Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних систем та технологій
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента Малика Дмитра Андрійовича
(ПІБ)
академічної групи 123-17СК-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему « Комп'ютерна система моніторингу стану серверного обладнання з детальним опрацюванням побудови, налаштування та безпеки корпоративної мережі»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Гнатушенко В.В.			
розділів:				
апаратний розділ	доц. Ткаченко С.М			
розрахункові мережі	ас. Панферова Я.В.			
економічний розділ	ст. викл. Яремчук І.О.			
охорона праці	доц. Іконніков М.Ю			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних систем
та технологій
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
«____»_____*2020 року*

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Малика Д.А. **академічної групи** 123-17ск-1
(прізвище та ініціали) (шифр)
спеціальності 123 «Компютерна інженерія»
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)
на тему « Комп'ютерна система моніторингу стану серверного обладнання з детальним опрацюванням побудови, налаштування та безпеки корпоративної мережі»
затверджену наказом ректора НТУ «Дніпровська політехніка» від _____№__

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати завдання, конкретизувати предмет та мету роботи	10.05.2020
Технічні вимоги до комп'ютерної системи	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати технічні вимоги до розробки комп'ютерної системи	17.05.2020
Спеціальна частина	Розв'язати завдання з розробки комп'ютерної системи з опрацюванням побудови і захисту інформації та налаштуванням корпоративної мережі	24.05.2020
Економічна частина	Економічно обґрунтувати доцільність витрат на створення та дослідження системи моніторингу	2.06.2020
Охорона праці	Розробити організаційно-технічні заходи, щодо реалізації правил безпеки при експлуатації системи моніторингу	5.06.2020

Завдання видано _____ проф. Гнатушенко В.В
(підпис керівника) (прізвище, ініціали)
Дата видачі 1 квітня 2020 р.
Дата подання до екзаменаційної комісії 10.06.2020 р.
Прийнято до виконання _____ Малик Д.А
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи на тему «Комп'ютерна система моніторингу стану серверного обладнання з детальним опрацюванням побудови, налаштування та безпеки корпоративної мережі» складається з вступу, п'яти розділів, висновків і списку літературних джерел.

Необхідність прогнозування, а тим самим і запобігання, поломок, оповіщення про них і зберігання інформації про стан систем і служб в будь-якій IT-системі , забезпечує актуальність даної роботи.

Записка містить 65 сторінок, 15 рисунків, 5 таблиць, 14 використаних джерел.

Ключові слова: моніторинг, серверне обладнання, Nagios, Zabbix, Cacti, автоматизована система моніторингу, аналіз.

Об'єкт дослідження: програмне забезпечення для моніторингу стану серверного обладнання

Мета роботи: розробка і реалізація проекту моніторингу стану серверного обладнання.

Розробка комп'ютерної системи моніторингу виконана відповідно до завдання на кваліфікаційну роботу бакалавра.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕН, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

У пояснювальній записці використовуються такі визначення, позначення і скорочення:

ОС – операційна система

БД – база даних

ІТ – інформаційні технології

SLA – Service Level Agreement - угода про рівень послуг

AD – Active Directory - активний каталог

SCOM – System Center Operations Manager - системний центр менеджера операцій

NPI – Number Plan Indicator – телефонний план нумерації

RRD – Round Robin Database – кільцева база даних

JMX – Java Management Extensions – розширення управління Java

DNS – Domain Name System – система доменних імен

SNMP – Simple Network Management Protocol – простий протокол мережевого управління

GSM – Groupe Special Mobile, перейменований в Global System for Mobile Communications - глобальний стандарт цифрового мобільного стільникового зв'язку

IPMI – Intelligent Platform Management Interface – інтелектуальний інтерфейс управління платформою

SSL – Secure Sockets Layer – рівень захищених сокетів

SSH – Secure Shell - безпечна оболонка

API – Application Programming Interface – інтерфейс програмування додатків

WMI – Windows Management Instrumentation – інструментарій управління Windows

URL – Uniform Resource Locator – єдиний покажчик ресурсів

ВСТУП

Моніторинг стану серверного обладнання, будь то маленька компанія або величезний дата-центр, потрібен, щоб системні адміністратори були сповіщені про поломки і проблеми в інфраструктурі раніше, або хоча б одночасно з користувачами. Необхідність прогнозування, а тим самим і запобігання поломок, оповіщення про них і зберігання інформації про стан систем в будь-який IT-системі забезпечує актуальність даної роботи. По суті своїй моніторинг – це комплекс швидкого знаходження проблеми, оповіщення про неї адміністраторів, а також діагностики, що дає повну і точну інформацію про поломку.

Моніторинг серверного обладнання включає в себе відстеження працездатності серверів, персональних комп'ютерів і мережі. В ході моніторингу збираються і обробляються дані для підтримки стабільності роботи бізнес-процесів.

Автоматизована система моніторингу обладнання допомагає своєчасно виявити проблеми в його роботі і усунути їх, а також попередити їх появу. У ході цього процесу складається детальна статистика і надається можливість створення повідомлень про виникнення певних ситуацій, позначених у системі.

Моніторинг роботи обладнання складається з наступних складових:

1. Відстеження стану серверів і ПК
2. Контроль мережевого обладнання
3. Спостереження за станом джерел безперебійного живлення

Відстеження стану серверів і ПК: ця функція включає в себе моніторинг апаратної і програмної частини. У першому випадку система здійснює перевірку таких параметрів, як:

– стан вінчестерів, процес їх завантаження;

- процес завантаження апаратних компонентів для виявлення «вузьких місць» у функціонуванні систем і ефективного планування модернізації;
- робота процесорів, оперативної пам’яті;
- процес завантаження мережевих інтерфейсів;
- температура обладнання;
- робота кольорів;
- напруга серверних компонентів, їх відповідність номінальному значенню.

Моніторинг програмної частини включає в себе:

- оповіщення про перевантаження операційної системи;
- оповіщення в разі відключення і розриву з'єднання з сервером;
- відстеження роботи встановлених додатків, оповіщення при їх інсталяції і деінсталяції.

Програма моніторингу також здійснює перевірку роботи служб і додатків, зокрема:

- процесу запуску і припинення роботи найважливіших з них
- роботи поштових серверів, СУБД, проксі-серверів

Контроль мережевого обладнання. Ця функція має на увазі:

- відстеження роботи обладнання;
- перевірку процесу завантаження портів і їх доступності;
- перевірку доступності окремих TCP портів;
- визначення часу відгуку;
- відстеження температури і завантаження центрального процесора.

Об'єктом дослідження є різні IT інфраструктури, предметом дослідження – засоби моніторингу цих інфраструктур (в даній роботі наводиться приклад реалізації моніторингу в службі підтримки серверів компанії ФЛП «Коваленко Е.В.».)

Мета роботи – розробка системи моніторингу, повністю відповідної до вимог конкретного проекту компанії, з мінімальними витратами.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Вимоги до систем моніторингу

Усі системи управління і моніторингу (СУМ) слугують для того, щоб знизити роль людського чинника при експлуатації IT-інфраструктури. Вочевидь, обсяг інформації, що має відстежувати системний адміністратор, просто величезний. Тому він повинен мати кваліфікацію, достатню для вирішення проблем. Слід розуміти, що система моніторингу не може замінити собою кваліфікацію користувачів і адміністраторів. Вона лише надає їм необхідні відомості для ухвалення правильного рішення.

Система моніторингу має давати чітку відповідь на питання, пов'язані з ефективністю функціонування підсистеми IT-інфраструктури. При цьому поряд зі зниженням наявних витрат об'єкта на інформаційні технології СУМ забезпечують подальший розвиток IT-інфраструктури у виваженому й стратегічно правильному напрямі. Насамперед тому, що керівництво об'єкта має розуміти, які процеси відбуваються на сучасному рівні розвитку IT-інфраструктури і яка модернізація необхідна в перспективі. Адже впровадження нових сервісів і технологій вимагає певного рівня розвитку IT-інфраструктури. Найчастіше впровадження нових програмно-апаратних засобів неможливе через "слабку" IT-інфраструктуру, коли система впроваджується, але не може потім повноцінно функціонувати.

Треба визначити ключові питання, пов'язані з управлінням інформацією, комунікаціями, моніторингом IT. Розкриваючи суть і значення моніторингу, його роль в управлінні IT, необхідно засвоїти його основні принципи й загальні вимоги до нього.

Моніторинг серверного обладнання реалізується через визначення важливих робочих показників, систематичну і своєчасну звітність, оперативне коригування виявлених відхилень. Він особливо важливий через складність IT-процесів і наявність пов'язаних із ними ризиків. Його мета –

гарантувати інформаційне забезпечення, яке допомагатиме організації досягти цілей і виконати робочі завдання серверного обладнання.

Основними об'єктами для моніторингу є: сервери, мережеве обладнання, додатки, бази даних, користувальницькі станції, спеціальні системи.

1.2 Функції систем моніторингу

Основним завданням системи моніторингу є надання актуальної інформації для аналізу стану IT-інфраструктури та швидкого виявлення виниклої несправності та її оперативне усунення. Системи моніторингу продуктивності дозволяють IT-фахівцям вчасно помітити зниження продуктивності і визначити «вузькі місця» в IT-інфраструктурі. Постійний моніторинг допомагає уникнути простоїв в її роботі, підтримувати всі IT-сервіси в робочому стані і зберігати необхідний рівень їх якості, а також спланувати її модернізацію.

Раніше роль моніторингу здійснювали адміністратори, а інформація про стан систем у кращому випадку збиралася ними ж у будь-яких неспеціалізованих програмах (через їхню відсутність), в гіршому – взагалі ніяк не накопичуючих і не агрегованих. Всі відомості про систему були прив'язані до практичного досвіду роботи з інфраструктурою у конкретного фахівця і повністю губилися при його догляді. Зараз з'явилося безліч напів- і повністю автоматизованих систем для моніторингу, які аналізують стан систем, збирають інформацію в колекції, які теж згодом можна вивчити при необхідності.

Існують досить специфічні види моніторингу, наприклад, від імені кінцевого користувача, коли в задані проміжки часу циклічно емулюються його дії. Зазвичай це робот, планувальник завдань, що запускає спеціальний, заздалегідь певний скрипт-сценарій, а потім рапортують про успіх виконання дій або про виниклі в процесі помилках.

Для зберігання отриманої інформації зазвичай використовується конфігураційна база даних під різними СУБД: інформація про об'єкти моніторингу представлена, як набір конфігураційних одиниць. Кожен сервер, кожний мережевий пристрій - це якась одиниця, все це зберігається в системі моніторингу з візуальними уявленнями: діаграмами, графіками та іншим.

Сама структура моніторингу значно видозмінюється з плином часу. Наприклад, одна з тонкощів виникла з появою і великому поширенні віртуалізації: якщо раніше була необхідність відстежувати стан тільки фізичних серверів, то тепер на кожному з них може бути ще кілька віртуальних.

Також системи моніторингу можна налаштувати на виконання будь-яких стандартних сервісних дій. Наприклад, очищати кошик при її заповненні або активувати архівування для будь-яких файлів, коли певний відсоток дискового простору стає зайнятим.

При виборі, розробці, впровадженні систем моніторингу спочатку необхідно визначитися з об'єктами, які будуть піддаватися стеженню, а також критичні події та показники, які і визначать кількість повідомлень при поломці, частоту сканування і інші параметри і наслідки. Для великих інфраструктур, наприклад дата-центрів, перед фінальним впровадженням зазвичай розгортають тестовий майданчик, де можна оцінити доцільність зроблених рішень і ухвал параметрів порогових значень.

Впровадження подібних рішень особливо важливо при використанні сервісного підходу до діяльності IT-підрозділів, коли всі процеси переглядаються з точки зору що надаються підрозділом IT-сервісів. Кожен бізнес-сервіс корпоративної системи по можливості інтерпретується як IT-сервіс, задається певний рівень якості його надання. Далі він описується в системі моніторингу як набір взаємопов'язаних компонентів IT-інфраструктури.

В результаті формується Угода про рівень якості сервісів (Service Level

Agreement, SLA). Згідно SLA система здійснює збір і зберігання інформації про якість надання IT-сервісів. На базі накопиченої інформації формуються звіти за певний період часу.

Аналіз звітної інформації допомагає здійснювати:

- перегляд рівня надання IT-сервісів;
- реорганізацію діяльності IT-підрозділу;
- модернізацію IT-інфраструктури.

Системи моніторингу можуть бути орієнтовані на споживачів різного рівня. Для великих систем зазвичай використовується величезна кількість різноманітних функцій, для маленьких зазвичай мало спільного аналізу вузлів і відправки оповіщень.

Серед основних функцій моніторингів можна виділити наступні:

- Стеження. Основна функція, що включає в себе періодичний збір показників з вузлів устаткування, сервісів і т.п.
- Зберігання інформації. Доповнення до стеження. здійснюється збір інформації за основними показниками кожного об'єкта моніторингу, для зберігання зазвичай використовуються бази даних.
- Побудова звітів. Здійснюється як на основі поточних даних стеження, так і по довго тривало зберігається. наприклад, довготривалий моніторинг навантаження на сервер може попередити, що споживані ресурси весь час збільшуються, значить необхідно збільшити доступні засоби або перенести частину завдань на інший сервер, вибір якого теж можна здійснити на основі довготривалого звіту.
- Візуалізація. Звіти в візуальному представленні: у вигляді графіків, спливаючих підказок, діаграм. Допомагають легкому сприйняттю інформації, а також можливий вибір для візуалізації декількох, найважливіших індикаторів, тоді як в звітах будуть представлені всі показники.

