Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Математичні моделі та методи
оцінки ресурсів для хмарних обчислювальних систем
(тема)

Виконав:
студент _2_ курсу, групи _ПМм–19–1
Танянський О. С.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо–професійна
(освітньо–професійна або освітньо–наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Єсілевський В.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Тевяшев А.Д.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо–професійна або освітньо–наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“____” _____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Танянському Олексію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Математичні моделі та методи оцінки ресурсів для хмарних обчислювальних систем

затверджена наказом по університету від 23 жовтня 2020 р. № 1422 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2020 р.

3. Вихідні дані до роботи Ланцюг Маркова з дискретним часом, алгоритм MapReduce, алгоритм RoundRobin

4. Перелік питань, що потрібно опрацювати в роботі
1. Системний аналіз проблеми хмарних обчислювань
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз проблеми
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	вересень 2020 р.	виконано
2	Вибір та обґрунтування методу	жовтень – листопад 2020 р.	виконано
3	Розробка алгоритму і програми	листопад – грудень 2020 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	листопад – грудень 2020 р.	виконано
5	Робота над текстом пояснювальної записки	грудень 2020 р.	виконано
6	Представлення роботи на рецензію в ЕК	грудень 2020 р.	виконано

Дата видачі завдання _1_ вересня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Єсілевський В.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 76 с., 12 табл., 24 рис., 35 джерел.

SOA СИСТЕМА, СЕРВІС, ПРИВАТНА ХМАРА, ЛАНЦЮГ МАРКОВА, ЙМОВІРНІСТЬ СТАНУ СИСТЕМИ, ПРОГНОЗУВАННЯ РЕСУРСІВ.

Об’єкт дослідження – застосування моделей та методів оцінки ресурсів для хмарних обчислювальних систем.

Мета роботи – розробка мобільної системи моніторингу ресурсів хмарних обчислювальних систем.

Методи дослідження – використання статистичних методів і алгоритму MapReduce.

Робота присвячена розробці алгоритмів для мобільної системи моніторингу та управління ресурсами хмарних та SOA систем. Була розроблена математична модель системи на основі ланцюга Маркова, що дозволяє робити прогнозування обчислювальних ресурсів для віртуальних машин. Були розраховані та проаналізовані характеристики цієї моделі, а також проведена програмна реалізація.

Система призначена для перегляду стану приватної хмари в режимі онлайн з пристроїв, що використовують платформу Android, а також прогнозування обчислювальних ресурсів приватної хмари. У систему входять: сервіс, що надає інтерфейси через які можна взаємодіяти з приватною хмарою, клієнтський додаток на базі ОС Android, який надає користувачеві інформацію щодо використання ресурсів хмари, а також налаштування, для регулювання стану хмари. Також, до системи входить агент для збору інформації щодо використання ресурсів віртуальної машини та інформативний вебсайт з можливістю регулювання роботи сервіса.

ABSTRACT

Introductory note: 76 pages, 12 tables, 24 figures, 35 sources.

SOA SYSTEM, SERVICE, PRIVATE CLOUD, MARKOV CHAIN, SYSTEM STATE POSSIBILITY, RESOURCE PREDICTION.

Object of research – application of models and methods of resource estimation for cloud computing systems.

Purpose of work – development of a mobile system for monitoring the resources of cloud computing systems.

Methods of research – use of statistical methods and MapReduce algorithm.

The goal of the work is to develop algorithms for mobile monitoring and resource management of cloud and SOA systems. The mathematical model based on a Markov chain which allows to predict computational resources of virtual machines has been developed. The characteristics of this model have been calculated and analyzed, and also the software has been developed.

The results of using these algorithms have been analyzed. Based on these results it is possible to make a conclusion about effectiveness of the developed algorithms.

The system is designed for viewing the status of private cloud in online mode using devices on Android platform (phone, tablet) and also predicting computational resources of the private cloud. The system includes: service that provides interfaces through which you can interact with a private cloud, the client application based on the Android OS, which provides the information about the cloud for user, and also the settings for controlling the state of the cloud. Also, the system includes the agent application for gathering information about virtual machine resource usage, and the informational website with options for adjusting service work process.

ЗМІСТ

ВСТУП

На сьогодняшній день моніторинг та прогнозування споживання ресурсів — це фундаментальна потреба для віртуалізованої системи, тому що віртуалізація стає дуже популярною у дата–центрах та великих обчислювальних інфраструктурах. Вона надає динамічне виділення та керування ресурсами, незважаючи на суворий поділ фізичних ресурсів системи. Однак запуск віртуального дата-центру з високою продуктивністю та ефективністю – це складна задача. Усередині фізичної машини монітор віртуальної машини повинен контролювати виділення фізичних ресурсів для кожної віртуальної машини.

Прогнозування ресурсів необхідно для такої системи, тому що хмарна інфраструктура використовує віртуальні ресурси за вимогою. Однак сьогоднішні моніторингові системи не достатні для прогнозування використання ресурсів віртуалізованої системи. Без належного моніторингу віртуалізовані системи можуть працювати повільніше, що безпосередньо впливає на хмарну інфраструктуру.

У даній роботі розглядається новий підхід до моделювання прогнозування ресурсів. Модель ґрунтується на обробці історії використання системи та прогнозуванні короткострокового використання ресурсів. У роботі детально розглянута модель прогнозування та моніторингу ресурсів на основі ланцюга Маркова. Під прогнозуванням ресурсів розуміється припущення щодо використання віртуальних ресурсів системи у майбутньому.

Також у роботі наведено приклад роботи системи із тестовими даними та статистичний аналіз цих даних, який дозволяє обчислити ймовірнісні характеристики процесу тестування. Після проведення статистичного аналізу виконується прогнозування обчислювальних ресурсів.

Розроблені у даній роботі методи та алгоритми дозволяють максимізувати ефективність мобільної системи моніторингу. Алгоритм MapReduce дозволяє структурувати велику кількість даних перед інтелектуальною обробкою. Також

система використовує алгоритм балансування навантаження, що дозволяє розподілити навантаження на сервера оптимальним чином.

