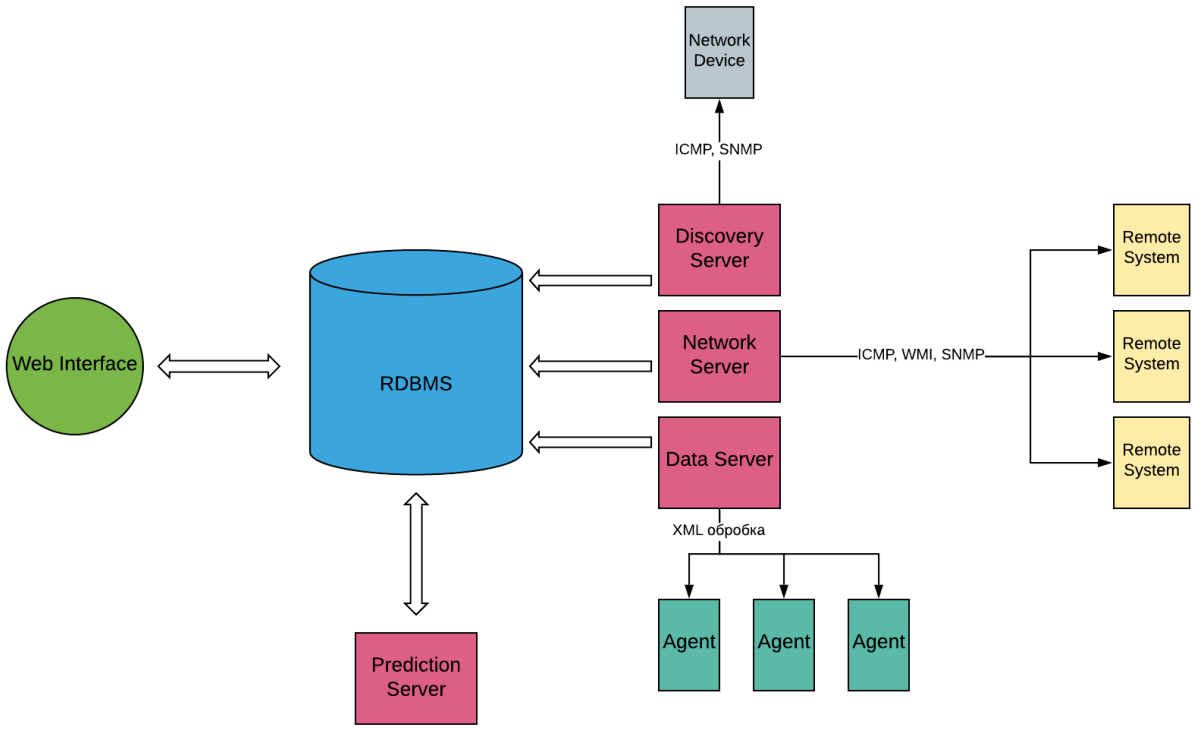

Рис. 2.1.1 - Структурна схема власного рішення

2.2. Опис компонентів системи

Архітектура системи буде мати такі компоненти: Discovery Server, Network Server, Data Server, Prediction Server, РСКБД, веб-інтерфейс та агентів.

ICMP (англ. Internet Control Message Protocol) - мережевий протокол. Ping - утиліта, за допомогою якого пакети ICMP ECHO_REQUEST відправляються на машину, яка має відповісти за допомогою ECHO_REPLY. Таким чином можна перевірити доступність машини, а також оцінити її навантаження залежно від втрачених пакетів та часу відповіді.

SNMP (англ. Simple Network Management Protocol) - мережевий протокол. Був створений для моніторингу мережевого обладнання, а також для зміни їхньої конфігурації. Будь-який пристрій, який можна моніторити за допомогою SNMP, має OID (англ. Object Identifier). Наприклад, пристрій для перевірки температури машини має свій OID. MIB - колекція значень основних властивостей об’єкта, над яким здійснюється моніторинг. Об’єкти в цій колекції мають OID.

WMI (англ. Windows Management Instrumentation) – реалізація стандарту WBEM, розширена компанією Microsoft. Побудована за об’єктно-орієнтованим принципом. Усі дані про ОС, її властивості, програми і прилади представлені у виді об’єктів класів, які мають свої властивості, методи та спадковість.

Discovery Server

За допомогою мережевих протоколів виявляє нові пристрої в мережі. По аналогії з механізмами автовиявлення в Zabbix та PandoraFMS, цей сервер може виконувати наперед задані завдання при виявленні нового об’єкта, наприклад, встановлення відповідних мережевих модулів. Автоматично додає нові об’єкти під моніторинг, якщо на них встановлено агента. Виконує перевірку на виявлення нових пристроїв із заданою періодичністю.