– Пошук вузьких місць. На основі аналітичних даних моніторингу можливо дізнатися, в яке саме місце інфраструктури найбільш сильно знижує загальні показники продуктивності.
– Автоматизація сценаріїв. Функція звільняє адміністраторів від рутинних завдань.

Завдяки наявності коштів для реалізації всіх цих функцій адміністратора більше не потрібно перевіряти вручну стан кожної складової системи, проблеми вирішуються і поломки усуваються більш оперативно, діагностика здійснюється багатомірно і точно, а також можна планувати розширення інфраструктури.

Використання систем моніторингу та управління дозволяє:

– оптимізувати використання інформаційних ресурсів;
– підвищити якість IT-сервісів і швидкість усунення збоїв в роботі обладнання та програмного забезпечення, мінімізувати час простою сервісів;
– забезпечити надійність, безпеку і узгоджене функціонування всіх компонентів IT-інфраструктури;
– полегшити модернізацію IT-інфраструктури;
– в кілька разів підвищити ефективність роботи IT-підрозділу.

1.3 Microsoft SCOM

System Center Operations Manager – система наскрізного моніторингу від Microsoft, в тому числі активного спостереження за станом мереж (спостереження за будь-якими мережевими пристроями, що підтримують SNMP, аж до рівня портів, а також виявлення віртуальних локальних мереж і комутаторів в таких мережах). В останніх версіях з'явилася можливість стеження не тільки за системами, під управлінням операційних систем сімейства Windows, але і за гетерогенними середовищами, що включають

UNIX і Linux. System Center Operations Manager призначений головним чином для організацій з числом машин більше 500 і числом серверів понад 30. Для менших організацій існує продукт System Center Essentials, що включає в себе частину функціоналу продуктів System Center Operations Manager і System Center Configuration Manager, але призначений для малих і середніх підприємств.

Сам продукт, починаючи з версії 2012 року, є сервісом високої доступності, завдяки відсутності серверів управління. У пулі з декількома серверами навантаження балансується і забезпечується доступність. На кожному сервері працює служба конфігурації, причому зберігання даних реалізовано не в пам'яті або XML-файлах, а в базі даних.

Microsoft також надає можливість інтеграції продукту з System Center Service Manager, завдяки чому з'являється можливість автоматичного створення інцидентів на основі повідомлень SCOM.

Що стосується тонкого спостереження за віртуальними середовищами, є засоби для інтеграції з пакетом System Center Virtual Machine Manager, який буде передавати System Center Operations Manager інформацію про віртуальних машинах, службах, приватних хмарах і вузлах.

Основні переваги:

- виняткова продуктивність і працездатність додатків для програмних середовищ Microsoft;
- забезпечує наскрізне управління службами для сервісів вашого центра обробки даних;
- сприяє поліпшенню ефективності і управління середовищами центрів обробки даних;
- уніфікований контроль в рамках приватних і загальнодоступних хмарних сервісів;
- підтримка Windows PowerShell 2.0 з набором нових командлетів.

Одне з головних достоїнств System Center Operations Manager – просунута візуалізація всього величезного зібраного набору даних, в основному у вигляді графіків і діаграм, причому візуалізація доступна не тільки в спеціальній консолі програми, але і через веб-інтерфейс. Елементи уявлення же можна піддавати тонкої настройки, приклад інтерфейсу на рисунку 1.1.

Версія 2012 підтримує розширене спостереження за змішаними середовищами, а саме за машинами під управлінням Unix і Linux (так званих «систем * nix: агент * nix підтримує HP-UX 11i версії 2 або 3 на базі PA-RISC і IA64, Sun Solaris 9 на базі SPARC і 10 на базі SPARC і 32-розрядної платформи, Red Hat Enterprise Linux 4, 5 і 6 на 32- і 64-розрядних платформах, Novell SuSE Linux Enterprise Server 9 на 32-розрядній платформі, 10 SP1 і 11 на 32- і 64-розрядних платформах, а також IBM AIX 5.3, 6.1 і 7.1 на базі POWER. Для моніторингу використовуються 2 ключа: sudo (для конфігурації стандартної облікового запису з потрібним рівнем доступу) і SSH (для безпечного обслуговування агента).

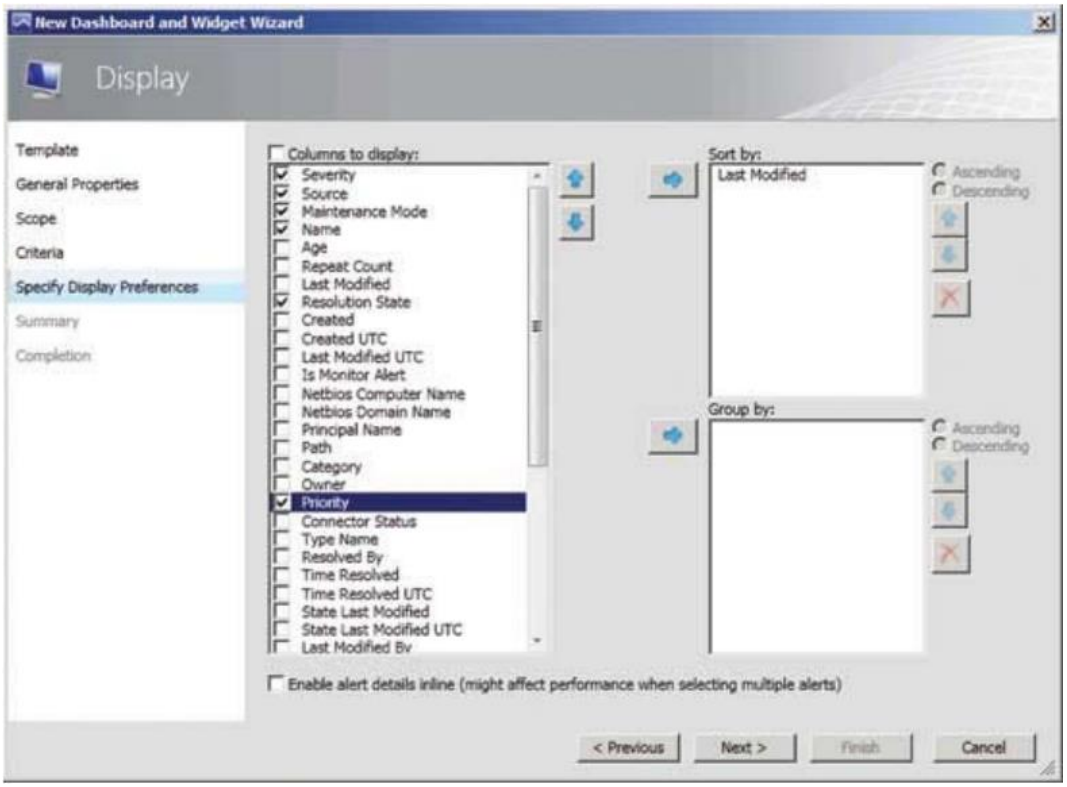

Рисунок 1.1 – Налаштування візуального представлення панелі SCOM

Отже, можна сказати, що System Center Operations Manager – це моніторинг високої доступності з спрощеної інфраструктурою, що використовується в організаціях з великим парком машин під управлінням різних сімейств операційних систем, що включає безліч різноманітних засобів стеження, в тому числі за мережевим обладнанням, а також розширені засоби уявлення зібраної інформації.

Однак дана система має ряд недоліків з точки зору вирішення конкретної технічної задачі:

- система моніторингу охоплює безліч загальних показників системи, але непридатна для стеження за специфічними параметрами;
- до сих пір робота з операційними системами поза сімейства Windows нестабільна;
- необхідність установки сервісу агента;
- неймовірна громіздкість і трудомісткість налаштування продукту «під себе»: система довше підходить для моніторингу загального стану і збору основних відомостей про великий структурі (наприклад, безліч клієнтських і серверних машин в домені).

Останній недолік обумовлює відмову від цієї системи як специфічного моніторингу проекту. Необхідно розгортати глобальний сервіс, встановлювати агентське програмне забезпечення на всі машини, що відслідковуються, і налаштовувати безліч параметрів – скасовувати показники, що збираються за замовчуванням. Система відноситься до продуктів для контролювання загального стану великий IT-структури без стеження за конкретними специфічними показниками, а значить абсолютно не підходить під поставлені проектом мети.

Також істотний недолік системи полягає у високій вартості даного програмного продукту.

1.4 Zabbix

Zabbix – вільно поширювана система для комплексного моніторингу мережевого обладнання, серверів та сервісів. Складається з чотирьох частин:

- Сервер моніторингу (ядро) – виконує періодичний опитування і отримання даних, обробляє їх, аналізує, також здійснює запуск скриптів для розсилки повідомлень. Може віддалено перевіряти мережеві сервіси, є сховищем, в якому зберігаються всі конфігураційні, статистичні та оперативні дані. Не може розташовуватися на сервері під керуванням операційної системи сімейства Windows, а також OpenBSD.
- Проксі – збирає дані про продуктивність і доступність від імені Zabbix сервера. Всі зібрані дані заносяться в буфер на локальному рівні і передаються до Zabbix сервера, до якого належить проксі сервер. Zabbix проксі є ідеальним рішенням для централізованого віддаленого моніторингу місць, філій, мереж, які не мають локальних адміністраторів. Він може бути також використаний для розподілу навантаження одного Zabbix сервера. В цьому випадку, проксі тільки збирає дані, тим самим на сервер лягає менше навантаження на ЦПУ і на введення / виведення диска.
- Агент – запускається на об'єктах, що відслідковуються, і надає дані сервера, здійснюючи контроль локальних ресурсів і додатків (таких як жорсткі диски, пам'ять, статистика процесора і т. д.) на мережевих системах, тобто ці системи повинні працювати з запущеним Zabbix агентом (проте моніторинг можна проводити не тільки за допомогою нього, але і по SNMP версій 1, 2, 3, запуском зовнішніх скриптів, що видають дані, і кілька видів зумовлених вбудованих перевірок, таких як ping, запит по http, ssh, ftp і іншим протоколам, а так же завмер часу відповіді цих сервісів. Zabbix агенти є надзвичайно ефективними через використання вбудованих системних викликів для збору інформації про статистикою. Zabbix-

агенти підтримуються не тільки на * nix операційних системах, а й на AIX і Windows. Підтримувані платформи вказані в таблиці 1.1.

– Веб-інтерфейс – засіб візуального представлення Zabbix, реалізований на PHP, для запуску вимагає наявності веб-сервера, представлений на рисунку 1.2.

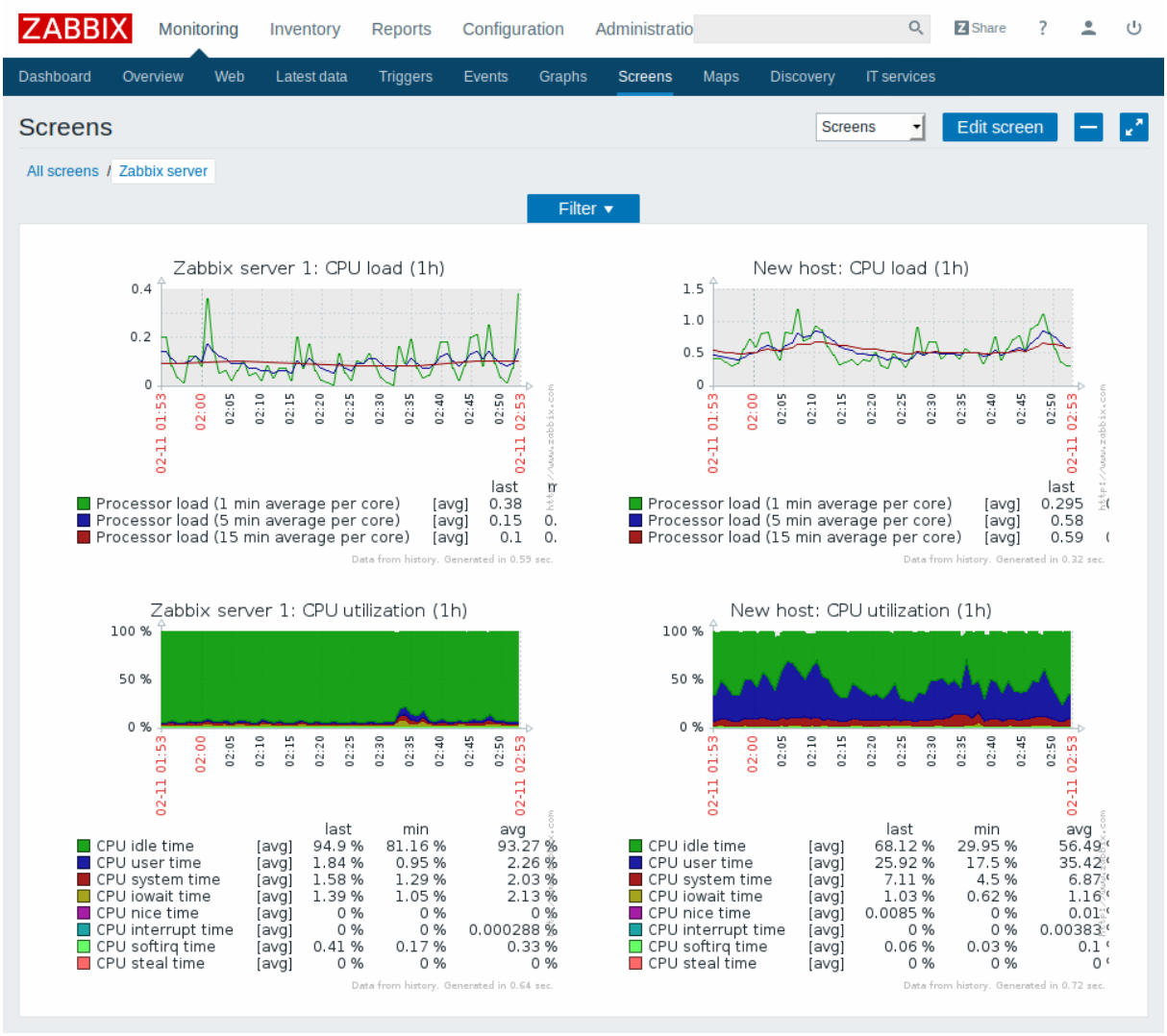

Рисунок 1.2 – Веб-інтерфейс Zabbix

Zabbix підтримує платформи, зазначені в таблиці 1.1.