Дана система не залежить від використовуваної платформи, має велику гнучкість і стійкість під час роботи. Може використовуватися у хмарних сервісах Amazon, Azure та Oracle.

1 СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМИ ЗАСТОСУВАННЯ МОДЕЛЕЙ ТА МЕТОДІВ ОЦІНКИ РЕСУРСІВ ДЛЯ ОПТИМІЗАЦІЇ ХМАРНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

1.1 Системний аналіз проблеми застосування моделей та методів оцінки ресурсів для оптимізації хмарних обчислювальних систем

1.1.1 Вербальна модель системи

Об’єкт дослідження – «застосування моделей та методів оцінки ресурсів для оптимізації хмарних обчислювальних систем».

Предмет дослідження – хмарні обчислювальні системи.

Досліджувана тема заснована на аналізі роботи хмарних систем.

Проведено дослідження моделей та методів оцінки ресурсів для оптимізації хмарних обчислювальних систем. Система «хмарні обчислювальні системи» відноситься до цілеспрямованих систем, для яких призначення визначається їх здатністю сприймати потреби та виконувати певні дії для задоволення цих потреб.

Головний вихід системи – модель установки. На входи системи подаються дані про стан хмарної системи, технічні характеристики та данні по хмарній системі, які однаково важні та являються невід’ємною частиною побудови моделі. За допомогою програмного продукту jetBrains IDE та мов програмування Java, JavaScript, C і безпосередньо за участю дослідника, здійснюється реалізація процесу оптимізації хмарних систем у мобільному додатку.

1.1.2 Морфологічний опис програми

Досліджувана частина – застосування теорії Марківських ланцюгів для прогнозування та моніторингу ресурсів хмарної системи.

Призначення моделі системи є надання математичної моделі для подальшого її використання при оптимізації в хмарній системі. Для реалізації мети системи використовуються різні ресурси. До таких ресурсів відносяться: інформаційні та технологічні.

Зовнішнє середовище – сукупність всіх об’єктів поза межі системи, через трансформаційних змін властивостей які впливають на систему, а так само тих об’єктів, властивості яких змінюються в результаті різної поведінки системи.

Модель зовнішнього середовища системи представлена на рисунку 1.1.

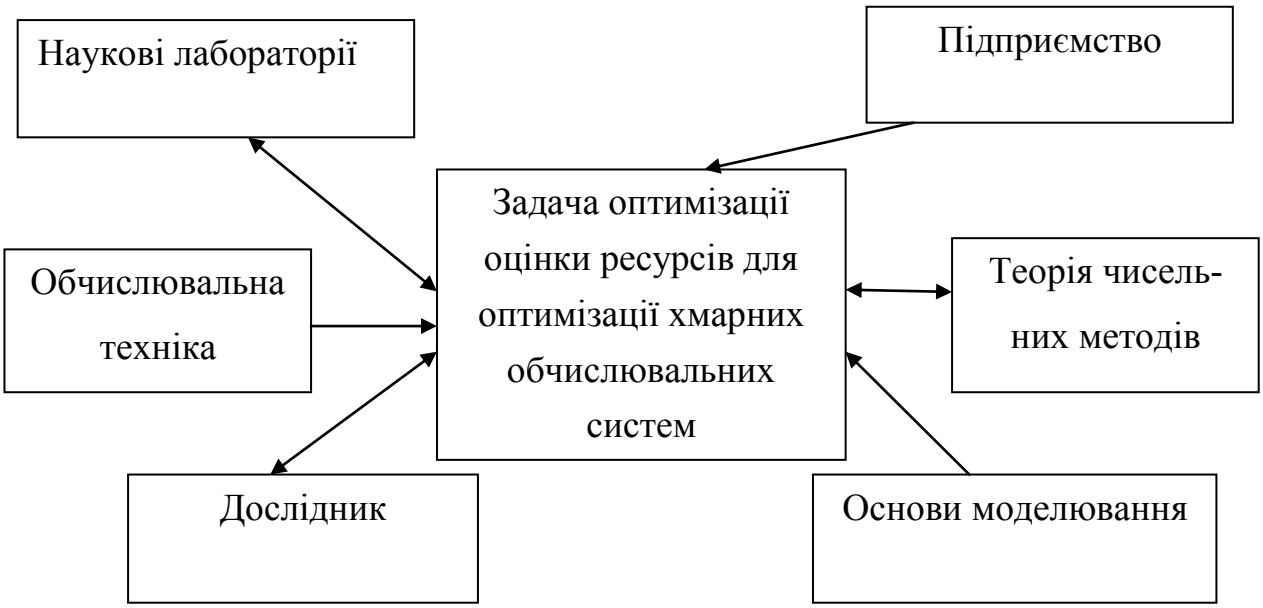


Рисунок 1.1 – Модель зовнішнього середовища системи

Об’єктами зовнішнього середовища є:

а) наукові лабораторії забезпечують умови для проведення дослідження;

б) обчислювальна техніка є засобом моделювання процесів управління хмарною системою;

в) дослідник безпосередньо проводить експерименти, вибирає методи процесу управління хмарною системою;

г) на підставі оптимізації будується спосіб рішення задачі управлінням хмарною системою;

д) промислові підприємства використовують результати дослідження,

впроваджуючи нові технології у виробництво;

є) теорія чисельних методів використовується для оптимізації управлінням хмарною системою.

Модель типу «чорний ящик» системи «оптимізація роботи хмарної системи» представлена на рисунку 1.2. Така модель є вихідною при побудові моделі складної системи, вона акцентує увагу дослідника на взаємодії системи із зовнішнім середовищем. Тут виходами системи є її цільові продукти. А входи – це взаємодія середовища на систему. Зміст «чорного ящика» не розкривається, тому що увага приділяється лише межі системи. Межа, в свою чергу, підкреслює цілісність системи, її відокремленість від зовнішнього середовища та взаємодію системи та середовища.

