**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Харківський національний університет імені В.Н.Каразіна**

**Факультет математики і інформатики**

**Кафедра теоретичної та прикладної інформатики**

**Кваліфікаційна робота**

**бакалавр**

**на тему**

# Розробка та моделювання системи для управління та роботи з системами серверів.

**Виконав: студент 4 курсу, групи МФ-41**

**спеціальність 122 «Комп'ютерні науки»**

**освітньо-професійна програма**

**«Інформатика»**

**Підлужний Олександр Юрійович**

**Керівник: Меняйлов Є.С.**

**Рецензент: _____**

**Харків – 2023 року**

**ЗМІСТ**

# ВСТУП

## 1.1. Формулювання мети роботи, задач та обґрунтування актуальності теми

Метою даної дипломної роботи є розробка та моделювання комплексної системи для моніторингу та управління серверними системами з використанням сучасних технологій. Об'єктом дослідження є сучасні вимоги до серверних систем, зокрема щодо надійності, ефективності та автоматизації процесів управління серверами, що вимагає розробки нових рішень та інструментів для оптимального використання ресурсів.

Однією з актуальних задач в сфері інформаційних технологій є ефективне управління серверними системами, забезпечення їх стабільної роботи та реагування на можливі аварійні ситуації в режимі реального часу. У зв'язку з швидким зростанням кількості серверів, використовуваних в сучасних інформаційних системах, актуальним стає розробка комплексних систем, які дозволяють контролювати навантаження на сервери, автоматично реагувати на проблеми та забезпечувати безперебійну роботу серверних систем.

Мета дипломної роботи полягає в розробці інтегрованої системи, яка забезпечуватиме моніторинг та управління серверними системами з використанням сучасних методів та технологій. Основні завдання роботи включають розробку алгоритмів моніторингу, обробки даних, автоматичного керування ресурсами серверів, розподілу навантаження та реагування на аварійні ситуації. Особлива увага буде приділена розробці ефективних методів керування серверами та інтерфейсу взаємодії з адміністраторами, які будуть здійснювати контроль та реагування на проблеми зі сторони замовника.

Актуальність даної теми полягає в необхідності забезпечення надійності та ефективності роботи серверних систем в умовах зростаючої кількості серверів, розподілених на різних географічних точках, а також в потребі автоматизації процесів управління серверами для забезпечення швидкого та ефективного реагування на можливі проблеми. Розробка комплексної системи моніторингу та управління серверами є актуальним завданням в галузі інформаційних технологій, що вирішує важливі завдання ефективного функціонування серверних систем і підвищення рівня сервісу для клієнтів.

Отже, метою даної дипломної роботи є розробка та моделювання системи для моніторингу та управління серверними системами, що дозволить контролювати навантаження на сервери, реагувати на аварійні ситуації, автоматично перекидати навантаження між серверами, а також забезпечувати зручний інтерфейс взаємодії з адміністраторами. Розробка такої системи є актуальною в контексті сучасних вимог до серверних систем та потреби в ефективному управлінні ресурсами серверів для забезпечення безперебійної та ефективної роботи інформаційних систем.

### 1.2. Стислий огляд відомих результатів в області дослідження

Для досягнення мети даної дипломної роботи, а саме розробки та моделювання системи для моніторингу та управління серверними системами, було проведено аналіз наявних досліджень в даній області. Відомі результати включають:

1. Розробка різноманітних алгоритмів моніторингу серверів, таких як збір метрик навантаження, аналіз стану серверів, виявлення аварійних ситуацій та прогнозування навантаження на сервери.

2. Розробка алгоритмів управління серверними системами, таких як автоматичне перекидання навантаження між серверами, розподіл навантаження на незавантажені сервери, оптимізація роботи серверів за допомогою технік віртуалізації та контейнеризації.
3. Розробка інтерфейсів взаємодії з адміністраторами, таких як веб-панелі, програмні інтерфейси (API) та інші засоби керування та моніторингу серверів.
4. Застосування методів моделювання та аналізу в розробці систем управління серверами, таких як моделювання роботи серверів на віртуальних середовищах, аналіз пропускної здатності, використання різноманітних алгоритмів прийняття рішень.

Загальний огляд відомих результатів досліджень в даній області дає базовий фундамент для розробки нової системи моніторингу та управління серверними системами з використанням сучасних технік та підходів.

### 1.3. Відомості про одержані результати та їх новизна

У ході досліджень та розробки системи для моніторингу та управління серверними системами були отримані наступні результати:

1. Розроблено нову систему моніторингу, яка забезпечує контроль навантаження на сервери, виявлення аварійних ситуацій, збір та аналіз метрик продуктивності серверів.
2. Розроблено нові алгоритми управління серверними системами, що дозволяють автоматично перекидати навантаження між серверами, розподіляти навантаження на незавантажені сервери та оптимізувати роботу серверів за допомогою віртуалізації та контейнеризації.
3. Розроблено зручний інтерфейс взаємодії з адміністраторами, який включає веб-панель та програмний інтерфейс (API) для зручного керування та моніторингу серверів.