Таблиця 1.1 – Підтримувані платформи

Платформа	Zabbix-сервер	Zabbix-агент
AIX	Підримується	Підримується
FreeBSD	Підримується	Підримується
HP-UX	Підримується	Підримується
Linux	Підримується	Підримується
Mac OS X	Підримується	Підримується
Novell Netware	–	Підримується
Open BSD	Підримується	Підримується
SCO Open Server	Підримується	Підримується
Solaris	Підримується	Підримується
Tru64/OSF	Підримується	Підримується
Windows NT 4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista	–	Підримується

За допомогою Zabbix можна здійснювати розподілений моніторинг до 1000 вузлів, де конфігурація молодших вузлів контролюється старшими в ієрархії. Також продукт включає централізований моніторинг лог-файлів, можливість створювати карти мереж (вручну за шаблоном), виконання запитів в різні бази даних, генерацію звітів і тенденцій, виконання сценаріїв на основі моніторингу, підтримку інтелектуального інтерфейсу управління платформами (IPMI).

Zabbix надає гнучкі можливості по налаштуванню умов-тригерів, які включаються при аваріях і неполадках, і система починає моргати лампочками (насправді червоними квадратиками), оповіщаючи адміністратора про можливу поломку. Також, при включенні тригера, веб-інтерфейс навіть починає пищати на манер будильника.

Zabbix досить самостійний і зможе відправити повідомлення на пошту, в jabber або sms за допомогою gsm-модему, або навіть спробувати самостійно підняти сервіс, що впав, виконавши заздалегідь певні дії, які запускаються при спрацьовуванні певних тригерів.

Для відображення логічної структури мережі можна вручну створювати карти мережі (приклад якої представлений на рисунку 1.3), що відображають саме розташування вузлів мережі і зв'язків між ними, причому поточний стан вузлів буде відображатися на карті.

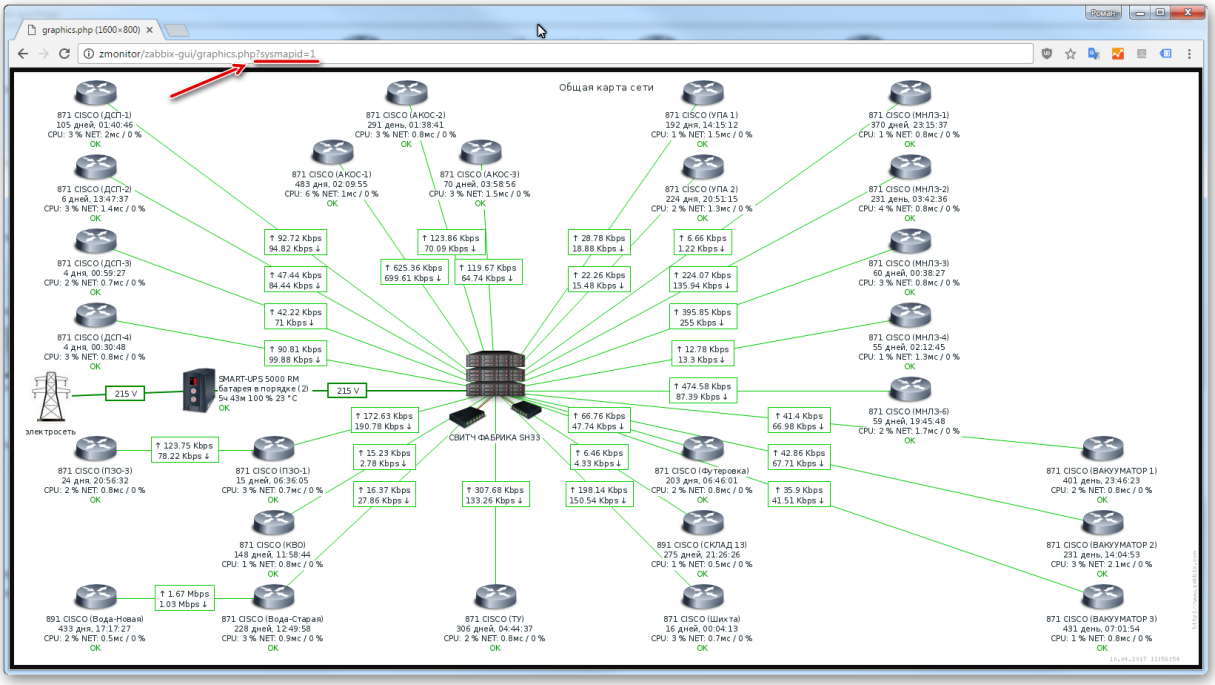

Рисунок 1.3 – Карта мережі в Zabbix

Автоматичне виявлення:

- автоматичне виявлення за діапазоном IP-адрес, доступним сервісам і SNMP перевірка;
- автоматичний моніторинг виявлених пристроїв;
- автоматичне видалення відсутніх хостів;
- розподіл по групах і шаблонами в залежності від що повертається результату.

В запасі у Zabbix є ще півдесятка функцій, які дозволяють ще більше спростити спостереження за мережею, такі як моніторинг стану веб-сайту за допомогою автоматичного виконання сценарію на кшталт імітації призначених для користувача дії на сайті. У підсумку це одна з найпотужніших і найпоширеніших систем моніторингу.

У підсумку ми отримуємо систему, яка найбільш підходить для наших цілей, яку також можна використовувати в якості «скелета» до своїх власних скриптів моніторингу. Однак в черговий раз варто відзначити громіздкість сервісу, відсутність повної документованості можливостей проекту, а також необхідність установки агентського програмного забезпечення на всі машини.

В якості додаткового недоліку варто відзначити складність делегування прав – машина з сервісом найчастіше керується операційною системою сімейства *nix, що робить трудомістким взаємодію з доменними користувачами і правами з Active Directory (Windows системи).

1.5 Nagios

Nagios (спочатку Netsaint) – вільно розповсюджувана програма для моніторингу систем і мереж. Спочатку розроблена для операційних систем на базі Linux, зараз однаково добре працює також і під Sun Solaris, FreeBSD, AIX і HPUX. За допомогою цієї програми доступні комплексне спостереження за всією IT-інфраструктурою, виявлення проблем відразу після їх виникнення, можливість ділитися отриманим при спостереженні даними із зацікавленими особами, моніторинг безпеки системи, і, як наслідок, скорочення часу простою і комерційних втрат

Можливості:

- моніторинг мережевих служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP);

- моніторинг стану хостів (завантаження процесора, використання диска, системні логи) в більшості мережевих операційних систем;
- підтримка віддаленого моніторингу через шифровані тунелі SSH або SSL;
- можливість побудови карт мереж (приклад на рисунку 1.4);
- проста архітектура модулів розширень (плагінів) дозволяє, використовуючи будь-яку мову програмування за вибором (Shell, C ++, Perl, Python, PHP, C # та інші), легко розробляти свої власні способи перевірки служб;
- паралельна перевірка служб;
- можливість визначати ієрархії хостів мережі за допомогою «батьківських» хостів, дозволяє виявляти і розрізняти хости, які вийшли з ладу, і ті, які недоступні;
- відправка повідомлень в разі виникнення проблем зі службою або хостом (за допомогою пошти, пейджера, смс, або будь-яким іншим способом, визначеним користувачем через модуль системи);
- можливість визначати обробники подій, що відбулися зі службами або хостами для проактивного вирішення проблем;
- автоматична ротація лог-файлів;
- можливість організації спільної роботи декількох систем моніторингу з метою підвищення надійності і створення розподіленої системи моніторингу;
- включає в себе утиліту nagiostats, яка виводить загальне зведення по всім хостам, за якими ведеться моніторинг.

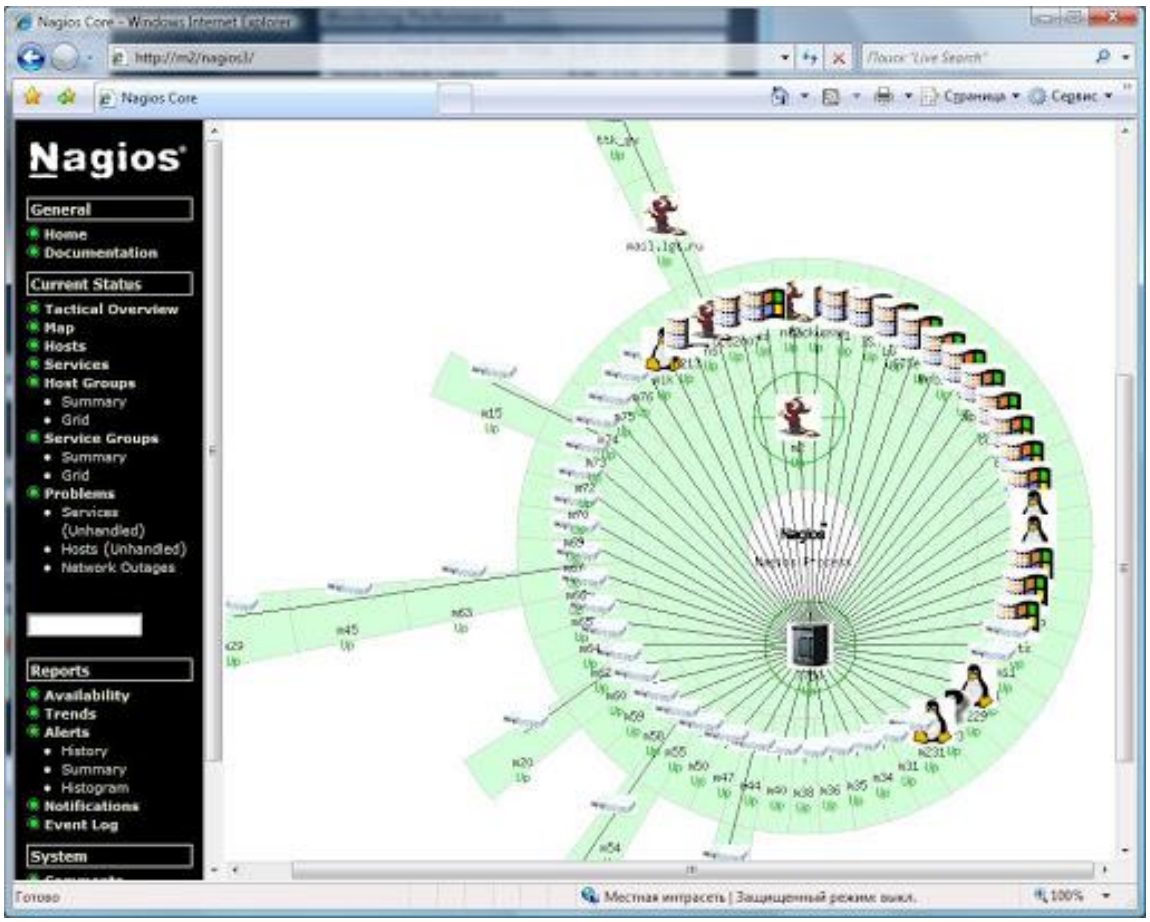

Рисунок 1.4 – Карта мережі в Nagios

Відмовою від використання системи послужили наступні причини:

– «загальний» характер моніторингу показників;
– проблема взаємодії з серверами під управлінням Windows;
– «мережева» спрямованість моніторингу.

1.6 Cacti

Cacti – безкоштовний додаток моніторингу, що дозволяє збирати статичні дані за певні часові інтервали і відображати їх в графічному вигляді за допомогою RRDtool утиліти, призначеної для роботи з круговими базами даних (Round Robin Database), які використовуються для зберігання інформації про зміну однієї або декількох величин за певний проміжок часу. Стандартні шаблони збору включають статистику по завантаженню

процесора, виділенню оперативної пам'яті, кількістю запущених процесів, використання вхідного та вихідного трафіку.

Cacti написаний в інфраструктурі Apache-PHP-MySql, дозволяє налаштовувати збір і відображення даних моніторингу на основі веб-інтерфейсу, представленого на рисунку 1.5, з юзер-френдлі організацією. Є можливість дописування власних агентів збору даних.