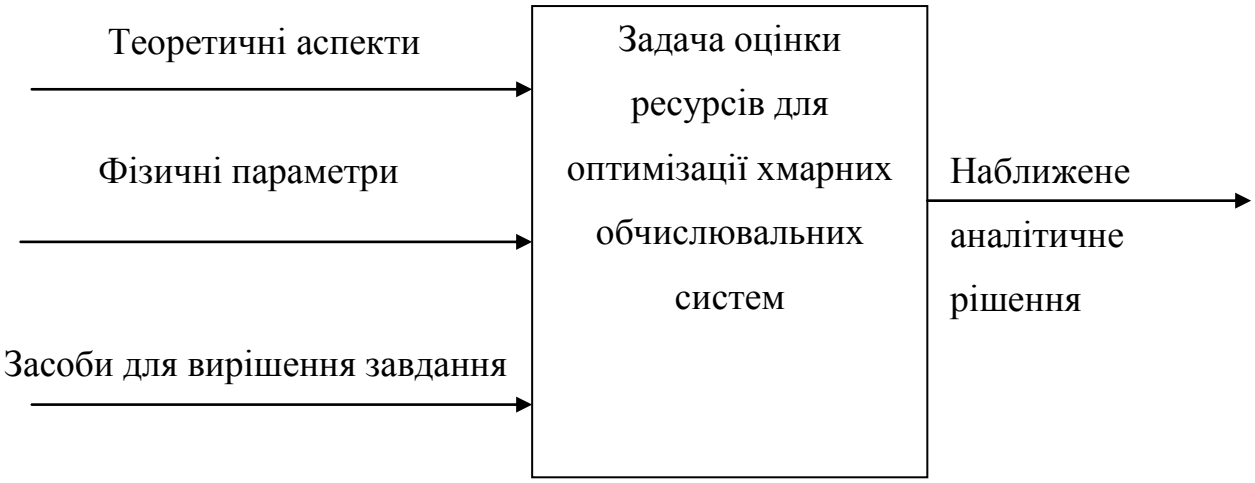


Рисунок 1.2 – Модель типа «чорний ящик»

У моделі типу «білий ящик» на рисунку 1.3 описаний склад системи та показані взаємозв’язки між елементами системи.

Дана система А складається з наступних підсистем: А1 и А2:

а) А1 – апарат управління, що складається з:

1) А11 дослідника;

2) А12 засобів управління: пульт управління та дисплей.

б) А2 – теоретична і обчислювальна база, що складається з:

1) А21 засобів виконання завдання: структури рішення і розрахункових формул;

2) А22 обслуговування: ЕОМ.

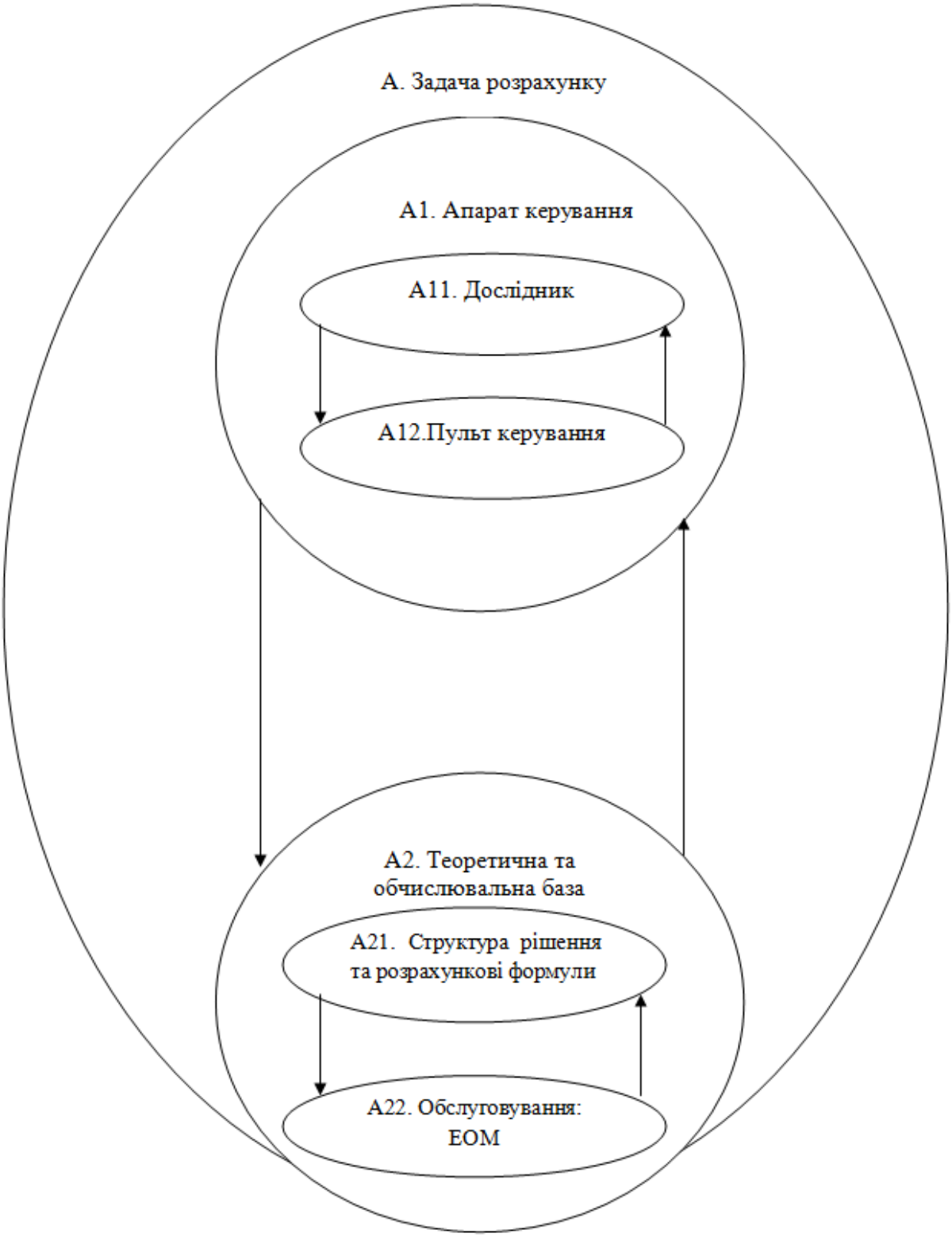


Рисунок 1.3 – Модель типу «білий ящик»

1.1.3 Функціональна модель системи

Перша діаграма в ієрархії діаграм – IDEF0 представлена на рисунку 1.4. Вона зображує функціонування системи в цілому. Така діаграма називається контекстною.