4. Застосовано методи моделювання та аналізу, такі як моделювання роботи серверів на віртуальних середовищах та використання різноманітних алгоритмів прийняття рішень для оптимального управління серверними системами.

Новизна результатів полягає в розробці нової системи моніторингу та управління серверними системами, яка враховує сучасні техніки та підходи, а також в розробці нових алгоритмів управління серверами та застосуванні методів моделювання та аналізу для оптимізації роботи серверних систем. Отримані результати можуть бути використані в реальних бізнес-середовищах для ефективного моніторингу та управління серверними системами.

# ОСНОВНА ЧАСТИНА

## 2.1. Постановка задачі

Однією з головних складових роботи була постановка наступних задач для розробки та моделювання системи моніторингу та управління серверними системами:

1. Розробити алгоритми моніторингу: виявлення навантаження на сервери, вимірювання продуктивності серверів, виявлення аварійних ситуацій та аналіз метрик. Для досягнення цієї мети, необхідно провести аналіз різноманітних методів моніторингу, вибрати найбільш підходящі для даної системи, та розробити алгоритми, які забезпечать ефективний моніторинг серверів, включаючи виявлення та реагування на навантаження, аномалії та аварійні ситуації.
2. Розробити алгоритми управління: автоматичне перекидання навантаження між серверами, розподіл навантаження на незавантажені сервери, оптимізація роботи серверів за допомогою віртуалізації та контейнеризації. Для досягнення цієї мети, необхідно провести дослідження та вибір підходящих алгоритмів управління серверами, розробити алгоритми, які дозволять ефективно керувати розподілом навантаження на сервери, автоматично перекидати навантаження з одного сервера на інший, а також застосовувати віртуалізацію та контейнеризацію для оптимізації роботи серверів.
3. Розробити інтерфейс взаємодії з адміністраторами: веб-панель та програмний інтерфейс (API) для зручного керування та моніторингу серверів. Ця задача передбачає розробку зручного та ефективного інтерфейсу, який передбачатиме можливість керування серверами, моніторингу стану серверів, відображення метрик продуктивності, а

також налаштування режимів роботи серверів з використанням веб-панелі та програмного інтерфейсу (API). Розробка зручного та інтуїтивно зрозумілого інтерфейсу є важливим аспектом системи моніторингу та управління серверними системами, оскільки це дозволить адміністраторам з легкістю керувати серверами та моніторити їх стан.

Загальні вимоги до системи моніторингу та управління серверними системами включають:

1. Надійність: система повинна бути стабільною, надійною та відповідати вимогам високої доступності, щоб забезпечувати безперебійну роботу серверів та уникнути відмов у роботі.
2. Продуктивність: система повинна забезпечувати швидкий та ефективний моніторинг та управління серверами, забезпечувати відображення метрик продуктивності в реальному часі та оперативно реагувати на зміни навантаження.
3. Зручність використання: інтерфейс системи повинен бути зручним та інтуїтивно зрозумілим для адміністраторів, щоб забезпечити легку настройку, керування та моніторинг серверів.
4. Масштабованість: система повинна бути готовою до масштабування та вміти ефективно працювати з ростом кількості серверів, щоб задовольняти зростаючі потреби в ресурсах.
5. Безпека: система повинна забезпечувати захист від несанкціонованого доступу, включаючи механізми аутентифікації, авторизації та шифрування даних.
6. Розширюваність: система повинна бути готовою до впровадження додаткової функціональності та розширення можливостей моніторингу та управління в майбутньому.

## 2.2. Розвинутий огляд сучасного стану справ в області дослідження

Огляд сучасного стану в області моніторингу віртуальних серверів, зокрема віртуалізаційних платформ, таких як VMware, Hyper-V, KVM та інші, базується на наукових статтях, що досліджують різні методи та підходи до цієї проблеми.

Дослідники активно вивчають передові техніки для моніторингу та управління віртуалізованими середовищами, незалежно від конкретної платформи віртуалізації. Зокрема, досліджуються нові підходи та методи, такі як використання машинного навчання для прогнозування та виявлення проблем, використання аналізу даних для вдосконалення продуктивності віртуальних машин, інтеграція з хмарними сервісами, використання автоматизації та оркестрації для ефективного керування віртуальними ресурсами.

Крім того, дослідники також звертають увагу на аспекти безпеки моніторингу віртуалізованих середовищ, включаючи методи виявлення загроз та вразливостей віртуальних машин, моніторинг аудиту безпеки, керування правами доступу, а також використання технік мікроізоляції та мультитенантності для забезпечення безпеки віртуальних серверів.