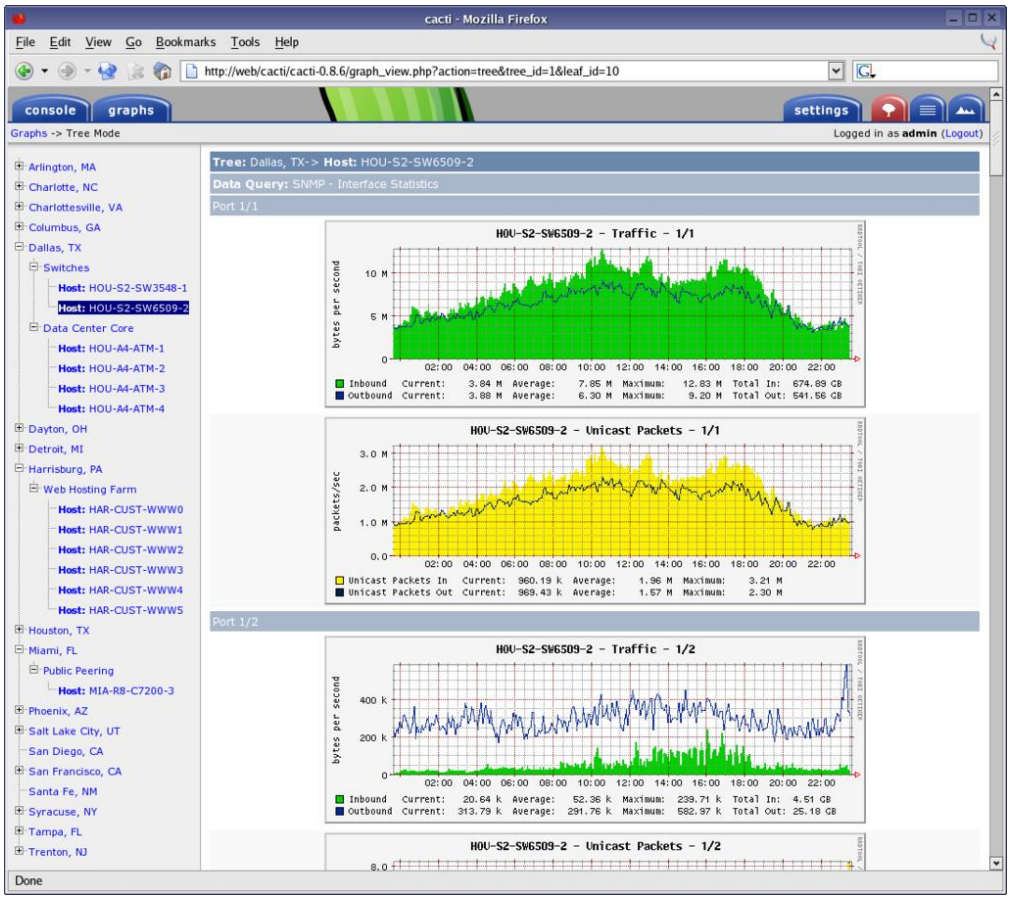

Рисунок 1.5 – Інтерфейс Cacti

Інтерфейс відображення статистики, зібраної з пристроїв, представлений у вигляді дерева, структура якого задається самим користувачем. Як правило, графіки групують за певними критеріями, причому один і той же графік може бути присутнім в різних гілках дерева. Є варіант перегляду заздалегідь складеного набору графіків, і є режим попереднього перегляду. Кожен з графіків можна розглянути окремо, при цьому він буде представлений за останні день, тиждень, місяць і рік.

Можливо самому вибрати часовий проміжок, за який буде згенеровано графік.

Переваги Cacti:

- висока швидкість розгортання при мінімальному додатковому кодуванні;
- простота і зручність інтерфейсу перегляду графіку і їх налаштування

Недоліки Cacti:

- досить швидке наростання кількості однотипних налаштувань в разі великого числа середовищ і серверів;
- обмежена продуктивність «нерідних» JMX рішень для Cacti;
- відсутність можливості інвентаризації.

Логіка роботи системи заснована на обслуговуванні ряду пристроїв (Devices – в термінології Cacti). Кожен пристрій – це хост, до якого є доступ по мережі, тобто воно характеризується IP-адресою або DNS-ім'ям. З пристроєм асоційовані сховища даних (Data Sources). Кожне таке сховище обслуговує один графік (Graph), причому на цьому графіку може малюватися кілька змінних - сховище для них усіх буде одне. Сховище створюється на основі шаблону даних (Data Template), який задає відповідність вхідних величин (отриманих з SNMP запитів або з скриптів) полям у базі даних і встановлює додаткові параметри зберігання цих величин. Самі ж вхідні величини виходять з методів збору даних (Data Input Methods) або запитів (Data Queries). Перші призначені для величин, кількість яких заздалегідь відомо (наприклад, кількість процесів – це завжди одне ціле число), а другі - навпаки (наприклад, статистика з мережевих інтерфейсів, число яких може бути різним). Графік генерується з кругової бази даних (сховища) кожен раз знову, коли завантажується сторінка. Алгоритм і параметри його створення задаються шаблоном графіка (Graph Template). Шаблони хостів (Host Templates) спрощують роботу з однотипними пристроями і дозволяють

прив'язати певні шаблони графіків і запити до даного типу хоста. Наприклад, для маршрутизаторів Cisco – один набір графіків, а для UNIX-серверів – інший.

Cacti дозволяє завести кілька користувачів і розмежувати їх права як на перегляд статистики, так і на управління системою. Логіка поділу доступу дозволяє для кожного користувача встановити загальну політику («Заборонити» або «Дозволити»), а потім зробити з неї виключення

В результаті Cacti дозволяє малювати графіки тільки основних показників продуктивності, тоді як спроби моніторингу нестандартних показників сильно знижують продуктивність продукту. Також проекту дуже часто необхідна інвентаризація (перерозподіл ресурсів і планування модернізації), а в даному пакеті вона відсутня.

1.7 Порівняльний аналіз вільно розповсюджуваних систем

Таблиці 1.2,1.3 та 1.4 дозволяють оцінити функціонал існуючих вільно розповсюджуваних систем комплексного моніторингу IT-інфраструктури.

Таблиця 1.2 – Функціонал вільно розповсюджуваних систем моніторингу

Назва	Діаграми	Звіти SLA	Логічне угрупування	Події	Прогнозування подій
Cacti	Так	Так	Так	Так	Так
Nagios	Так	Через плагін	Так	Так	Ні
Zabbix	Так	Так	Так	Так	Так

Таблиця 1.3 – Функціонал вільно розповсюджуваних систем моніторингу

Назва	Автоматичне виявлення	Агент	SNMP	Syslog	Зовнішні скрипти
Cacti	Через плагін	Ні	Так	Через плагін	Так
Nagios	Через плагін	Так	Через плагін	Через плагін	Так
Zabbix	Так	Підтримується	Так	Так	Так

Таблиця 1.4 – Функціонал вільно розповсюджуваних систем моніторингу

Назва	Плагіни	Складність створення плагінів	Тригери / Тривоги	Інвентаризація	Карти
Cacti	Так	Середня	Так	Ні	Через плагін (Weathermap)
Nagios	Так	Легка	Так	Через плагін	Динамічні і настроюються
Zabbix	Так	Легка	Так	Через плагін	Так

2 ТЕХНІЧНІ ВИМОГИ ДО КОМПЮТЕРНОЇ СИСТЕМИ

2.1 Основні завдання

На початку розвитку інформаційних технологій навіть дуже великими компаніями використовувалися досить тривіальні рішення: всі системи були настільки спрощеними, що досить було команди фахівців, щоб підтримувати систему в робочому стані. Справді, мінливих складових було занадто мало, тому і необхідність моніторингу.

По мірі ускладнення систем в IT з'явилися різні частини, які з комплексними внутрішніми складовими: системи передачі повідомлень, бази даних, інфраструктурні компоненти, службу каталогів і ін. Всі ці частини потрібно підтримувати в працездатному стані, що досить складно без систем моніторингу, при створенні і використанні яких доведеться зіткнутися з низкою проблем:

– Моніторинг та управління по частинах.

Моніторинг кожної частини, компонента здійснюється спеціалізованим програмним забезпеченням, налаштованим на специфічну задачу. Однак сервіс, який використовується для відстеження однієї становить, не може відстежувати іншу. Це найбільш фундаментальна проблема. Використання окремих, строго специфічних інструментів є фундаментальною проблемою. Це призводить до розгляду кожної з систем в ізоляції – не враховуючи навколишні системи, які також впливають на роботу вихідної. Висновок: спеціалізовані інструменти повинні бути другою лінією моніторингу, для більш детального аналізу при необхідності. Перша ж лінія стеження повинна збирати і надавати огляд звітів інформації комплексно по всіх системах в інфраструктурі «в одному вікні».

– Відсутність зв'язку між користувачем, сервіс-деском і IT-менеджментом

Взаємодія і зв'язок – це ключовий компонент, який робить працездатною будь-яку команду. У разі IT, зазвичай використовуються системи для організацій хелп-деск, маючи на увазі, що це розумний спосіб здійснення комунікацій, але цього не завжди буває достатньо. Системи хелп-деск, як правило, будуються на концепції реагування на проблему і подальшому управлінні реакцією, і за визначенням, вони практично не проактивні.

Керуючі комунікації настільки ж важливі, як і непрості. Забезпечення чесних цифр в рівнях сервісного обслуговування, часу реакції, відсутність сервісів і так далі – це все вкрай важливо, якщо менеджменту необхідно приймати добре опрацьовані рішення в плані IT, але достовірну інформацію часто буває отримати зовсім не просто.

– Моніторинг не того і не там.

Зазвичай IT служба бачить багатокомпонентний, розподілений додаток, при цьому фахівці по кожному компоненту здійснюють моніторинг основних показників: вимірюють їх продуктивність, використовуючи технічні метрики, такі як утилізація процесора, час відповіді і т.д. Коли якийсь з компонентів виходить за допустимі значення, то відповідний інженер починає його детальну перевірку.

Користувачі та IT-служба вимірюють різні речі. У SLA, орієнтованих на IT, може вказуватися конкретний час відповіді на запити, що посилаються на сервер СУБД, але це малокорисні, якщо додаток, з точки зору користувача, працює «повільно». Що ще гірше, якщо починати мігрувати сервіси та компоненти в «хмару», то втрачається велика частина здібностей визначати продуктивність компонентів тим способом, яким звично було це робити в ЦОД до міграції. В результаті жодну зі сторін не влаштують положення такого SLA.

Все це треба міняти, причому з точки зору користувача. Продуктивність індивідуальних компонентів важлива, але тільки в тій мірі, наскільки це впливає на загальну продуктивність, що відчувається на

конкретному робочому місці. Необхідно прописувати такі SLA, в яких і користувач і служба IT знаходяться «на одній сторінці», потім управляти виконанням цих SLA тими способами і інструментарієм, які дозволяють це успішно робити. Деякі організації повідомляють, що вони переходять, або вже перейшли до роботи IT на основі надання сервісів – це, в широкому сенсі слова, означає, що компанія шукає спосіб реалізувати роботу IT служби у вигляді набору сервісів для різних департаментів організації та їх користувачів. Саме до такої моделі слід рухатися службам інформаційних технологій.

– Втрата інформації

Знання про інфраструктуру компанії – і про те, як вирішувати проблеми повинно бути акумульовано і збережено. Дана вимога не тільки допоможе швидше вирішувати виникаючі питання в майбутньому, але також допоможе запобігати їх появі, дозволяючи приймати більш зважені рішення в області IT. В рамках компанії цю проблему зазвичай вирішують створенням централізованої системи бази знань, яка заповнюється фахівцями в міру виникнення і вирішення ними різних інцидентів. Значення, зібрані монітором, також повинні акумулюватися, щоб створювати довготривалі звіти для з'ясування динаміки роботи того чи іншого вузла інфраструктури, щоб використовувати їх, наприклад, для планування розширення пулу ресурсів конкретного сервісу, або для виявлення вузького місця.

2.2 Вимоги до моніторингу

Вибір способів і об'єктів моніторингу залежить від безлічі факторів – зміни мережі, що діють в ній сервісів і служб, конфігурації серверів і встановленого на них програмного забезпечення, можливостей забезпечення, яке використовується для моніторингу і т.п.

На найзагальнішому рівні можна говорити про такі елементи як:

- перевірка фізичної доступності обладнання;
- перевірка стану (працездатності) запущених служб і сервісів;
- детальна перевірка не критичних, але важливих параметрів функціонування: продуктивності, завантаження і т.п .;
- перевірка параметрів, специфічних для сервісів і служб даного конкретного оточення (наявність деяких значень в таблицях БД, вміст лог-файлів).

Проектом, над яким буде здійснювати моніторинг, були висунуті наступні конкретизують уточнення до спільного переліку елементів моніторингу.

До першого пункту можна віднести перевірку доступності в мережі хоста, віртуальної машини. Тут досить тестування наявності з'єднання і графіку швидкості проходження сигналу.

До другого пункту можна віднести графік перевірки завантаженості центрального процесора.

До третього пункту віднесемо графік перевірки використовуваної оперативної пам’яті.