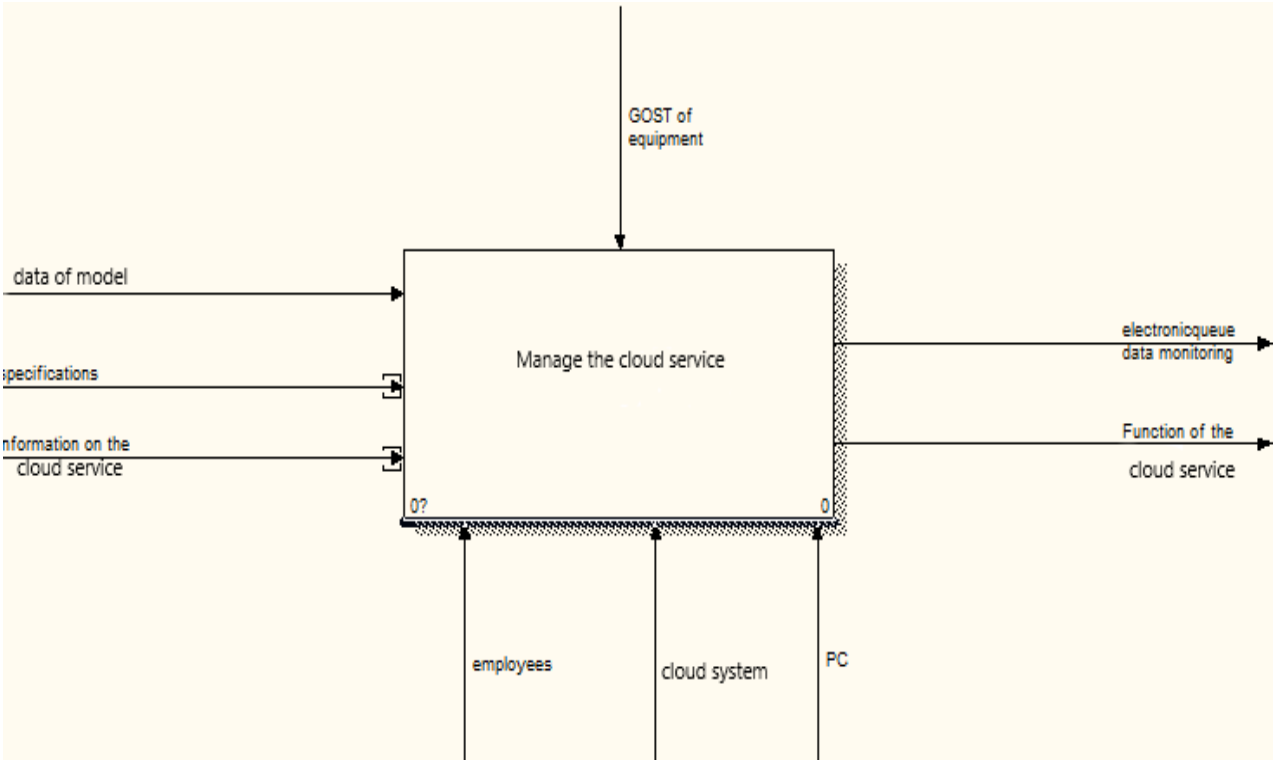


Рисунок 1.4 – Контекстна діаграма

В методології IDEF0 процес представляється у вигляді набору елементів, які взаємодіють між собою, а також можна визначити ресурси, що споживані кожною роботою.

Після того, як контекст описаний, будуються наступні діаграми в ієрархії. Кожна наступна діаграма є більш докладним описом однієї з робіт на діаграмі, що розташована вище.

У нашому випадку розглядається функціонування системи «оптимізація роботи хмарної системи». Друга діаграма демонструє основні функції системи та їх деталізації за рівнями, рисунки 1.5 – 1.7.