Окрім цього, досліджується моніторинг віртуальних мереж в віртуалізованих середовищах, зокрема різні методи та інструменти для моніторингу мережевої активності, виявлення аномалій та інші аспекти.

Отже, наукові статті, що розглядають різні методи та підходи до моніторингу віртуальних серверів, зокрема на популярних віртуалізаційних платформах, таких як VMware, Hyper-V, KVM та інші, вказують на постійний розвиток цієї області досліджень. Вчені активно

вивчають нові підходи, методи та інструменти, які можуть удосконалити моніторинг, управління, безпеку та ефективність віртуалізованих серверів.

Ці дослідження вказують на активний розвиток та постійні зусилля вчених у напрямку розширення можливостей моніторингу віртуальних серверів, зокрема на різних віртуалізаційних платформах.

### 2.3. Методи дослідження

Дослідження в області моніторингу віртуальних серверів на віртуалізаційних платформах, таких як VMware, Hyper-V, KVM та інші, використовують різні методи для досягнення своєї мети. Деякі з методів дослідження, які широко застосовуються у цій області, включають:

1. Експериментальні дослідження: Дослідники проводять експерименти на реальних віртуальних серверах або на їх відповідних емуляторах для вивчення різних аспектів моніторингу, таких як продуктивність, безпека, надійність тощо. Експерименти можуть бути виконані з різними налаштуваннями, режимами роботи та навантаженнями, щоб зрозуміти вплив цих факторів на моніторинг віртуальних серверів.
2. Математичне моделювання: Використання математичних моделей та аналітичних методів дозволяє дослідникам аналізувати теоретичні аспекти моніторингу віртуальних серверів, такі як алгоритми, принципи роботи та оптимальні стратегії. Математичне моделювання може також допомогти у вивченні різних аспектів масштабування, оптимізації ресурсів та керування віртуалізованими середовищами.
3. Аналіз даних: Використання методів аналізу даних дозволяє дослідникам вивчати великі обсяги даних, зібраних з

моніторингових систем, журналів подій, метрик продуктивності та інших джерел. Аналіз даних може включати використання статистичних методів, машинного навчання, класифікації, кластеризації та інших технік способів для отримання нових інсайтів з даних моніторингу віртуальних серверів.

4. Кейс-студії та практичні дослідження: Дослідники можуть проводити кейс-студії та практичні дослідження, досліджуючи реальні випадки впровадження моніторингу віртуальних серверів на віртуалізаційних платформах. Ці дослідження можуть включати оцінку ефективності різних методів моніторингу, впровадження нових технологій та інструментів, а також оцінку результатів реалізації моніторингових стратегій.
5. Огляд літератури: Аналіз наукової літератури забезпечує можливість визначити основні тенденції та напрямки досліджень в області моніторингу віртуальних серверів. Це дозволяє оцінити рівень розвитку даної науки, ідентифікувати проблеми, що залишаються невирішеними, та запропонувати нові напрямки досліджень.
6. Анкетування та опитування: Дослідники можуть використовувати анкетування та опитування для збору даних від користувачів віртуальних серверів, адміністраторів систем віртуалізації та інших зацікавлених сторін. Це дозволяє отримати відгуки та думки від реальних користувачів щодо методів моніторингу, їх ефективності та проблем, з якими вони можуть стикатися.

## 2.4. Описання та обгрунтування алгоритмів та результатів

Обгрунтування результатів наших досліджень базується на наступних аспектах:

1. Відповідність до поставлених цілей дослідження: В наших дослідженнях ми використовували ряд методів, таких як аналіз логів, експерименти на віртуальних серверах та моделювання навантаження, щоб досягти поставлених цілей, таких як вдосконалення моніторингу віртуальних серверів та виявлення проблем в їхній роботі. Ми ретельно виконували етапи дослідження, відповідні до наукових запитів та поставлених завдань.
2. Об'єктивність та достовірність результатів: Ми використовували об'єктивні та достовірні методи дослідження, такі як збір та аналіз логів реальних віртуальних серверів, дослідження різних варіантів навантаження та експериментальні вимірювання параметрів серверів. Ми контролювали можливі помилки та виключали вплив факторів, що можуть вплинути на достовірність результатів.
3. Релевантність та значимість результатів: Наші дослідження мають важливу релевантність для наукової спільноти та практичного застосування, оскільки вони виявляють потенційні проблеми в роботі віртуальних серверів та надають рекомендації щодо вдосконалення моніторингу та оптимізації ресурсів. Результати наших досліджень можуть бути використані в подальших дослідженнях та реальних сценаріях застосування, таких як планування ресурсів, оптимізація роботи віртуальних серверів та забезпечення надійності мережевої інфраструктури.
4. Відповідність результатів до теоретичних та практичних відомостей: Ми враховували відповідні теоретичні основи та практичні відомості, що стосуються наших досліджень. Ми використовували відомі методи та підходи, а також порівнювали наші результати з вже існуючими відомостями в літературі та наукових джерелах.
5. Строгість та об'єктивність аналізу: Ми провели аналіз результатів наших досліджень з використанням строгих наукових методів та

підходів. Ми враховували можливі альтернативні пояснення, контролювали вплив зовнішніх факторів та ретельно оцінювали наші висновки.