До четвертого пункту віднесемо виведення списку працюючих процесів.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ

3.1 Веб-інтерфейс системи моніторингу

Для того щоб подивитися веб-інтерфейс розробленої системи моніторингу, спочатку необхідно запустити віртуальну машину з встановленою на ній ОС Ubuntu (див.рис. 2.1)

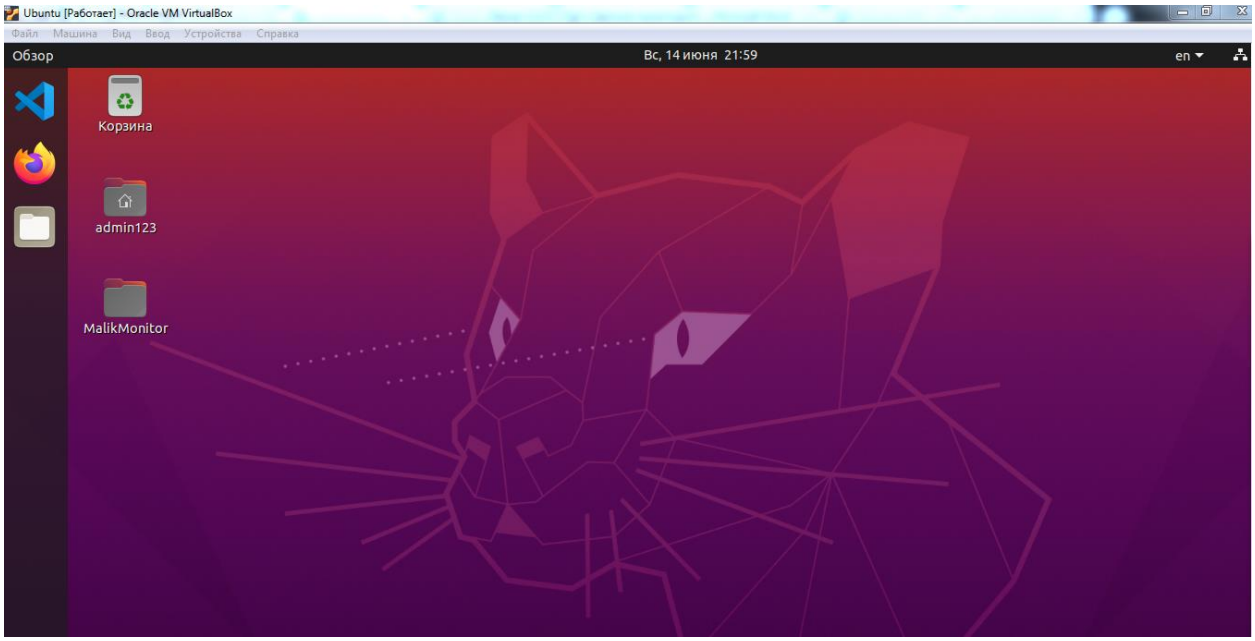

Рисунок 3.1 – Віртуальна машина з встановленою ОС Ubuntu

Далі необхідно відкрити папку MalikMonitor, клацнути в ній правою кнопкою миші та із меню, що з’явилося, обрати «Відкрити в терміналі» (див.рис. 3.2).

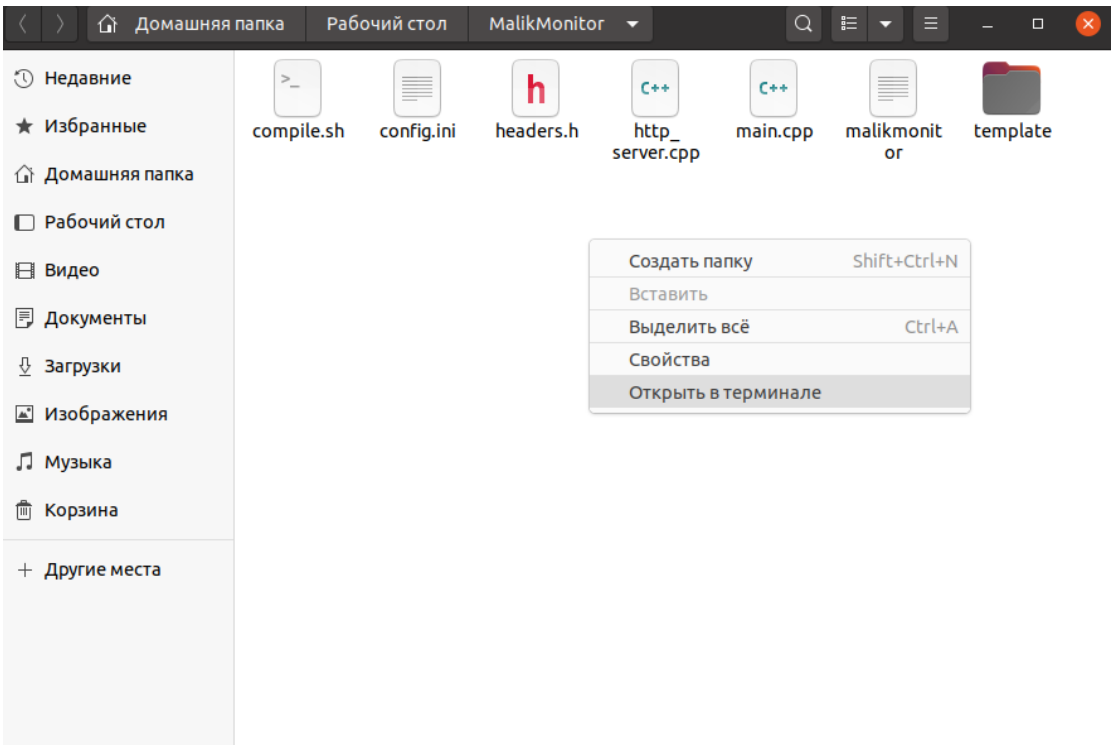

Рисунок 3.2 – Зміст папки «MalikMonitor»

У відкритому терміналі потрібно прописати рядок «./compile.sh» і натискаємо Enter (див.рис. 3.3).

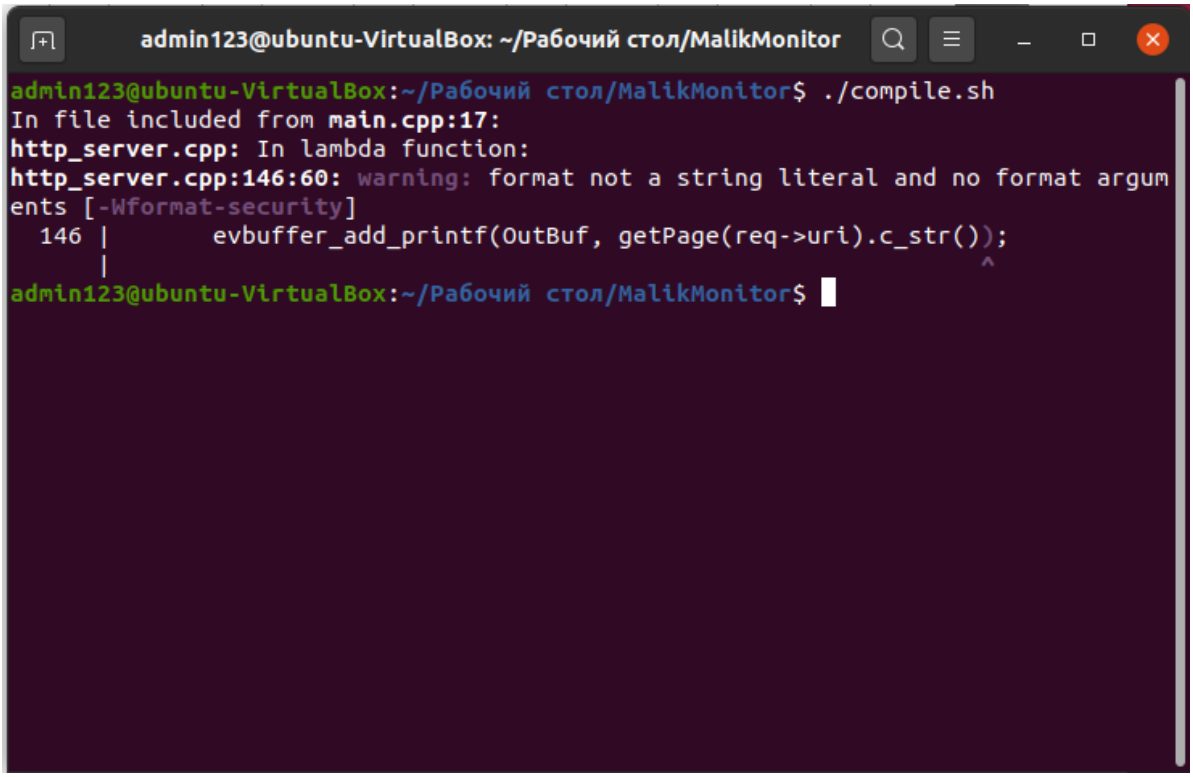

Рисунок 3.3 – Зовнішній вигляд терміналу та команд

Далі необхідно прописати рядок «sudo ./malikmoniror», натиснути Enter, і користувач побачить поле для введення паролю. Після введення паролю «admin123» і натиснення клавіші Enter, робота з терміналом завершується.

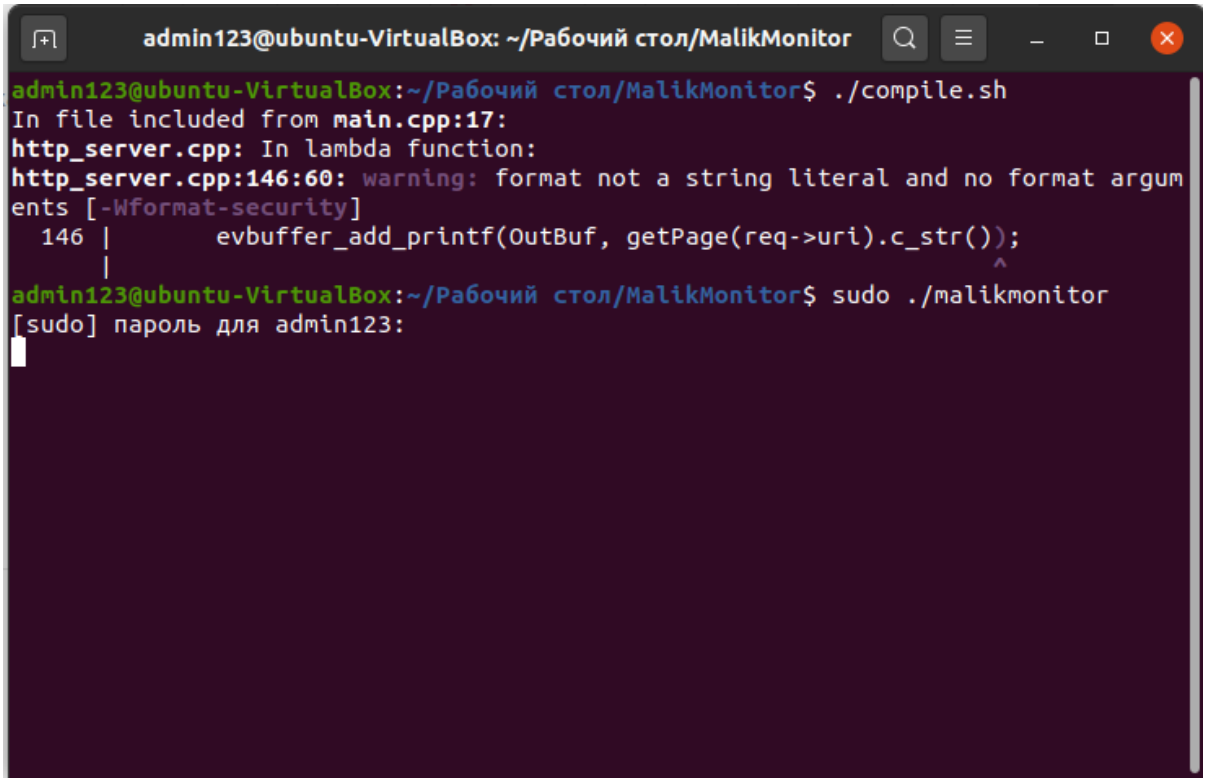

Рисунок 3.4 – Команди, необхідні для запуску системи моніторингу

Після всіх перерахованих вище дій, користувачу потрібно зайти в веб-браузер Firefox і в рядку пошуку прописати «localhost: 44444/». Після завантаження користувач побачить веб-інтерфейс розробленої системи моніторингу (див.рис. 3.5).

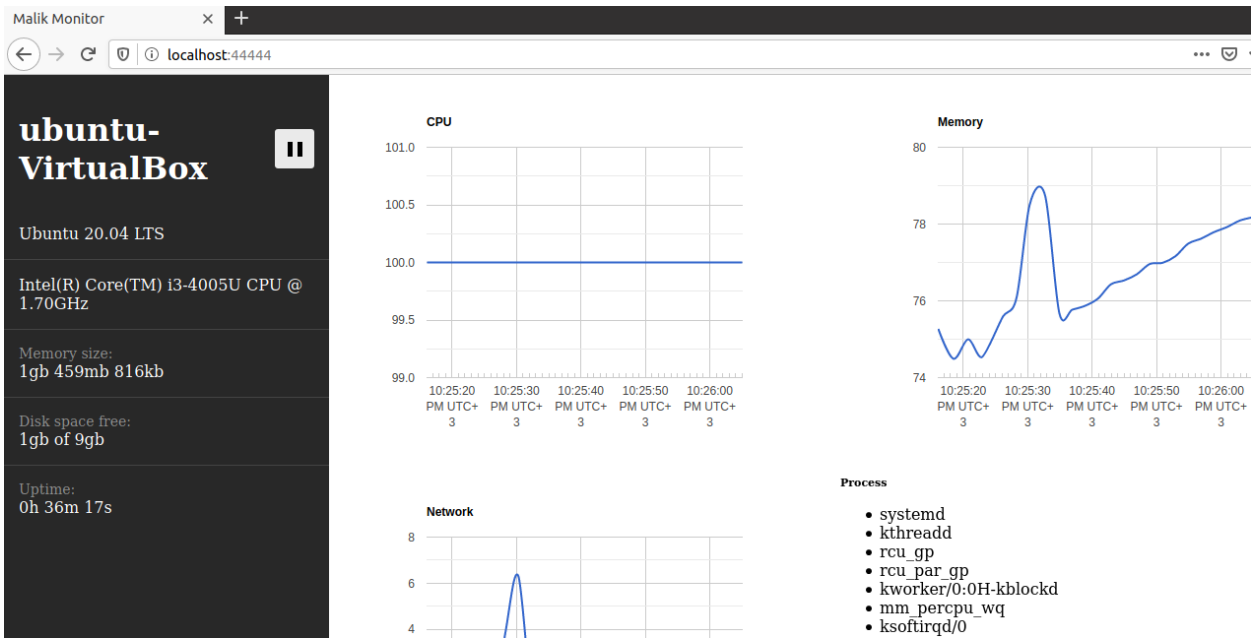

Рисунок 3.5 – Веб-інтерфейс розробленої системи моніторингу стану серверного обладнання

Розглянемо все побачене докладніше. Тут користувач бачить графік завантаження процесора у відсотках (див.рис. 3.6), графік кількості зайнятої оперативної пам'яті у відсотках (див.рис. 3.7), графік інтернет-трафіку в кб/с (див.рис. 3.8) і список працюючих процесів (див.рис. 3.9).