Network Server

Так само використовує мережеві протоколи для моніторингу віддалених пристроїв, а також WMI для моніторингу ОС. Обробляє результати і в тому разі, якщо значення відповідних метрик відхиляються

від заданих значень або виконується заданий логічний вираз, активуться тригер. Між тригерами можна встановити залежності. Таким чином про проблему недієздатності пристроїв, які зумовлюють роботу одне одного, буде одне повідомлення про головну проблему, а не повідомлення про недієздатність кожного пристрою окремо.

Prediciton Server

Компонент з штучним інтелектом, який відповідає за побудову трендів зміни значень метрик. Використовує дані з бази даних, зберігає їх. Результати відображаються у вигляді звіту. Так само як Zabbix, активує тригер, якщо через заданий проміжок часу значення метрик будуть відхилятися від норми.

Data Server

Компонент, що збирає та обробляє інформацію надіслану агентами. Приймає та обробляє XML-пакети. У випадку невідповідності даних заданим нормам, запускає тригер.

Агент

Компонент системи, який збирає інформацію про об’єкт. Надсилає інформацію у вигляді XML-пакетів. Модуль це команда або скрипт, який виконує агент. Для кожного агента є його конфігураційний файл з кодуванням UTF-8, в якому обов’язково вказується IP серверу, якому він має надсилати інформацію, та ім’я госту, вказується шлях для локального збереження даних перед відправленням до серверу і шлях для лог-файлів.

RDBMS (РСКБД)

Реляційна база даних, в якій зберігається вся зібрана інформація, яку потім відображає web-інтерфейс. Система підтримує MySQL та PostgreSQL.

Web-інтерфейс

Інтерфейс користувача, з якого користувач може побачити звіти та графи про роботу системи, сповіщення про проблеми, змінити або додати способи для сповіщення, змінити конфігурацію агента, встановити нових агентів, модифікувати їх, додати виявленні системою нові прилади під моніторинг.

Загальна робота системи полягає в тому, що Network та Data сервери збирають інформацію про прилади і зберігають оброблену інформацію до бази даних. Якщо оброблена ними інформація містить в собі відхилення від нормальних значень, ці сервери запускають тригери і користувач бачe сповіщення в інтерфейсі, а також отримує це сповіщення заданим способом (email, SMS тощо). Discovery сервер виявляє нові пристрої в мережі у межах заданого діапазону IP і виконує, якщо вони є, завдання при автовиявленні (наприклад, встановлення відповідних модулів або сповіщення). Користувач має вирішити, які виявлені пристрої додавати під моніторинг, а які ні, а потім може проглядати список всіх об'єктів. Prediction сервер на основі даних заданого проміжку визначає тренди і в разі, якщо вони через деякий час досягнуть критичного значення, запускає тригер.

Розділ 3: Розробка компоненту системи

Детально розроблятися буде агент, компонент, який відповідає за збір даних на гості.

Для агента обов'язково існує конфігураційний файл, в якому зазначена необхідна інформація для роботи агента, наприклад, IP адреса серверу, на який потрібно надсилати дані. Агент може входити у режим broker_mode, по аналогії з broker режимом в PandoraFMS. Усі зібрані дані будуть перекладатися в XML-формат та надсилатимуться на сервер. У кожного агента є тимчасове місце зберігання даних перед відправкою на сервер. Так само, як Zabbix-агенти, агент буде написано з використанням мови C, яка відома своєю швидкістю та поширеністю.

C - типізована процедурна мова програмування. Особливостями є відображення машинних команд у мовні конструкції та прямий доступ до пам'яті й керування нею.

XML (англ. Extensible Markup Language) - мова розмітки ієрархічно структурованих даних, яка призначена для обміну між різними застосунками.

Конфігураційний файл містить таку інформацію:

- server_ip - IP-адреса серверу, до якого надсилаються дані.
- host_ip - IP-адреса серверу, на якому збираються дані.
- agent_name - ім'я агента.
- broker_mode - прапор для переходу в режим брокера.

- temp - шлях до тимчасового зберігання даних перед відправкою на сервер.
- log_file - шлях до лог-файлу.

Також в конфігураційному файлі визначаються модулі, які має виконувати агент:

```
module_begin
    module_name  <ім’я модулю>
    module_description <опис модулю>
    module_exec <локальна команда для виконання>
    module_min_warning <мінімальне значення для попередження}>
    module_min_critical <мінімальне значення для критичної проблеми>
    module_max_warning <максимальне значення для попередження>
    module_max_critical <максимальне значення для критичної проблеми>
module_end
```

Для початку роботи агент використовує скрипт, який відслідковує статус агента, а також може його зупинити.