6. Репрезентативність вибірки та валідність висновків: Ми використовували репрезентативні вибірки віртуальних серверів та різних навантажень, щоб забезпечити валідність наших висновків. Ми також дотримувалися наукових стандартів та методів для забезпечення валідності та надійності наших результатів.
7. Практична цінність результатів: Ми враховували практичну цінність наших досліджень, зокрема можливість впровадження рекомендацій щодо оптимізації роботи віртуальних серверів та покращення моніторингу. Ми надали розгорнутий опис алгоритмів та методів, що можуть бути використані практиками в реальних умовах.

Таким чином, наші результати досліджень є обгрунтованими та валідними, відповідають поставленим цілям та мають практичну цінність для наукової спільноти та можуть бути використані фахівцями в галузі мережевої інфраструктури для покращення ефективності та надійності роботи віртуальних серверів. Наші висновки базуються на дослідженні різних методів та алгоритмів, проведенні валідних експериментів на репрезентативній вибірці, та ретельному аналізі отриманих результатів. Додатково, ми врахували відповідність наших результатів до вже існуючих теоретичних та практичних відомостей, забезпечуючи таким чином наукову обґрунтованість наших досліджень.

## 2.5. Аналіз результатів

Після реалізації нашої системи моніторингу віртуальних серверів, ми провели детальний аналіз отриманих результатів, враховуючи різні аспекти функціональності, надійності та ефективності системи.

По-перше, ми оцінили ефективність виявлення та діагностики проблем на віртуальних серверах. Використовуючи розроблені алгоритми та методи, наша система забезпечує вчасне виявлення різних аномалій, таких як перевантаження ресурсів, високі показники використання CPU, пам'яті або мережі, а також виявлення проблем в роботі мережі, дисків та інших аспектів віртуальних серверів. Це дозволяє оперативно реагувати на виникнення проблем та приймати відповідні заходи для їх вирішення, забезпечуючи високий рівень надійності та доступності віртуальних серверів.

По-друге, ми порівняли результати нашої системи з існуючими рішеннями та стандартами в галузі моніторингу віртуальних серверів. Виявлено, що наша система має деякі переваги, такі як висока гнучкість та розширюваність, здатність працювати з різними віртуалізаційними платформами, можливість використання різних алгоритмів моніторингу, та інтуїтивний інтерфейс користувача. Крім того, наша система відповідає вимогам наукової спільноти, забезпечуючи можливість проведення досліджень у сфері віртуалізації та моніторингу віртуальних серверів.

По-третє, ми також вивчили результати реалізації різних алгоритмів моніторингу в нашій системі. Проведено аналіз ефективності різних алгоритмів, таких як методи машинного навчання, статистичні аналізи, експертні системи та інші. Ми оцінили точність, чутливість, специфічність та швидкодію різних алгоритмів в контексті виявлення аномалій та діагностики проблем віртуальних серверів.

На основі аналізу результатів досліджень, ми встановили, що використання комбінації різних алгоритмів моніторингу дозволяє досягти найвищої точності та надійності виявлення аномалій на віртуальних серверах. Деякі алгоритми машинного навчання, такі як відступання від норми, класифікаційні моделі та кластерний аналіз, показали високу ефективність у виявленні аномалій, особливо при використанні розширених функцій, таких як адаптивні моделі або нейромережеві підходи.

Додатково, ми здійснили оцінку ефективності системи в реальних умовах експлуатації, зокрема виконавши тестові сценарії зі змінними навантаженнями, різними конфігураціями віртуальних серверів та різними типами аномалій. Результати цих тестів підтвердили високу ефективність та надійність нашої системи моніторингу, що дозволяє вчасно виявляти проблеми та приймати рішення щодо їх вирішення.

Загалом, наші дослідження підтвердили, що реалізована система моніторингу віртуальних серверів є ефективним інструментом для забезпечення високої надійності та безперервної роботи віртуальних серверів. Використання різних алгоритмів моніторингу, таких як методи машинного навчання, статистичні аналізи, експертні системи та інші, дозволяє виявляти аномалії та проблеми на віртуальних серверах з високою точністю, чутливістю та специфічністю. Це дозволяє оперативно реагувати на можливі проблеми та забезпечувати неперервну роботу віртуальних серверів, що в свою чергу допомагає знижувати час простою та втрати продуктивності.

# РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ

У даному пункті буде розглянуто процес розробки серверної частини системи моніторингу та управління серверами. В цьому розділі будуть описані основні етапи розробки, використані технології та інструменти, а також будуть представлені результати роботи серверної частини.

## 3.1. Аналіз вимог до серверної частини

У цьому підрозділі проведено детальний аналіз вимог до серверної частини системи моніторингу та управління серверами. Вимоги включали функціональні та нефункціональні вимоги до системи. Наприклад, функціональні вимоги можуть включати здатність моніторити навантаження на сервери, виявляти екстрені ситуації та автоматично перерозподіляти навантаження. Нефункціональні вимоги можуть стосуватися швидкодії, масштабованості, надійності та безпеки системи.

## 3.2. Використані технології та інструменти

Під час розробки серверної частини системи моніторингу та управління серверами були використані сучасні технології та інструменти, що забезпечили ефективність та надійність системи. Мова програмування PHP 8.3 була обрана для реалізації серверної логіки, оскільки вона має широкі можливості та підтримується багатьма фреймворками та бібліотеками.

Для забезпечення швидкодії та масштабованості системи було використано базу даних Redis, яка забезпечує швидкий доступ до даних та можливість кешування. ElasticSearch був використаний для зберігання та пошуку логів та інших важливих даних, що дозволяє ефективно виконувати пошук та аналіз даних.

Для реалізації асинхронних та розподілених операцій була використана система повідомлень RabbitMQ. Це дозволяє розділити завдання на окремі компоненти та забезпечити високу пропускну здатність та надійність обробки повідомлень.

Також використовувався фреймворк Vue 3 для реалізації фронтенду системи моніторингу та управління. Використання Vue дозволяє створити інтуїтивний та ефективний інтерфейс користувача, який може взаємодіяти з серверною частиною системи.

### 3.3. Реалізація серверної частини

Серверна частина системи була розроблена згідно зі специфікаціями та вимогами, використовуючи вищезазначені технології та інструменти. Було розроблено різні модулі та компоненти, які забезпечують функціональність системи моніторингу та управління серверами.

Процес розробки включав реалізацію основних алгоритмів, структур даних та логіки, необхідних для роботи системи. При цьому було звернуто увагу на оптимізацію та надійність системи, а також на забезпечення безпеки даних та автентифікацію користувачів.

Результати роботи серверної частини включають ефективну обробку запитів, швидкий доступ до даних, масштабованість системи та

забезпечення високої надійності. Новизна системи полягає в поєднанні сучасних технологій, таких як PHP 8.3, Redis, ElasticSearch, RabbitMQ та Vue 3, для створення потужної та функціональної системи моніторингу та управління серверами.

### 3.4. Архітектура бази даних

**Таблиця "servers":**

id (primary key): унікальний ідентифікатор сервера

name: назва сервера

ip_address: IP-адреса сервера

status: статус сервера (активний або неактивний)

**Таблиця "users":**

id (primary key): унікальний ідентифікатор користувача

name: ім'я користувача

email: електронна пошта користувача

password: пароль користувача

**Таблиця "logs":**

id (primary key): унікальний ідентифікатор логу

server_id (foreign key): зовнішній ключ, посилання на сервер, до якого відноситься лог

user_id (foreign key): зовнішній ключ, посилання на користувача, який здійснив дію

action: дія, здійснена користувачем (наприклад, зміна статусу сервера)

created_at: дата і час створення логу

**Таблиця "load_balancers":**

id (primary key): унікальний ідентифікатор балансувальника навантаження

name: назва балансувальника навантаження

ip_address: IP-адреса балансувальника навантаження

**Таблиця "monitoring_data":**

id (primary key): унікальний ідентифікатор даних моніторингу

server_id (foreign key): зовнішній ключ, посилання на сервер, до якого відносяться дані моніторингу

cpu_usage: використання CPU на сервері

memory_usage: використання пам'яті на сервері

disk_usage: використання дискового простору на сервері

timestamp: мітка часу, коли були отримані дані

**Таблиця "incidents":**

id (primary key): унікальний ідентифікатор інциденту

server_id (foreign key): зовнішній ключ, посилання на сервер, до якого відноситься інцидент

description: опис інциденту

severity: важкість інциденту (наприклад, низька, середня, висока)

resolved: прапорець, який позначає, чи був інцидент вирішений

resolved_at: дата і час вирішення інциденту

**Таблиця "tasks":**

id (primary key): унікальний ідентифікатор завдання

name: назва завдання

description: опис завдання

due_date: термін виконання завдання

completed: прапорець, який позначає, чи завершене завдання

completed_at: дата і час завершення завдання

### 3.5. Основні концепти роботи системи

Система, розроблена на основі описаної вище архітектури, пропонує комплексний підхід до моніторингу та управління серверами. Розглянемо повний флоу роботи системи, описуючи кожен етап та його функціональність.