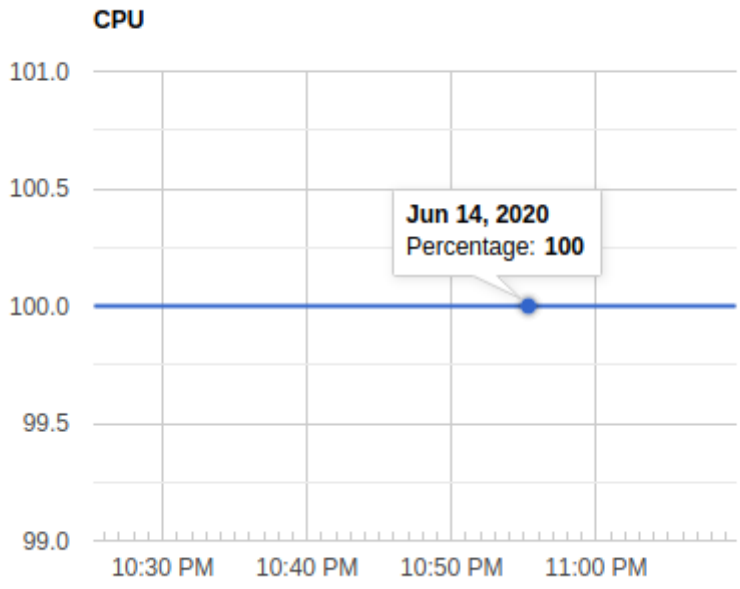

Рисунок 3.6 – Графік завантаження процесора

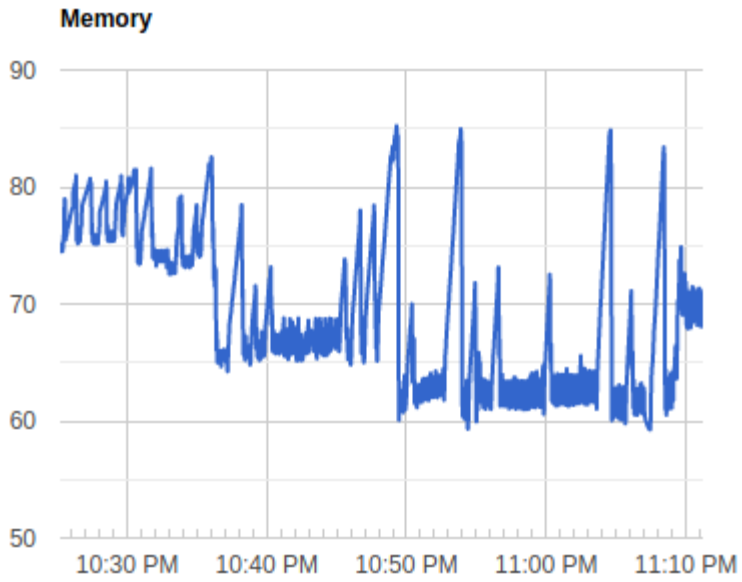

Рисунок 3.7 – Графік використання оперативної пам’яті

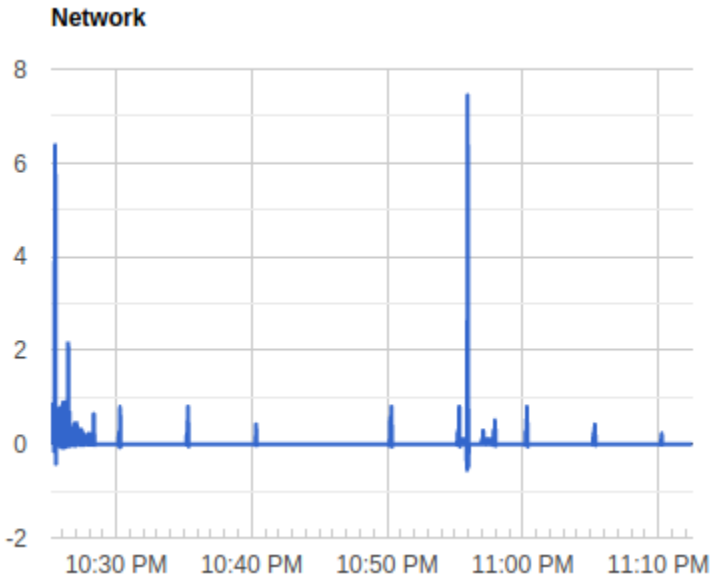

Рисунок 3.8 – Графік інтернет-трафіку

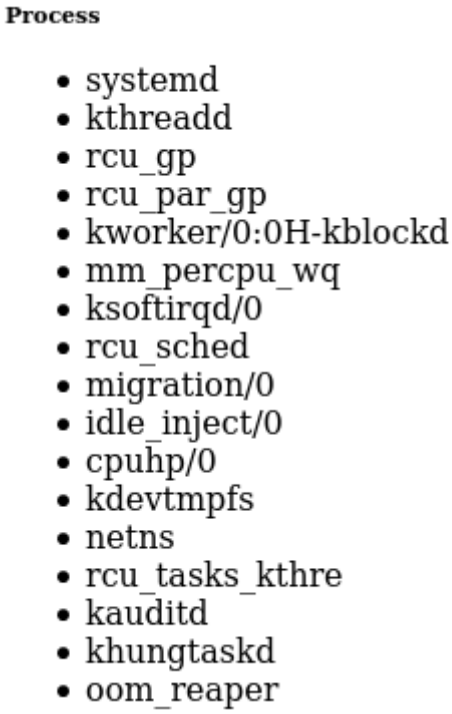

Рисунок 3.9 – Список працюючих процесів

На боковій панелі системи моніторингу розміщена кнопка, що призупиняє або відновлює роботу системи моніторингу, також там знаходяться дані про ОС, процесор, кількість оперативної пам’яті, кількість вільного місця на диску та час роботи сервера без зупинок (див.рис. 3.10).

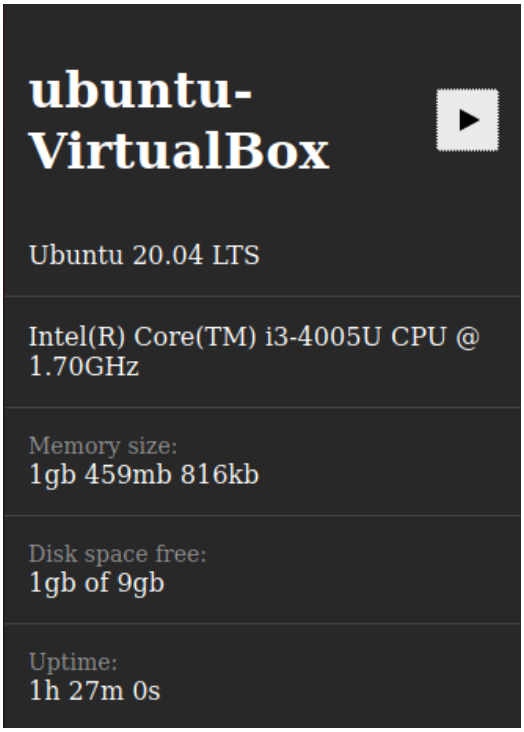

Рисунок 3.10 – Бокова панель системи моніторингу

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Розрахунки витрат на розробку програмного продукту

Витрати на розробку програмного продукту містять витрати на заробітну плату розробника програми $З_{зп}$.

Заробітна плата розробника програмного забезпечення:

$$З_{зп} = t \cdot С_{пр}, \text{ грн.}$$

де t – загальна кількість годин, витрачена на розробку програмного забезпечення;

$С_{пр}$ – середня годинна тарифна ставка програміста становить:

$С_{пр}$ = 69 грн./година.

Заробітна плата за розробку програмного забезпечення дорівнює:

$$З_{зп} = 60 \cdot 69 = 4140 \text{ грн.}$$

Висновок

Розробка комп’ютерної системи моніторингу стану серверного обладнання підприємства ФОП «Коваленко Е.В.» з опрацюванням побудови, налаштування та безпеки корпоративної мережі доцільно, так як при мінімальних витратах на заробітну плату програміста в 4140грн., підприємство отримує самостійну працюючу систему моніторингу, яка повністю відповідає вимогам.

5 ОХОРОНА ПРАЦІ

Виробниче середовище впливає на здоров’я робітника шкідливими і небезпечними факторами. Шкідливий виробничий фактор – це виробничий фактор, вплив якого па працівника може привести його до захворювання. Небезпечний виробничий фактор – це виробничий фактор, вплив якого на працівника може привести до його травми.

В кваліфікаційній роботі розглядається удосконалення комп’ютерної системи ФОП «Коваленко Е.В.». Об'єктом для аналізу небезпечних і шкідливих факторів взято приміщення з персональними комп'ютерами.

5.1 Аналіз шкідливих і небезпечних вражаючих факторів

У приміщенні на працівника можуть впливати небезпечні і шкідливі виробничі фактори, які вказані в таблиці 5.1.

Таблиця 5.1 – Аналіз шкідливих і небезпечних вражаючих факторів

№	Найменування ШНВФ	Джерела ШНВФ	Нормуючий документ
1	Можливість ураження електричним струмом	Електропроводка, блок живлення персонального комп’ютера	ДЕСТ 12.1.038-82. Електробезпека. Гранично допустимі значення напруг дотику і струмів
2	Тепловиділення від устаткування	Персональний комп'ютер, периферійні пристрої	СанПиН 2.2.4.548-96
3	Недостатня освітленість робочої зони	Робоче місце	СНіП II-4-79

Продовження таблиці 5.1

№	Найменування ШНВФ	Джерела ШНВФ	Нормуючий документ
4	Специфічний характер зорової роботи; вимушена локалізована поза	Монітор, робоче місце оператора ПЕОМ	СанПин 2.2.2/2.4.1340-03

5.2 Інженерно-технічні заходи з охорони праці

5.2.1 Заходи щодо забезпечення електробезпеки

Відповідно до класифікації ПУЕ за небезпекою ураження електричним струмом приміщення операторів ПК відноситься до приміщень з підвищеною небезпекою, так як існує можливість одночасного дотику до опалювальних батарей приміщення, з'єднаних з землею і корпусів електрообладнання. В операторській використовується обладнання з напругою живлення 220 В. Лінія електромережі для живлення ПК та периферійного обладнання виконується як окрема групова трьохпровідна мережа шляхом прокладки фазного і нульового робочого та захисного провідників. Нульовий захисний провідник служить для занулення електроприймачів. При напрузі до 1000 В застосовують трьохпровідний мережу з ізольованою нейтраллю.

Електропроводка в операторській повинна бути виконана прихованим методом, прокладена в гнучких металорукавах, що робить силові ланцюги недоступними для працюючих.

Струмовий захист реалізується з використанням автоматів, які розривають електричну мережу при високих струмах навантаження. Для забезпечення захисного відключення використовуємо УЗО ВД1-63, основним призначенням якого є забезпечення безпеки людини в разі дотику до занулення (заземленому) корпусу при замиканні на нього фази, а також при безпосередньому дотику до струмоведучих частини електроустановки.

Основні заходи, спрямовані на попередження випадків ураження електричним струмом в операторській, такі:

- щодня проводити очистку монітора від пилу;

- забороняється знімати захисну кришку системного блоку комп'ютера;
- усунення можливості випадкового дотику до струмоведучих частин електроустаткування, що знаходиться під напругою;
- малі напруги;
- надійна ізоляція струмоведучих частин електрообладнання і своєчасний його ремонт;
- захисне занулення;
- захисне відключення та застосування плавких запобіжників.

5.2.2 Заходи щодо захисту від підвищеної температури

При виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням в операторських приміщеннях повинні дотримуватися оптимальні умови мікроклімату (температура повітря 22 - 24 °С, відносна вологість 60 - 40%, швидкість руху повітря не більше 0,1 м/сек.).

Кімната операторів ПК повинна бути обладнана власною системою вентиляції та кондиціонування на базі спліт-системи LG A12LHR продуктивністю 510 м3 /год, яка стабілізує температуру повітря в регульованих межах 14 ... 32 ± 2ºС.

5.2.3 Освітлення робочої зони

В адміністративній будівлі в приміщенні операторів ПК використовується поєднане природне і штучне освітлення. Згідно СНиП 23.05- 95 – середня точність зорової роботи, найменший розмір об'єкта розрізнення складає 0,3÷0,5 мм.

Штучне освітлення може бути двох систем – загальне і комбіноване. В операторському пункті використана система комбінованого освітлення, тобто до загального освітлення додано місцеве, створюване світильниками, концентрує світловий потік безпосередньо на робочих місцях.

Для освітлення приміщення використані, найбільш економічні люмінесцентні лампи типу ЛБ. Для місцевого освітлення використані лампи розжарювання.

На робочому місці відсутні різкі тіні.

Для внутрішнього оздоблення інтер'єру приміщень з ПК використані диффузионно-відбивні матеріали з коефіцієнтами відбиття світла для стелі 0,7÷0,8; для стін 0,5÷0,6; для підлоги 0,3÷0,5.