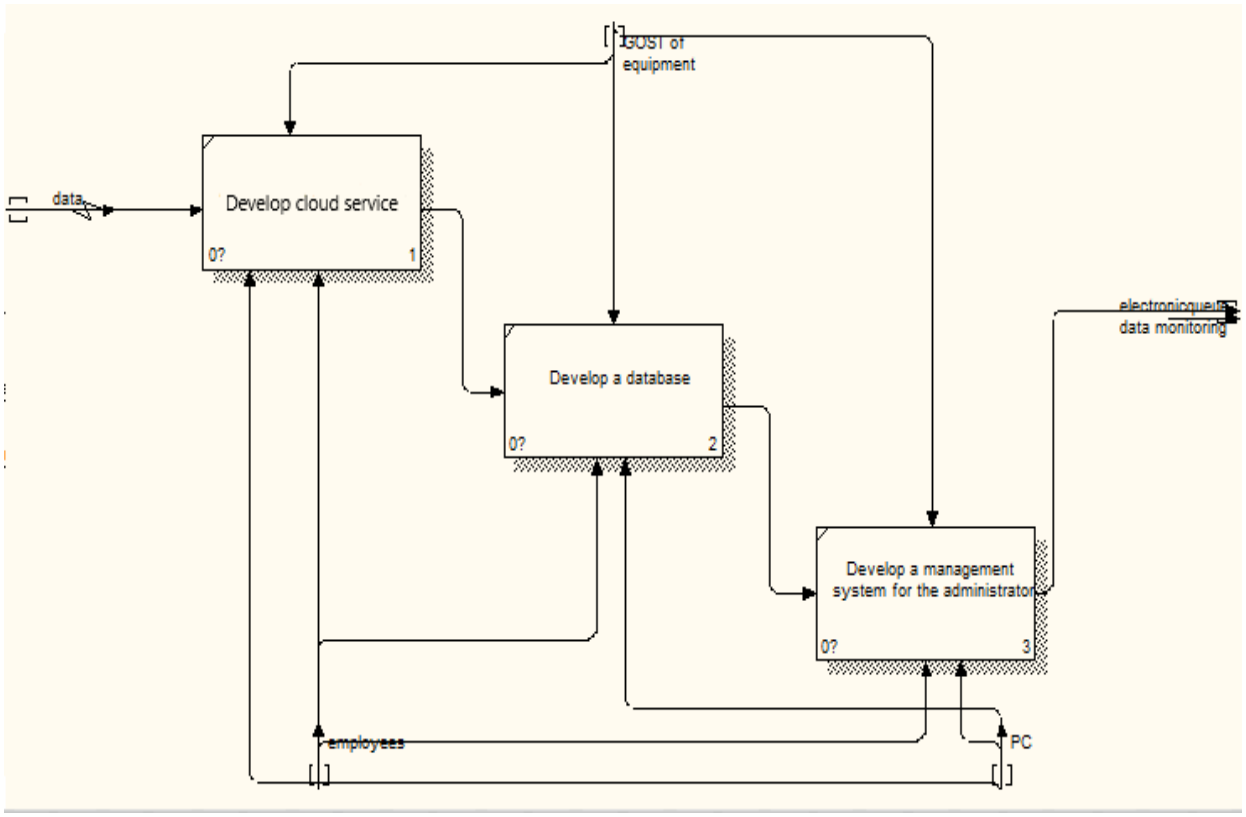


Рисунок 1.5 – Основні функції системи: рівень A.0

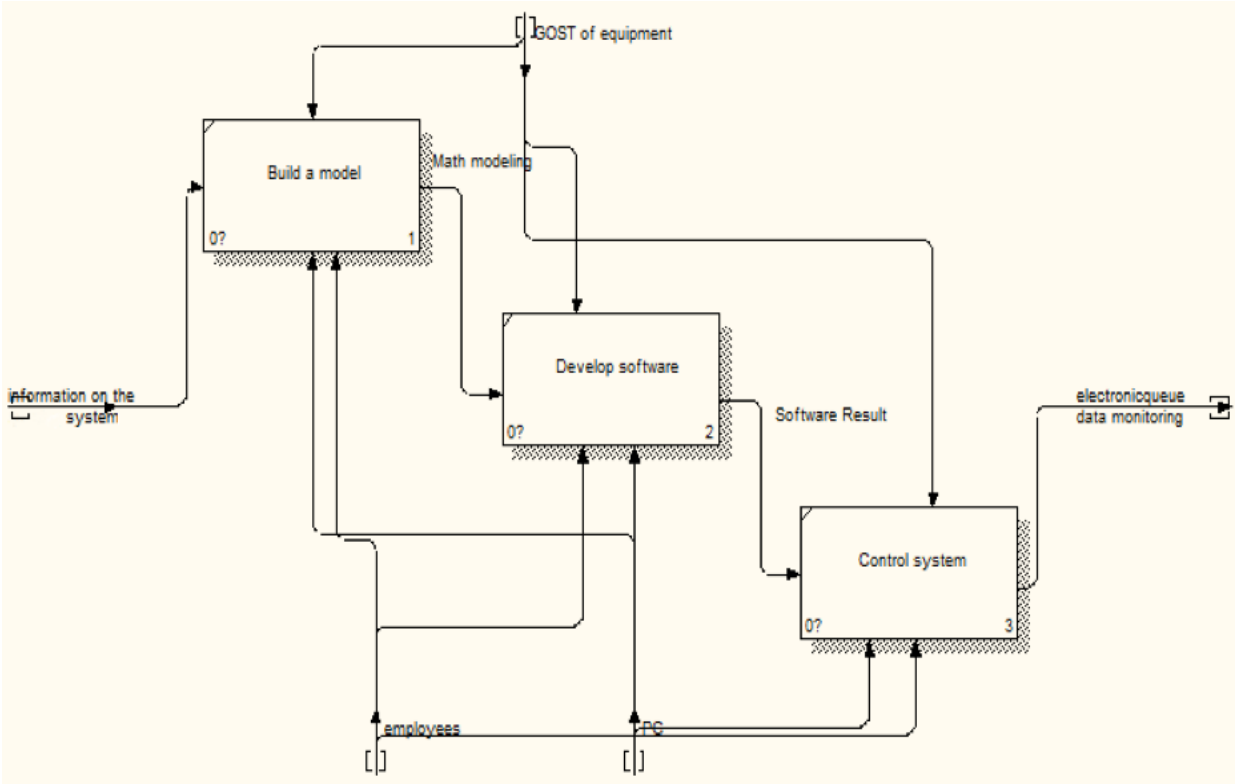


Рисунок 1.6 – Основні функції системи: рівень A.1

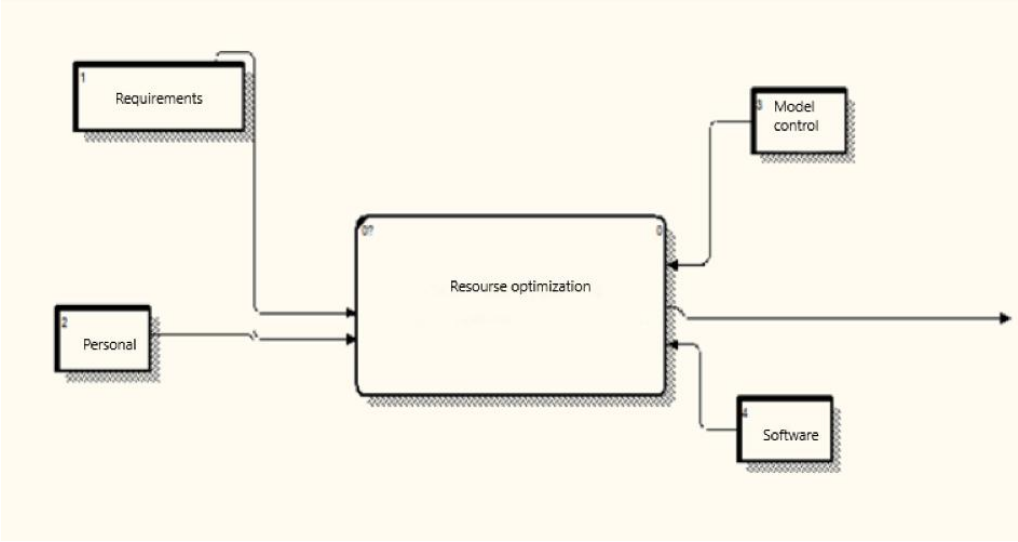


Рисунок 1.7 – DFD – діаграма 1–го рівня декомпозиції

1.2 Аналіз сценаріїв вирішення проблеми

1.2.1 Модель аналізу проблеми системи

Об’єктом дослідження є проблема, яка містить систему «оптимізація роботи хмарної системи» на основі моделі «Марківських ланцюгів».

На просторі розглянутих об’єктів виявимо ряд компонентів, що впливають на отримання необхідного результату. Ці незадоволеності розбиваються на три групи:

а) небажані властивості:

– недостатня точність обраного методу.

б) критичні властивості:

– не враховується властивість сервера.

в) бажані властивості:

– отримання наближеного рішення в аналітичному вигляді з заданою точністю;

– можливість подальшого дослідження отриманого рішення;

– підвищення керованості процесом роботи хмарного сервісу.