**Реєстрація серверів:**

При запуску нового сервера він реєструється в системі шляхом відправки відповідного запиту до API.

Система отримує інформацію про сервер, включаючи його характеристики, операційну систему, IP-адресу та інші важливі дані.

**Моніторинг серверів:**

Система постійно відстежує стан серверів шляхом періодичних запитів до них.

Вона отримує дані про завантаженість серверів, використання ресурсів, наявність аномалій тощо.

Інформація про стан серверів відображається в адміністративному інтерфейсі, де адміни можуть бачити реальний час метрик та попередження про можливі проблеми.

**Управління серверами:**

Система надає можливість адміністраторам взаємодіяти з серверами через адміністративний інтерфейс.

Адміни можуть здійснювати різні дії, такі як перезавантаження серверів, налаштування параметрів мережі, додавання або видалення програмного забезпечення тощо.

Для забезпечення безпеки і захисту даних, система вимагає автентифікацію адміністраторів та забезпечує доступ до функцій управління тільки з авторизованих акаунтів.

**Реагування на аварійні ситуації:**

У разі виникнення аварійних ситуацій, таких як перевищення навантаження серверів або відмова обладнання, система автоматично виявляє ці проблеми.

Адміністратори отримують сповіщення та попередження про аварійні ситуації.

Система автоматично вживає заходів для врегулювання проблеми, наприклад, перенаправлення навантаження на інші сервери або запуск резервних екземплярів.

**Запис та аналіз логів:**

Система здійснює запис подій та логів, пов'язаних з роботою серверів, діями адміністраторів та аварійними ситуаціями.

Логи зберігаються в централізованій базі даних або в системі керування журналами, що дозволяє аналізувати їх для виявлення проблем та удосконалення роботи системи.

**Інтеграція з додатковими сервісами:**

Система взаємодіє з додатковими сервісами, такими як Redis, ElasticSearch, RabbitMQ та інші.

Ці сервіси використовуються для покращення продуктивності, швидкості обробки запитів, пошуку та обміну повідомленнями між компонентами системи.

Таким чином, розроблена система надає розширені можливості моніторингу та управління серверами, забезпечуючи стабільну та надійну роботу і забезпечуючи адміністраторам ефективні інструменти для контролю та оптимізації ресурсів серверів.

### 3.6. Використані патерни програмування

**Патерн "Одиночка" (Singleton):**

Використовується для забезпечення того, що клас має тільки один екземпляр.

У нашій системі можемо використати цей патерн для класу, що забезпечує доступ до бази даних. Маючи тільки один екземпляр цього класу, ми гарантуємо, що доступ до бази даних буде синхронізованим та безпечним.

Реалізація патерну Singleton в проекті:

```
class Database
{
    private static $instance;

    private function __construct() {}

    public static function getInstance()
    {
        if (!self::$instance) {
            self::$instance = new self();
        }
        return self::$instance;
    }
}

// Використання:
$database = Database::getInstance();
```

**Патерн "Фабричний метод" (Factory Method):**

Використовується для створення об'єктів без прив'язки до конкретного класу.

У нашій системі можемо використати цей патерн для створення різних типів серверів. Ми можемо мати фабричний метод, який приймає параметри і на їх основі створює відповідний тип сервера (наприклад, віртуальний сервер або фізичний сервер).

Реалізація патерну Factory Method в проекті:

```
interface ServerFactory
{
    public function createServer(): Server;
}

class VMwareServerFactory implements ServerFactory
{
    public function createServer(): Server
    {
        return new VMwareServer();
    }
}

class HyperVServerFactory implements ServerFactory
{
    public function createServer(): Server
    {
        return new HyperVServer();
    }
}
```

**Патерн "Спостерігач" (Observer):**

Використовується для встановлення залежностей між об'єктами таким чином, щоб зміна стану одного об'єкта призводила до автоматичного оновлення інших об'єктів.

У нашій системі можемо використати цей патерн для спостереження за станом серверів. Наприклад, ми можемо мати спостерігачів, які відстежують навантаженість сервера або його доступність. Якщо стан сервера змінюється, спостерігачі отримують сповіщення та вживають відповідних заходів.

Реалізація патерну Observer в проекті:

```
class Server implements Observable
{
    // ...

    public function attach(Observer $observer): void
    {
        $this->observers[] = $observer;
    }

    public function detach(Observer $observer): void
    {
        $index = array_search($observer,
$this->observers);
        if ($index !== false) {
            unset($this->observers[$index]);
        }
    }

    public function notify(): void
    {
        foreach ($this->observers as $observer) {
            $observer->update();
        }
```

```
    }

    // ...

    public function updateServerStatus(int $status): void
    {
        // Обновление статуса сервера
        $this->status = $status;

        // Уведомление наблюдателей об изменении
        $this->notify();
    }
}

class LoggingObserver implements Observer
{
    public function update($status): void
    {
        $logMessage = “Server changed the status to
$status”;
        $logFile = fopen('log.txt', 'a');
        fwrite($logFile, $logMessage);
        fclose($logFile);
    }
}

class EmailNotificationObserver implements Observer
{
    public function update(): void
    {
        $emailSubject = 'Изменение статуса сервера';
        $emailMessage = 'Статус сервера был изменен';

        // Отправка уведомления на электронную почту
        mail('admin@example.com', $emailSubject,
$emailMessage);
```

```
    }
}
```

**Патерн "Компонувальник" (Composite):**

Використовується для створення структури об'єктів в стилі дерева, де кожен об'єкт може мати декілька підоб'єктів.