5.2.4 Заходи по боротьбі з шкідливими факторами при роботі з персональним комп'ютером

Розробники програмного забезпечення за характером роботи багато часу проводять з ПК. В даному випадку працівник схильний до впливу напруги зору. Це призводить до підвищеного стомлення зору і загального стомлення.

Для безпечної і комфортної роботи при експлуатації персонального комп'ютера розроблені наступні заходи:

1. Площа на одне робоче місце має становити не менше 6 м2 , об'єм – не менше 20 м3 .

2. Висота робочої поверхні для монітора повинна становити 680-800 мм (рекомендовані розміри столу: висота – 725 мм, ширина - 600-1400 мм, глибина – 800-1000 мм).

3. Робоче сидіння працівника має складатися з: сидіння, спинки і підлокітників.

4. Монітор і клавіатура повинні бути розміщені на поверхні столу або на спеціальній робочій поверхні окремо від столу, яка повинна бути відрегульована по висоті, на відстані 100-300мм від краю.

5. Щоб освітлення не створювало сліпучих відблисків, комп'ютер повинен бути розташований так, щоб пряме світло не попадало на екран.

6. Верхній край екрана слід розташовувати на рівні очей або трохи нижче.

7. Оптимальна відстань від очей до екрана 600-700 мм.

8. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5; поверхня підлоги повинна бути рівною, неслизькою.

9. Вологе прибирання повинна проводитися на початку робочого дня, а також під час перерви.

5.3 Пожежна профілактика

Приміщення операторів ПК із вибухопожежної безпеки відноситься до категорії В4 (приміщення закритих розподільних пристроїв, з вимикачами і електронною апаратурою). Вогнестійкість приміщення визначається по таблиці меж вогнестійкості будівельних конструкцій - 3 ступінь вогнестійкості.

В операторській можливі три джерела виникнення пожежі:

- випадкове замикання в електропроводці;
- необережне поводження з нагрівальними приладами;
- несправність або перегрів комп'ютерної техніки та електронної апаратури.

Організаційними заходами щодо забезпечення пожежної безпеки є навчання працівників правилам пожежної безпеки; розробка і реалізація норм і правил пожежної безпеки, інструкцій про порядок роботи з пожежонебезпечними речовинами і матеріалами.

Для забезпечення пожежної безпеки необхідно:

- забезпечити електробезпеку згідно п. 5.2;
- встановити пристрої автоматичної протипожежної сигналізації та пожежогасіння виробництва СКБ «Електронмаш» ППКП «Варта 8/32» (призначений для прийому сигналів від автоматичних і ручних пожежних сповіщувачів та видачі інформації на

сповіщувачі, а також включення ланцюгів управління установками автоматичних систем пожежогасіння). В якості технічних засобів виявлення пожежі використані автоматичні пожежні сповіщувачі теплової ІПК-9, і димової ІПК-4, а також ручні пожежні сповіщувачі ІПР-1К. Як засоби оповіщення про пожежу застосований оповіщувач комбінований (світлозвуковий) УЗС-1;

- встановити резервний ручної пускач системи пожежогасіння;
- встановити план евакуації при пожежі.

Для операторської для гасіння пожеж застосовні вуглекислотні вогнегасники ОУ-80 ГОСТ 9230-77, Торжокское ПО ППТ - 2 шт., Комбінований вогнегасник ОК-100.01 ТУ 22-4614-50, Торжокское ПО ППТ - 1 шт.

5.4 Заходи з ергономіки

Велике значення в створенні оптимальних умов праці має планування робочого місця, яка повинна задовольняти вимогам зручності виконання робіт, економії енергії і часу розробника програмного забезпечення.

Сидяча тривала робота шкідлива людині в принципі: працівник сутулиться або подається вперед і його хребет деформується, травмуючи диски; він піднімає плечі і згинає руки, тримаючи їх в напрузі – і природно вони починають хворіти. Перетискаючи судини, він перевантажує серце; ну а про хронічні розтягненнях сухожиль кистей рук і постійно погіршується зір можна не говорити. Поза, а отже і здоров'я, залежать, в кінцевому підсумку, від розмірів і дизайну робочого місця.

Важливим аспектом ергономіки є методи проектування робочих місць. Наукова організація робочого простору базується на даних про середній зоні охоплення рук людини – 35-40 см. Ближній зоні відповідає область, що охоплюється рукою з притиснутим до тулуба ліктем, далекій зоні – область витягнутої руки. При компонуванні робочого місця не слід забувати про те,

що найбільш важливі з знарядь праці слід розташовувати попереду і праворуч від людини. Клавіатура, як найбільш часто використовуваний пристрій введення. Параметри цієї зони: кут – 70 °, глибина – 30-40 см. Решта пристроїв – кут – 130 градусів, глибина 70-80 см.

Серед робочих столів найбільш ергономічної визнана криволінійна кутова форма. За рахунок угнутості велика частина їх площі виявляється використовуваною, оскільки потрапляє в зону охоплення руками людини, рівну 35-40 см. Площа стільниці хорошого столу не може бути менше 1 $м^2$. Висота від підлоги до стільниці, як правило, повинна дорівнювати рекомендованим європейськими нормами 74 см.

Для повноцінної роботи розробника також необхідні навісні полиці або тумбочками на колесах, які допоможуть організувати робоче місце за принципом «все під рукою».

Основним робочим положенням розробника є положення сидячи. Конструкція робочого стільця (крісла) повинна забезпечувати підтримку раціональної робочої пози під час роботи на ПК. Робочий стілець (крісло) повинен бути підйомно-поворотним і регульованим по висоті і кутам нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння.

Забарвлення приміщень і меблів повинна сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, в яких виконується одноманітна розумова робота, що потребує значної нервової напруги і великого зосередження, фарбування повинна бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

ВИСНОВКИ

Бізнес в будь-якій сфері сильно зав'язаний на доступності і працездатності його IT-інфраструктури 24/7/365. Щоб забезпечити цю працездатність, необхідно заздалегідь виявляти вузькі місця в конфігурації систем і мереж, а також швидко дізнаватися про наявність поломки і її причини. Для цих потреб в компаніях, де подібне стеження нездійсненно за рахунок тільки фахівців, прийнято використовувати системи моніторингу.

Системи моніторингу бувають платні та вільно поширювані, а також розрізняються за своїм наповненням «з коробки», масштабованості, необхідних ресурсів і рівнем знань, необхідним для прийнятного налаштування.

При виборі, розробці, впровадженні систем моніторингу спочатку потрібно визначитися з об'єктами, які будуть піддаватися стеженню, а також критичними подіями і показниками, які і визначать кількість повідомлень при поломці, частоту сканування і інші параметри і наслідки. Причому оцінювання показників в першу чергу потрібно здійснювати не з точки зору технічного інженера, а з точки зору кінцевого користувача.

Хід виконання роботи повністю відповідав поставленим у вступі завданням. Для початку було висунуто загальні вимоги до моніторингу підтримуваних проектів, володіючи списком яких можна було приступити до пошуку відповідного готового рішення.

Потім докладно розглянуті різні існуючі системи моніторингу серверів від широкого різноманіття фірм-розробників, проаналізовано їх позитивні і негативні сторони, відповідність їх сформульованим раніше вимогам.

У підсумку, зроблено висновок про неможливість або неоптимальність використання вищерозглянутих існуючих систем. Після чого критерії відповідності моніторингу поставленим завданням в проекті були максимально уточнені, завдяки чому став можливий вибір оптимальної

основи для його створення - вже використовуваного для інших потреб розробки пакету MalikMonitor.

По мірі виникнення проблем і нових завдань, в проекті розроблялися різноманітні модулі, вівся пошук додаткових бібліотек і способів інтеграції з існуючими сервісами як для максимальної зручності написання скриптів, так і для відстеження специфічних показників. Паралельно здійснювалося поетапне тестування створених скриптів і їх впровадження в експлуатацію. У підсумку, розроблений моніторинг став невідривної частиною підтримуваних проектів, що забезпечує високу доступність, відмовостійкість і інформацію про стан систем.

В цілому, пропонований продукт для моніторингу серверів допоможе значно знизити час простою сервісів, підвищити якість послуг, що надаються клієнту компанією, а також значно поліпшити продуктивність, уникнувши, при цьому, невиправданих вкладень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 7 безкоштовних програм для моніторингу мережі та серверів [Електронний ресурс] https://networkguru.ru/monitoring-seti-setevogo-oborudovaniia-serverov/
2. Більш ніж 80 засобів моніторингу системи Linux [Електронний ресурс] https://habr.com/ru/company/ua-hosting/blog/281519/
3. Топ-10 кращих программ для моніторингу мережі [Електронний ресурс] https://www.softinventive.ru/best-network-monitoring-tools/
4. Nagios [Електронний ресурс]: Nagios Documentation – http://www.nagios.org/
5. .Zabbix [Електронний ресурс]: What is Zabbix – http://www.zabbix.com
6. Вікіпедія [Електронний ресурс]: Моніторинг серверів - http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0% BE%D1%80%D0%B8%D0%BD%D0%B3_%D1%81%D0%B5%D1%80%D0%B2% D0%B5%D1%80%D0%BE%D0%B2
7. Моніторинг мережі Linux [Електронний ресурс] – https://losst.ru/monitoring-seti-linux
8. Свой http-сервер менее чем в 40 строк кода на libevent и C++11 [Електронний ресурс] – https://habr.com/ru/post/217437/
9. Захват пакетов при помощи библиотеки libpcap [Електронний ресурс] – http://rus-linux.net/MyLDP/algol/libpcap.html
10. Написание своего HTTP сервера с использованием libevent [Електронний ресурс] – http://incpp.blogspot.com/2009/05/http-libevent.html
11. Вікіпедія [Електронний ресурс]: Порівняння систем моніторингу мережі – http: //ru.wikipedia.org

12. Хабрахабр [Електронний ресурс]: Що таке моніторинг в IT - Режим доступу: http://habrahabr.ru/company/croc/blog/144941/
13. Хабрахабр [Електронний ресурс]: Практичний моніторинг – http://habrahabr.ru/post/145923/
14. IT-послуги системного адміністратора [Електронний ресурс]: IT-послуги для серверів http://www.server-support.com.ua/IT- % D1% 83% D1% 81% D0% BB% D1% 83% D0% B3% D0% B8 /% D1% 81% D0% B5% D1% 80% D0% B2% D0% B5% D1% 80% D1% 8B /