Проведемо аналіз незадоволеності шляхом побудови ієрархічної моделі:

а) перший рівень (фокус моделі) – невдоволень;

б) другий рівень – класифікація невдоволень;

в) третій рівень – характеристика компонентів, впливають на рішення поставленої задачі.

Аналіз вектора пріоритетів першого рівня показав, що найбільш значними виявилися бажані якості ($W_1 = 0,716$), наступні – критичні ($W_2 = 0,2$), потім небажані ($W_3 = 0,083$). На рисунку 1.8 приведена діаграма пріоритетів.

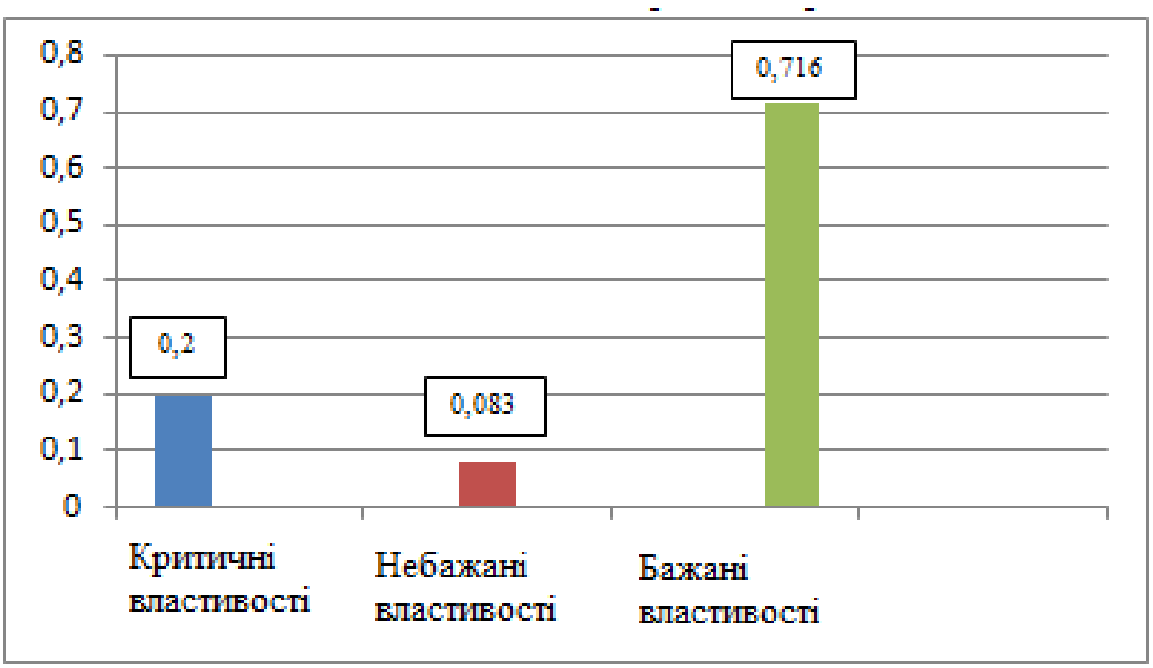


Рисунок 1.8 – Діаграма пріоритетів за критерієм порівняння «незадоволеності»

Найбільше власне значення матриці суджень одно $\lambda_{max} = 4,06$. Індекс узгодженості ІУ = 0,53 і ставлення узгодженості СУ = 0,13<0,2.

Аналіз переваги кожного з розглянутих невдоволень по відношенню до кожної якості першого рівня виявив пріоритети за критеріями порівняння «небажані якості». На рисунках 1.9 – 1.11 наведені діаграми пріоритетів по перерахованим критеріям порівняння.