Алгоритм роботи агента демонструє наведена блок-схема:

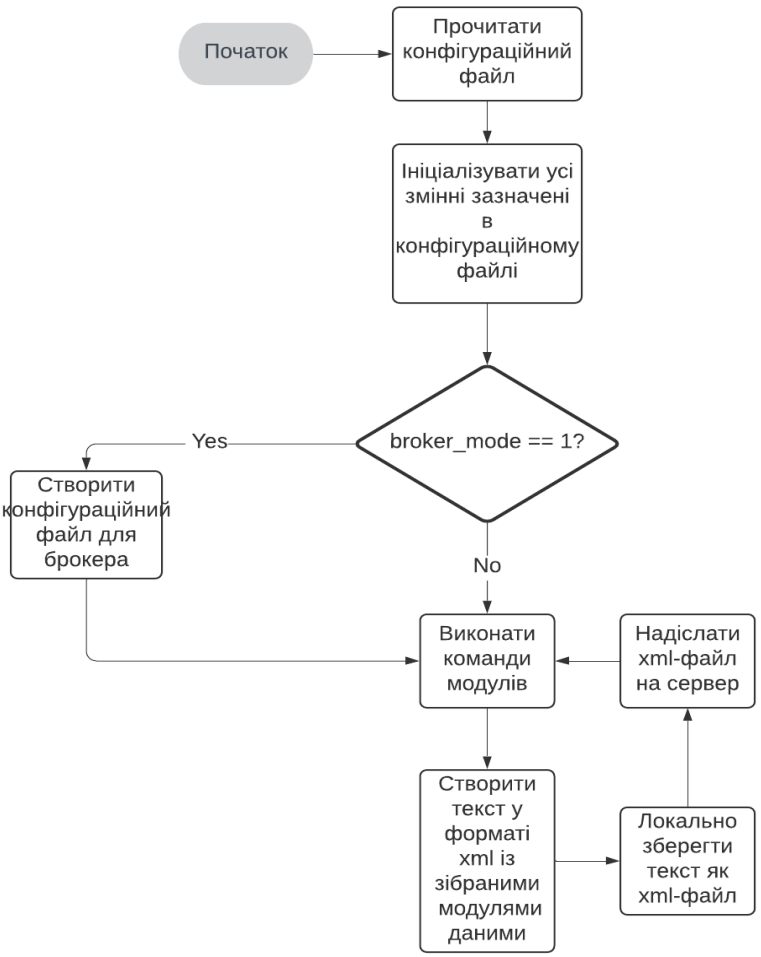

Рис. 3.1 - Блок-схема алгоритму роботи компонента

Методи, які забезпечуватимуть основне функціонування агента:

- void load_configs() - метод, який завантажує дані з конфігураційного файлу.
- void log_message(char* msg) - метод, який записує повідомлення в лог-файл.
- void create_broker_conf() - метод, який в режимі брокера створює для брокера окремий конфігураційний файл.
- void send_file() - метод для надсилання xml-файлу до сервера.
- void execute_module() - метод, що запускає модулі.
- void write_data_xml(char* data) - метод, який перетворює дані в xml-формат.

Приклад надісланого агентом xml-файлу. В ньому зазначається дані про агента і дані, зібраним кожним модулем. В даному прикладі модуль, який збирає дані про відсоток використання ЦП, знайшов значення 42%.

Приклад тексту в xml-файлі:

```
<agent_data description= group= os_name='linux' os_version='Ubuntu
10.10' version='3.2(Build 101227)' timestamp='2020/05/09 12:24:03'
agent_name='foo' timezone_offset='3' address='192.168.1.53'
custom_id='KS0602' url_address='http://localhost:8080'>
<module>
   <name>CPU</name>
   <description>CPU usage percentage</description>
   42
 </module>
</agent_data>
```

Код до компонента наведено у додатку А.

Висновки

Метою роботи було здійснити порівняльний аналіз та розробити своє власне рішення для системи моніторингу мережі підприємства.

Під час порівняльного аналізу:

- Було виділено основні функції системи моніторингу.
- Було розглянуто найпоширеніші на ринку системи моніторингу Zabbix, Nagios та PandoraFMS.
- Розглядаються архітектура кожної системи, особливості реалізації основних функцій кожною з цих систем, їхні недоліки та переваги.
- Результати аналізу були занесені у таблицю для наочного порівняння.