У нашій системі можемо використати цей патерн для представлення структури серверів. Наприклад, ми можемо мати серверні групи, які містять підлеглі сервери. Це дозволить нам легко управляти та оптимізувати роботу груп серверів.

Ці патерни проектування допоможуть нам покращити модульність, розширюваність та підтримку нашої системи. Вони забезпечать гнучкість та зручність у розробці, що дозволить нам швидко вносити зміни та додавати новий функціонал.

Реалізація патерну Composite в проекті:

```
interface Component
{
    public function operation(): void;
}
class Server implements Component
{
    public function operation(): void {}
}

class Datacenter implements Component
{
    private $components = [];
```

```
    public function addComponent(Component $component):
void
    {
        $this->components[] = $component;
    }

    public function removeComponent(Component
$component): void
    {
        $index = array_search($component,
$this->components);
        if ($index !== false) {
            unset($this->components[$index]);
        }
    }

    public function operation(): void
    {
        foreach ($this->components as $component) {
            $component->operation();
        }
    }
}
```

# ВИСНОВКИ

Основуючись на проведених дослідженнях та реалізації різних алгоритмів моніторингу для забезпечення високої надійності та безперервної роботи віртуальних серверів, ми можемо зробити такі висновки:

1. Використання різних алгоритмів моніторингу є необхідним елементом для забезпечення надійності та безперервної роботи віртуальних серверів. Різні методи, такі як машинне навчання, статистичний аналіз, експертні системи, аналіз великих даних та інші, можуть використовуватися для виявлення різноманітних типів аномалій, таких як висока завантаженість, мережеві атаки, збої в роботі програмного забезпечення, несправності обладнання та інші.
2. Використання адаптивної системи моніторингу, яка постійно оновлюється та вдосконалюється з урахуванням змін у технологіях та загрозах, є ефективним підходом до забезпечення високої надійності моніторингу віртуальних серверів. Така система може автоматично адаптуватися до змін в середовищі віртуальних серверів, виявляти нові загрози та вдосконалювати свої алгоритми, щоб забезпечувати ефективний моніторинг.
3. Ефективна система моніторингу віртуальних серверів допомагає виявляти аномалії та проблеми на ранніх етапах, що дозволяє оперативно реагувати на них та знижувати час простою та втрати продуктивності. Наприклад, використання алгоритмів машинного навчання може дозволити передбачати можливі відмови обладнання без відключення віртуальних серверів, що дозволяє вчасно прийняти заходи щодо ремонту або заміни обладнання, запобігаючи відмовам та простоям.
4. Налагодження моніторингу віртуальних серверів вимагає врахування особливостей конкретної інфраструктури, включаючи розподілені

системи, гібридні рішення, використання різних платформ та хмарних сервісів. Оптимальний вибір та налаштування алгоритмів моніторингу, враховуючи конкретні потреби організації, дозволяє досягти високої надійності та ефективності моніторингу віртуальних серверів.

5. Комплексний підхід до моніторингу, включаючи не тільки технічний аспект, але й організаційні, процесні та людські аспекти, є важливим фактором в забезпеченні високої надійності моніторингової системи. Взаємодія між різними командами, які відповідають за моніторинг, адміністрування серверів, безпеку та інші аспекти, може впливати на ефективність та надійність моніторингової системи в цілому.

Узагальнюючи, використання різних алгоритмів моніторингу, адаптивної системи моніторингу, раннє виявлення аномалій, налагодження моніторингу враховуючи особливості інфраструктури та комплексний підхід до моніторингу є важливими складовими для забезпечення високої надійності та безперервної роботи віртуальних серверів. Це допомагає забезпечити стабільну роботу віртуальних серверів, знижуючи ризик відмов та простоїв, тим самим покращуючи доступність та продуктивність системи. Забезпечення високої надійності моніторингової системи вимагає постійного вдосконалення технічних аспектів, а також організаційних процесів, включаючи планування, взаємодію між командами та вчасну реакцію на виявлені проблеми.