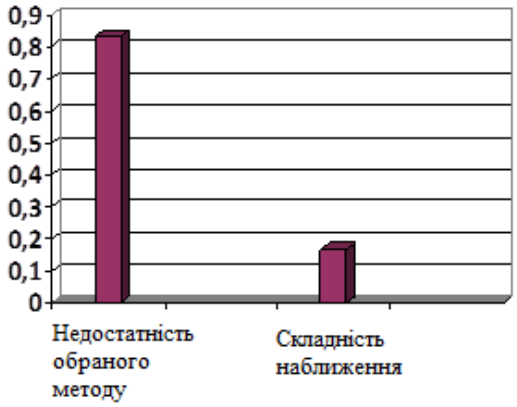


Рисунок 1.9 – Діаграма пріоритетів за критерієм порівняння «небажані властивості»

Рисунок 1.10 – Діаграма пріоритетів за критерієм порівняння «критичні властивості»

Рисунок 1.11 – Діаграма пріоритетів за критерієм порівняння «бажані властивості»

Далі необхідно здійснити синтез локальних пріоритетів, тобто оцінку узагальнених пріоритетів. Для цього слід отримати вектор глобальних пріоритетів невдоволень по відношенню до мети верхнього рівня.

В результаті отримуємо наступну діаграму, представлену на рисунку 1.12.

Рисунок 1.12 – Діаграма глобальних властивостей

Скористаємося принципом Парето для зниження розмірності. Для цього виберемо з діаграми всі незадоволеності, сума яких дорівнює 0,8. В результаті виділяємо найбільш значущі з них, вказавши на важливі властивості:

– отримання наближеного рішення в аналітичному вигляді із заданою точністю – 0,268;

– залежність траєкторії руху точки рідини від її початкового положення – 0,26;

– підвищення керованості процесом перемішування – 0,102;

– можливість подальшого дослідження отриманого рішення для знаходження періодичних точок – 0,1;

– недостатня точність обраного методу – 0,091;

– розробка можливості моделювання – 0,068.

Отже, найбільший вплив на проблему надають отримання наближеного рішення в аналітичному вигляді з заданою точністю (0,268) і залежність траєкторії руху точки від її початкового положення (0,26), тому якщо усунути дані

незадоволеності, тому можна вирішити вихідну проблему на 52,8%.

З проведеного нами аналізу незадоволеності робимо висновок, що при вирішені зазначеної проблеми зусилля будуть спрямовані на досягнення бажаної мети, сформульованої в термінах наближення до якогось ідеалу, тобто до створення ПС–системи нових бажаних властивостей. Таке рішення проблеми, зване перспективним, зменшить ймовірність прорахунку в можливих наслідках.

Ґрунтуючись на проведених аналізах невдоволень, можна сформулювати кілька можливих логічних результатів (контрастних сценаріїв) рішення задачі оптимізації роботи хмарного сервісу:

– сценарій «Точні аналітичні методи» (СЦ–1). Цей сценарій характеризує отримання точного рішення в аналітичному вигляді за допомогою теорії математичної фізики;

– сценарій «Наближені аналітичні методи» (СЦ–2). Згідно цьому сценарію рішення виходить в аналітичному вигляді застосуванням чисельних методів;

– сценарій «Сіткові методи» (СЦ–3). При цьому сценарії буде отримано наближене рішення у вигляді таблиці чисел.

Побудуємо калібровочну таблицю, в якій за психометричною шкалою Т. Сааті для кожного сценарію експертним шляхом визначені збільшення розглянутих показників.

Таблиця 1.1 – Калібровочний варіант показників

Показники стану системи	СЦ1 (0,403)	СЦ2 (0,444)	СЦ3 (0,153)	Узагальнений сценарій
Вид результатів	+3	+5	–1	+3,276
Час обчислення	+4	+2	+3	+2,959
Точність рішення	+5	+4	+2	+4,097
Універсальність методу	–1	+3	+5	+1,694

1.2.2 Модель вирішення проблеми

Для вирішення першої частини завдання необхідно вибрати метод вирішення задачі знаходження оптимізації серед точних аналітичних, чисельних і аналітико-чисельних методів. У кожної групи є свої переваги та недоліки, так наприклад, в точних аналітичних методах геометрична інформація враховується вдалим вибором системи координат; в методі комфортних відображень – побудовою відповідної відображає функції. Однак такі підходи далеко не завжди можливі. У кожному разі процес побудови рішення необхідно здійснювати заново для областей різної геометрії. Серед наближених методів вирішення крайових задач найбільш розроблені сіткові та проекційні (варіаційні) методи. Ці методи носять універсальний характер і можуть бути застосовані для різних диференціальних операторів, різних областей і типів крайових умов. Однак в сіткових методах вся вихідна інформація наводиться до цифрового вигляду і в результаті рішення задачі виходять числові масиви, що ускладнює аналітичне дослідження властивостей рішення.

Після розробки моделей контрастних сценаріїв агрегується ієрархічна модель прямого процесу аналізу проблеми.

У цій моделі розглядаються наступні елементи за рівнями:

– нульовий рівень (фокус моделі): рішення задачі моделювання;

– перший рівень (фактори): вид результатів, час обчислення, ресурси ЕОМ, точність рішення, універсальність методу;

– другий рівень (цілі акторів): отримання рішення, отримання програмного продукту, підвищення точності, можливість моделювання;

– третій рівень (контрастні сценарії): точні аналітичні методи, наближені аналітичні методи, сіткові методи.

За цією моделлю за допомогою експертних оцінок розрахуємо локальні пріоритети елементів кожного рівня моделі по відношенню до кожного пов’язаного елементу верхнього рівня, а потім оцінимо вектор глобального пріоритету елементів третього рівня.

Починаємо перший етап, який полягає в аналізі впливу факторів першого рівня ієрархії на мету аналізу – нульовий рівень.

Коротко прокоментуємо: аналіз вектора пріоритетів показує, що найбільш значним виявився фактор «точність рішення» ($W = 0,479$).

На наступному етапі розглянемо вплив цілей акторів другого рівня на фактори першого рівня, а саме до аналізу переваги кожної, з розглянутих цілей, по відношенню до кожного фактору першого рівня.