Під час розробки власного рішення:

- Була розроблена схема архітектури, яка поєднувала особливості кожної системи моніторингу. Схема враховує реалізацію всіх основних функцій системи моніторингу.
- Були розглянуті мережеві протоколи, які використовуються компонентами системи.
- До кожного компонента був даний детальний опис його роботи в системі.
- Описано загальний принцип роботи самої системи.

В роботі виконується розробка компонента системи:

- Були обрані мова програмування для написання компоненту та формат файлу, який отримується під час його роботи.
- Наведено блок-схему алгоритму роботи компоненту
- Дано перелік методів для роботи цього компонента
- Дано приклад тексту файлу, який отримується під час роботи компонента.

Список використаної літератури

1. Nagios XI Documentation [Електронний ресурс] https://library.nagios.com/library/products/nagios-xi/documentation/
2. Zabbix Documentation [Електронний ресурс] https://www.zabbix.com/documentation/
3. Pandora: Documentation en [Електронний ресурс] https://pandorafms.com/docs/index.php?title=Pandora:Documentation_en
4. Mastering Zabbix [Електронний ресурс] https://learning.oreilly.com/library/view/mastering-zabbix/9781783283491
5. Monitoring a production-ready microservice [Електронний ресурс] https://www.oreilly.com/content/monitoring-a-production-ready-microservice/?imm_mid=0ee8c5&cmp=em-webops-na-na-newsltr_20170310

Додаток А
Код до розробленого компоненту

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

void load_configs()
{
   static char    *hosts_active;

   struct cfg_line    cfg[] =
   {
         {"Server",              &SERVER_IP,              TYPE_STRING_LIST,
                  0,             0},
         {"Hostname",                  &HOST_IP,          TYPE_STRING,
                  0,             0},


         {"LogFile",             &LOG_FILE,               TYPE_STRING,
                  PARM_OPT,   0,             0},
         {"LogFileSize",               &LOG_FILE_SIZE,          TYPE_INT,
                  PARM_OPT,   0,             1024},
         {"LoadModule",                &LOAD_MODULE,            TYPE_MULTISTRING,
                  PARM_OPT,   0,             0},
         {"Temp",                &TEMP,                TYPE_STRING,
                  PARM_OPT,   0,             0},
         {"DataBasePath",              &DB_PATH,                TYPE_STRING,
                  PARM_OPT,   0,             0},
         {"BrokerMode",                &BROKER_MODE,                  TYPE_INT,
                  PARM_OPT,   0,             0},
         {"ProxyMode",                 &PROXY_MODE,             TYPE_INT,
                  PARM_OPT,   0,             0},

   parse_cfg_file(CONFIG_FILE, cfg);

   }
}


void create_broker_conf()
```

```
{
    FILE * fp;
        char * line = NULL;
        size_t len = 0;
        size_t read;

        fp = fopen(CONFIG_FILE);
        fo = fopen(BROKER_CONF_FILE);
        if (fp == NULL)
        exit(EXIT_FAILURE);

        while ((read = getline(&line, &len, fp)) != -1) {
         fputs(line, fo);
        }

        fclose(fo);
        fclose(fp);

}


void send_file() {

    {
        int sockfd, nBytes;
        FILE* fp;

        sockfd = socket(AF_INET, SOCK_DGRAM, SERVER_IP);

        if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)


        nBytes = recvfrom(sockfd, net_buf,
                NET_BUF_SIZE, sendrecvflag,
                (struct sockaddr*)&addr_con, &addrlen);

        if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
                sendto(sockfd, net_buf, NET_BUF_SIZE,
                sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
                break;
        }

        sendto(sockfd, net_buf, NET_BUF_SIZE,
                sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
        clearBuf(net_buf);
```

```
        }
        if (fp != NULL)
        fclose(fp);
        }


void execute_module(struct Modules m)
{
        evaluate_module_coomand (m, value[0]);
                        write_module_xml (m, value);

}

void write_data_xml()
{
    while(module) {

        module = *(Modules++);
        execute_module(module);
        }


        XML_DATA = "<?xml version='1.0' encoding=”+CONF_ENCODING+”?>\n" .
                "<agent_data description='" +CONF_DESCRIPTION +  "' os_name='+OS+'
os_version='+OS_VERSION+ ' version='"+ CONF_AGENT_VERSION +
                "' agent_name='"+ CONF_AGENT_NAME +"' timezone_offset='".
CONF_TIMEZONE_OFFSET+"' custom_id='" + CONF_ID +”>”
}


exit(EXIT_SUCCESS);
```