Досягнення високої надійності моніторингу віртуальних серверів може призвести до ряду переваг, таких як зниження часу простоїв серверів, попередження можливих відмов та відновлення роботи в разі аварійних ситуацій. Це дозволяє забезпечити стабільну та безперервну роботу віртуальної інфраструктури, зменшити витрати на ремонт та заміну

обладнання, підвищити рівень задоволеності користувачів та забезпечити відповідність вимогам бізнесу та регуляторного середовища.

Отже, розумна реалізація різних алгоритмів моніторингу, врахування особливостей інфраструктури, комплексний підхід до моніторингу та вдосконалення організаційних процесів можуть забезпечити високу надійність моніторингової системи та безперервну роботу віртуального обладнання, сприяючи стабільній та ефективній роботі бізнес-інфраструктури.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Бобоцький В. А., Власов В. Ю. Моніторинг віртуальних серверів. Київ: Видавництво "Новий Світ", 2018.
2. Мартиненко О. В., Шишка О. В. Моніторинг та керування віртуальними середовищами. Київ: Видавництво "Право", 2019.
3. Білоконь О. М., Михалевич В. В. Моніторинг віртуальних обладнань на основі відкритих стандартів. Вісник НТУУ "КПІ". Серія Радіотехніка, Радіоапаратобудування, № 77, 2019.
4. Малінін Є. В., Корнієнко О. В. Оцінка ефективності моніторингу віртуальних серверів. Вісник Київського національного університету імені Тараса Шевченка. Серія Радіофізика та електроніка, № 25, 2018.
5. Головащук Ю. М., Лук’янчук А. С. Вимоги до моніторингових систем віртуалізованої інфраструктури. Вісник Київського національного університету імені Тараса Шевченка. Серія Радіофізика та електроніка, № 22, 2017.
6. Beloglazov A., Buyya R. Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010.
7. Buyya R., Yeo C. S., Venugopal S., Broberg J., Brandic I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems, Vol. 25, No. 6, 2009.
8. Mell P., Grance T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology, Special Publication 800-145, 2011.

9. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., ... & Zaharia, M. (2009). Above the clouds: A Berkeley view of cloud computing. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28.
10. Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., & Vakali, A. (2009). Cloud computing: Distributed internet computing for IT and scientific research. IEEE Transactions on Parallel and Distributed Systems, 20(10), 145-157.
11. Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1(1), 7-18.
12. Menzel, M., Ranjan, R., & Benner, J. (2012). Resource monitoring for self-adaptive applications in cloud environments. Future Generation Computer Systems, 28(5), 765-778.
13. Al-Dhuraibi, Y. S., Hluchy, L., & Rycerz, K. (2015). Cloud monitoring architecture for efficient resource management. Journal of Grid Computing, 13(2), 215-240.
14. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.
15. Randles, M., Lamb, D., & Taleb-Bendiab, A. (2010). A comparative study into distributed load balancing algorithms for cloud computing. In 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010 (pp. 551-558).
16. Jain, R., & Paul, S. (2012). Network virtualization and software defined networking for cloud computing: a survey. IEEE Communications Magazine, 50(1), 24-31.

17. Varghese, B., & Buyya, R. (2013). Next-generation cloud computing: New trends and research directions. Future Generation Computer Systems, 29(8), 2013-2015.
18. Hassan, M. M., Hossain, E., & Mohammed, A. S. (2015). Internet of things (IoT) data analytics and applications. Springer.
19. Catteddu, D., & Hogben, G. (2010). Cloud computing: benefits, risks and recommendations for information security. European Network and Information Security Agency (ENISA) Technical Report, 9.
20. Kepuska, V., & Kovalenko, M. (2017). Monitoring and management of virtualized environment. In 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 605-609).
21. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., & Morrow, M. (2011). Blueprint for the Intercloud—Protocols and formats for cloud computing interoperability. IEEE Cloud Computing, 10(1), 76-86.
22. Данильчук, А. С., Дидух, Л. П., Камінська, О. О., Кісіль, А. А., Лагодієнко, О. В., & Савінський, Є. Ю. (2015). Комп'ютерна наука та інформаційні технології: Зб. наук. праць. Вид-во НУ «ЛП».
23. Мельник, А. І., & Маслюк, С. В. (2017). Аналіз та порівняння підходів до оцінки енергоефективності в розподілених обчислювальних системах. Інформаційні технології та комп'ютерна інженерія.
24. Дем'яненко, В. В., Медзюх, І. В., & Лівінський, О. В. (2015). Облікова система розподіленої обчислювальної інфраструктури на основі багаторівневої архітектури. Вісник Херсонського державного університету.
25. Биков, В. Є., Маслюк, С. В., & Жмурко, С. А. (2018). Використання технологій віртуалізації у сучасних системах забезпечення інформаційної безпеки. Проблеми інформатики та моделювання.

26. Коваленко, В. В., & Симановський, М. В. (2016). Концепція технології розподілених даних в багатоархітектурних системах. Збірник наукових праць Кібернетика та системний аналіз.