ДОДАТОК А. ЛІСТИНГ ФАЙЛУ main.cpp

```
#include <iostream>
#include <sys/sysinfo.h>
#include <fstream>
#include <string>
#include <vector>
#include <unistd.h>
#include <sys/utsname.h>
#include <sys/statvfs.h>
#include <pcap.h>
#include <cstring>
#include <thread>
#include <dirent.h>
#include <algorithm>

#include "headers.h"

#include "http_server.cpp"

using namespace std;

//данная функция вызывается при появлении пакета (трафика) по сети
void networkPacket(u_char *useless,const struct pcap_pkthdr* pkthdr,const
u_char* packet)
{
   network_len += pkthdr->len;
}

//данная функция запускает мониторинг сети, а именно отслеживает пакеты
трафика
void getNetworkPackets() {
   char *dev;
   char errbuf[PCAP_ERRBUF_SIZE];
   pcap_t* descr;
```

```
struct bpf_program fp;       /* to hold compiled program */
bpf_u_int32 pMask;           /* subnet mask */
bpf_u_int32 pNet;           /* ip address*/
pcap_if_t *alldevs, *d;
char dev_buff[64] = {0};
int i =0;

// Prepare a list of all the devices
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
   fprintf(stderr,"Error in pcap_findalldevs: %s\n", errbuf);
   exit(1);
}

dev = alldevs->name;

// If something was not provided
// return error.
if(dev == NULL)
{
   printf("\n[%s]\n", errbuf);
   exit(1);
}

// fetch the network address and network mask
pcap_lookupnet(dev, &pNet, &pMask, errbuf);

// Now, open device for sniffing
descr = pcap_open_live(dev, BUFSIZ, 0,-1, errbuf);
if(descr == NULL)
{
   printf("pcap_open_live() failed due to [%s]\n", errbuf);
   exit(1);
}

// Compile the filter expression
```

```
    if(pcap_compile(descr, &fp, "tcp", 0, pNet) == -1)
    {
        printf("\npcap_compile() failed\n");
        exit(1);
    }

    // Set the filter compiled above
    if(pcap_setfilter(descr, &fp) == -1)
    {
        printf("\npcap_setfilter() failed\n");
        exit(1);
    }

    // For every packet received, call the callback function
    pcap_loop(descr,-1, networkPacket, NULL);
}

//данная функция возврщает все запущенные процессы
vector<string> getProcess() {
    vector<string> process;
    DIR *dir;
    if ((dir = opendir ("/proc/")) != NULL) {
        struct dirent *ent;
        while ((ent = readdir (dir)) != NULL) {
            string name = ent->d_name;
            ifstream stream;
            stream.open("/proc/" + name + "/comm");
            string s; getline(stream,s);
            process.push_back(s);
        }
        closedir (dir);
    }
    return process;
}

//основная функция которая запускается при старте программы
```

```
int main() {
        //стандартный порт, на котором работает веб-сервер для веб-интерфейса
   string http_server_port = "4444";
        //стандартное количество потоков веб-сервера
   string http_server_threads = "4";
        //начало чтения с файла настроек config.ini,
        //мы считываем два параметра это порт и количество потоков для веб сервера,
        //если не будет указано, то это не беда, будут использоваться стандартные и наличие файла тоже не обязательно
   ifstream in("config.ini");
   if(in.is_open()) {
      string line;
      while(getline(in, line)) {
         if(line.find("PORT")==0) {
            http_server_port = line.substr(5);
         } else if(line.find("THREADS")==0) {
            http_server_threads = line.substr(8);
         }
      }
   }
   in.close();
        //конец чтения с файла настроек

        //создание и запуск в фоне потока для мониторинга трафика сети
   thread networkPackets(getNetworkPackets);
   networkPackets.detach();
        //запуск веб сервера
   http_server(atoi(http_server_port.c_str()),atoi(http_server_threads.c_str()));
   return 0;
}

//данная функция возврщает hostname
string getHostname() {
   ifstream stream("/proc/sys/kernel/hostname");
```

```
    string str;
    getline(stream,str);
    stream.close();
    return str;
}

//данная функция возвращает название процессора
string getCpu() {
    ifstream stream("/proc/cpuinfo");
    string str;
    for(int i = 0; i < 16;i++) stream >> str;
    getline(stream,str);
    stream.close();
    return str;
}

//данная функция возвращает количество оперативной памяти в байтах
string getMemory() {
    ifstream stream("/proc/meminfo");
    string str;
    stream >> str; stream >> str;
    stream.close();
    return str;
}

//данная функция вовращает количество оперативной памяти в виде 1gb
200mb 300kb (1 гиг 200 мегабайт 300 килобайт)
string getMemoryView() {
    ifstream stream("/proc/meminfo");
    string str;
    stream >> str; stream >> str;
    stream.close();
    int num = atoi(str.c_str());
    int percent = num / 100;
    int gb = (num / 1024) / 1024;
    int mb = (num-gb*1024*1024) /1024;
```

```
    int kb = (num - (gb*1024*1024+mb*1024));
    return to_string(gb)+"gb "+to_string(mb)+"mb "+to_string(kb)+"kb ";
}

//данная функция возвращает сколько используется сейчас оперативной
памяти
string getMemoryUsed() {
    ifstream stream("/proc/meminfo");
    string str;
    stream >> str; stream >> str;
    int num = atoi(str.c_str());
    int free = 0;
    for (int i = 0 ; i < 3 ; i++) {
        stream >> str; stream >> str; stream >> str;
        free += atoi(str.c_str());
    }
    num -= free;
    stream.close();
    return to_string(num);
}

//данная фукнция возвращает сколько времени работает сервер с момента
последнего выключения системы
string getUptime() {
    struct sysinfo o;
    sysinfo(&o);
    long up = o.uptime;
    return to_string(up);
}

//данная фукнция возвращает сколько времени работает сервер с момента
последнего выключения системы в виде 1h 20m 30s (1 час 20 минут 30 сек)
string getUptimeView() {
    struct sysinfo o;
    sysinfo(&o);
    long up = o.uptime;
```

```
    int hour = (up/60/60);
    int min = (up - hour*60*60) / 60;
    int sec =  ((up - hour*60*60) - min*60);
    return to_string(hour)+"h "+to_string(min)+"m "+to_string(sec)+"s";
}

//данная функция возвращает текущее время исполнения процессора
vector<float> readCpuStats() {
    vector<float> ret;
    ifstream stat_file("/proc/stat");
    if (!stat_file.is_open())
    {
        cout << "Unable to open /proc/stat" << std::endl;
        return ret;
    }
    int val;
    string tmp;
    stat_file >> tmp;
    for (int i = 0; i < 4; ++i)
    {
        stat_file >> val;
        ret.push_back(val);
    }
    stat_file.close();
    return ret;
}

//данная функция возвращает в процентах сколько используется процессора
//для измеренеия сколько процентов, получает значение времени
исполнения процессора и через пол секунды следующее, дальше путем
вычислений уже получаем фактичускую нагрузку в процентах
float getCpuLoad()
{
    vector<float> stats1 = readCpuStats();
    usleep(500000);
    vector<float> stats2 = readCpuStats();
```

```
    int size1 = stats1.size();
    int size2 = stats2.size();
    if (!size1 || !size2 || size1 != size2) return 2;
    for (int i = 0; i < size1; ++i) stats2[i] -= stats1[i];
    int sum = 1;
    for (int i = 0; i < size1; ++i) sum += stats2[i];
    float load = 100 - (stats2[size2 - 1] * 100 / sum);
    return load;
}

//данная функция возвращает название ОС
string getDistribName(){
    ifstream stream("/etc/os-release");
    string str;
    getline(stream,str);
    str = str.substr(6,str.find("\"")+1);
    stream.close();
    return str;
}

//данная функция возвращает версию ОС
string getDistribVer(){
    ifstream stream("/etc/os-release");
    string str;
    getline(stream,str);
    getline(stream,str);
    str = str.substr(9,str.find("\"")+1);
    stream.close();
    return str;
}

//возвращает общий объем жесткого диска
long getDiskTotalSpace()
{
  struct statvfs stat;
```

```
  if (statvfs("/", &stat) != 0) {
    // error happens, just quits here
    return -1;
  }

  return stat.f_blocks * stat.f_bsize;
}

//возвращает количество свободного места на жестком диске
long getDiskFreeSpace()
{
  struct statvfs stat;

  if (statvfs("/", &stat) != 0) {
    return -1;
  }

  return stat.f_bfree * stat.f_bsize;
}
```

ДОДАТОК Б. ЛІСТИНГ ФАЙЛУ http_server.cpp

```
#include <stdexcept>
#include <iostream>
#include <memory>
#include <chrono>
#include <thread>
#include <cstdint>
#include <vector>
#include <string>
#include <fstream>
#include <evhttp.h>

#include <chrono>
#include <thread>

#include "headers.h"

using namespace std;

//функция заменяет в строке один подтекст на другой
bool replace(std::string& str, const std::string& from, const std::string& to) {
   size_t start_pos = str.find(from);
   if(start_pos == std::string::npos)
      return false;
   str.replace(start_pos, from.length(), to);
   return true;
}

//данная функция перед тем как отдать клиенту веб сервера (посетителю
сайта) страницу, заменяет на ней нужный текст на данные с сервера
string pageReplace(string html) {
   replace(html,"%%hostname%%",getHostname());
   replace(html,"%%cpu%%",getCpu());
   replace(html,"%%memory%%",getMemory());
   replace(html,"%%memory-view%%",getMemoryView());
   replace(html,"%%memory-used%%",getMemoryUsed());
   replace(html,"%%uptime%%",getUptime());
   replace(html,"%%uptime-view%%",getUptimeView());
   replace(html,"%%distrib-name%%",getDistribName());
```

```
   replace(html,"%%distrib-ver%%",getDistribVer());
   replace(html,"%%disk-total-space%%",to_string(getDiskTotalSpace()));
   replace(html,"%%disk-total-space-
view%%",to_string(getDiskTotalSpace()/1024/1024/1024)+"gb");
   replace(html,"%%disk-free-space%%",to_string(getDiskFreeSpace()));
   replace(html,"%%disk-free-space-
view%%",to_string(getDiskFreeSpace()/1024/1024/1024)+"gb");
   return html;
}

//данная функция при запросе на веб сервере данных, отдает данные с
сервера в зависимости от запроса
string getData(string get) {
  using namespace std::this_thread; // sleep_for, sleep_until
  using namespace std::chrono; // nanoseconds, system_clock, seconds

  network_len = 0;
   string html = "";
   if(get=="/get=stats") {//проверяет совпадает ли запрос, если да то отдает
такие данные, иначе идет дальше
     sleep_for(nanoseconds(10));
     sleep_until(system_clock::now() + seconds(1));
     html = "{";
     html += "\"cpu_load\": " + to_string(getCpuLoad()) + ",";
     html += "\"memory\": " + getMemory() + ",";
     html += "\"memory_used\": " + getMemoryUsed() + ",";
     html += "\"disk_free_space\": " + to_string(getDiskFreeSpace()) + ",";
     html += "\"disk_free_space_view\": \"" +
to_string(getDiskFreeSpace()/1024/1024/1024) + "gb\",";
     html += "\"network_traffic\": " + to_string(network_len) + ",";
     html += "\"process\": [";
     for (const auto &s : getProcess()) {
       if(s.length()>0 && s.find("loop")!=0 && s!="malikmonitor") {
         html += "\""+s+"\",";
       }
     }
     html = html.substr(0,html.length()-1);
     html += "]}";
   } else if(get=="/get=disk-free-space") {
      html = to_string(getDiskFreeSpace());
   } else if(get=="/get=disk-free-space-view") {
      html = to_string(getDiskFreeSpace()/1024/1024/1024)+"gb";
```

```
  } else if(get=="/get=disk-total-space") {
    html = to_string(getDiskTotalSpace());
  } else if(get=="/get=disk-total-space-view") {
    html = to_string(getDiskTotalSpace()/1024/1024/1024)+"gb";
  } else if(get=="/get=hostname") {
     html = getHostname();
  } else if(get=="/get=cpu") {
     html = getCpu();
  } else if(get=="/get=memory") {
     html = getMemory();
  } else if(get=="/get=memory-view") {
     html = getMemoryView();
  } else if(get=="/get=memory-used") {
     html = getMemoryUsed();
  } else if(get=="/get=uptime") {
     html = getUptime();
  } else if(get=="/get=uptime-view") {
     html = getUptimeView();
  } else if(get=="/get=cpu-load") {
     html = to_string(getCpuLoad());
  } else if(get=="/get=distrib-name") {
    html = getDistribName();
  } else if(get=="/get=distrib-ver") {
    html = getDistribVer();
  } else if(get=="/get=network-traffic") {
   sleep_for(nanoseconds(10));
   sleep_until(system_clock::now() + seconds(1));
   html = to_string(network_len);
  } else if(get=="/get=process") {
     html = "{\"process\": [";
     for (const auto &s : getProcess()) {
       if(s.length()>0 && s.find("loop")!=0 && s!="malikmonitor") {
         html += "\""+s+"\",";
       }
     }
     html = html.substr(0,html.length()-1);
     html += "]}";
  }
  return html;
}

//функция отдает страницу при запросе на веб-сервер
```

```
string getPage(char *uri) {
   string html;
   string path = uri;
   if(path.find("/get=")==0) {
      html = getData(path);
   } else {
      if(path=="/") {
         path = "./template/index.html";
      } else {
         path = "./template"+path;
      }
      ifstream stream(path);
      string str;
      while(getline(stream,str))
         html += str;
      stream.close();
      if(path.find(".html")) {
         html = pageReplace(html);
      }
   }
   return html;
}

//основная функция веб-сервера, она его запускает и исполняет
void http_server(uint16_t SrvPort,int const SrvThreadCount)
{
 char const SrvAddress[] = "127.0.0.1";//айви-адрес веб-сервера
 try
 {
        //функция первоначально обрабатывает запрос поступивший на веб-
сервер и отдает пользователю ответ
   void (*OnRequest)(evhttp_request *, void *) = [] (evhttp_request *req, void *)
   {
    auto *OutBuf = evhttp_request_get_output_buffer(req);
    if (!OutBuf)
     return;
    evbuffer_add_printf(OutBuf, getPage(req->uri).c_str());
    evhttp_send_reply(req, HTTP_OK, "", OutBuf);
   };
        //запуск веб сервера и его работа
   std::exception_ptr InitExcept;
   bool volatile IsRun = true;
```

```
  evutil_socket_t Socket = -1;
  auto ThreadFunc = [&] ()
  {
   try
   {
    std::unique_ptr<event_base, decltype(&event_base_free)>
EventBase(event_base_new(), &event_base_free);
    if (!EventBase)
     throw std::runtime_error("Failed to create new base_event.");
    std::unique_ptr<evhttp, decltype(&evhttp_free)>
EvHttp(evhttp_new(EventBase.get()), &evhttp_free);
    if (!EvHttp)
     throw std::runtime_error("Failed to create new evhttp.");
     evhttp_set_gencb(EvHttp.get(), OnRequest, nullptr);
    if (Socket == -1)
    {
     auto *BoundSock = evhttp_bind_socket_with_handle(EvHttp.get(),
SrvAddress, SrvPort);
     if (!BoundSock)
      throw std::runtime_error("Failed to bind server socket.");
     if ((Socket = evhttp_bound_socket_get_fd(BoundSock)) == -1)
      throw std::runtime_error("Failed to get server socket for next instance.");
    }
    else
    {
     if (evhttp_accept_socket(EvHttp.get(), Socket) == -1)
      throw std::runtime_error("Failed to bind server socket for new instance.");
    }
    for ( ; IsRun ; )
    {
     event_base_loop(EventBase.get(), EVLOOP_NONBLOCK);
     std::this_thread::sleep_for(std::chrono::milliseconds(10));
    }
   }
   catch (...)
   {
    InitExcept = std::current_exception();
   }
  };
  auto ThreadDeleter = [&] (std::thread *t) { IsRun = false; t->join(); delete t; };
  typedef std::unique_ptr<std::thread, decltype(ThreadDeleter)> ThreadPtr;
  typedef std::vector<ThreadPtr> ThreadPool;
```

```
    ThreadPool Threads;
    for (int i = 0 ; i < SrvThreadCount ; ++i)
    {
      ThreadPtr Thread(new std::thread(ThreadFunc), ThreadDeleter);
      std::this_thread::sleep_for(std::chrono::milliseconds(500));
      if (InitExcept != std::exception_ptr())
      {
        IsRun = false;
        std::rethrow_exception(InitExcept);
      }
      Threads.push_back(std::move(Thread));
    }
    //std::cout << "Press Enter fot quit." << std::endl;
    std::cin.get();//пауза веб-сервера, чтоб не закрывался
    IsRun = false;
  }
  catch (std::exception const &e)
  {
    std::cerr << "Error: " << e.what() << std::endl;
  }
}
```