Отримані були такі значущості цілей. Для фактору «вид результату» найбільш пріоритетним є отримання адекватного рішення ($W = 0,475$). Для фактору «час обчислення» найбільш пріоритетним є підвищення точності ($W = 0,487$). Для фактору «ресурси ЕОМ» найбільш пріоритетним є отримання програмного продукту ($W = 0,648$). З точки зору точності вирішення найбільш пріоритетним виявилася мета «точність рішення». Для фактору «універсальність методу» найбільш пріоритетним є можливість моделювання.

Переходимо до аналізу сценаріїв третього рівня по відношенню до кожної з цілей акторів другого рівня. В результаті отримуємо узагальнений вектор пріоритетів сценаріїв по відношенню до кінцевої мети – вирішення задачі оптимізації роботи хмарного сервісу.

Таблиця 1.2 – Узагальнений вектор пріоритетів сценаріїв

Сценарії	Вектор пріоритетів
«Точні аналітичні методи»	0,403
«Наближені аналітичні методи»	0,444
«Сіткові методи»	0,153

Після оцінки ймовірних логічних сценаріїв стану ПС–системи починаємо розробляти моделі бажаних сценаріїв.

Таблиця 1.3 – Результати по критеріям вигоди і збитку

Сценарії Критерії	«Точні аналітичні методи»	«Наближені аналітичні методи»	«Сіткові методи»
Критерій «вигода» V	0,2006	0,5437	0,2560
Критерій «збиток» U	0,4166	0,3997	0,3837
Відношення «вигода / збиток» V/U	0,4800	1,3600	0,6672

Для вибору одного з сформованих бажаних сценаріїв агрегируя дві змістовні моделі, за якими експертним шляхом оцінимо їх коефіцієнти значущості (компоненти вектора глобальних пріоритетів) за критеріями «вигода» та «збитки». Результати аналізу занесені до таблиці 1.3.

Розглянуті бажані сценарії ранжируємо в порядку зростання відносини V/U і для подальшого розгляду вибираємо сценарій з максимальним значенням величини V/U; у нашому випадку це Сценарій №2 – «Наближені аналітичні методи». В результаті проведеного аналізу ми розглянули шляхи вирішення проблеми оптимізації роботи хмарного сервісу. Аналіз показав, що найбільш пріоритетним є сценарій «Наближені аналітичні методи».

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Хмарні технології забезпечують практично необмежену гнучкість з точки зору вибору проєкту. Але на підприємствах потрібна перевірена та узгоджена методологія щодо їх впровадження. Припускаючи, що вся або частина IT-інфраструктури знаходиться в хмарі, необхідно управляти нею в реальному часі.

Хмара забезпечує необмежені поліпшення масштабованості та маневре-

ності бізнесу. Хоча з ростом використання хмари виникає необхідність у моніторингу продуктивності мережі. Не можна просто ігнорувати час відгуку, недостатнє або надмірне використання хмарних ресурсів або розриви інформації. Моніторинг хмарної інфраструктури допомагає уважно стежити за часом реакції, доступністю, рівнями використання ресурсів, виконанням, а також за усуненням неполадок.

Моніторинг хмари – це моніторинг хмарного середовища, який включає в себе ряд кроків, включаючи аналіз, IT–моніторинг і управління хмарними додатками. Цей процес включає використання як ручного, так і автоматичного IT–моніторингу (за допомогою інструментів ITOM / NMS) і певних прийомів і методів управління хмарної інфраструктурою, щоб переконатися, що він працює оптимально і ефективно.

Це спосіб вивчення та роботи з операційним робочим процесом і процедурами для хмарної служби, додатків або ресурсу. Зазвичай це виконується за допомогою програмного забезпечення або програм механізованої перевірки, які забезпечують основний доступ і повноваження над хмарної структурою.

Адміністратори можуть перевірити робочий стан і надійність хмарних серверів і додатків. Хмара має безліч аспектів і дуже важливо відстежувати роботу ресурсів, щоб гарантувати, що все працює стабільно. Сюди можна включити наступні завдання:

– перевірка сайту: відстеження процедур, трафіку, доступності та використання ресурсів сайтів з підтримкою хмарних обчислень;

– перевірка віртуальної машини: моніторинг основи віртуалізації і окремих віртуальних машин;

– моніторинг баз даних: моніторинг форм, запитів, доступності та використання ресурсів хмарної бази даних;

– моніторинг віртуальних машин: моніторинг ресурсів віртуальної системи, гаджетів, асоціацій та виконання віртуальних машин;

– моніторинг розподіленого сховища: моніторинг запасів ресурсів і їх процедур, що надаються віртуальним машинам, адміністраторам баз даних і

додатків.

Таким чином, необхідно розробити систему для оцінки ресурсів для хмарних обчислювальних систем для перегляду стану приватної хмари в режимі онлайн з пристроїв, що використовують платформу Android, а також забезпечити прогнозування обчислювальних ресурсів приватної хмари. У систему входять: сервіс, що надає інтерфейси, через які можна взаємодіяти з приватною хмарою; клієнтський додаток на базі ОС Android, який надає користувачеві інформацію щодо використання ресурсів хмари, а також налаштування для регулювання стану хмари. Також до системи входить агент для збору інформації щодо використання ресурсів віртуальної машини та інформативний вебсайт з можливістю регулювання роботи сервіса.

1.3.2 Формальна постановка задачі

Для оцінки ресурсів хмарної обчислювальної системи використовуються дані, отримані за N днів роботи системи. Ці дані містять інформацію про навантаження на центральний процесор, оперативну пам'ять, жорсткий диск та мережевий трафік. Всі дані за N днів роботи є узагальненням отриманих даних за менші інтервали часу. Аналіз статистичних даних дозволить зрозуміти особливості розподілу навантаження на процесор та оперативну пам’ять та зробити висновки стосовно кореляції цих величин. Для цього необхідно провести розрахунок наступних ймовірнісних характеристик системи:

– вибіркове емпіричне середнє значення;

– вибіркова дисперсія;

– вибіркове середнє квадратичне відхилення;

– коефіцієнт варіації;

– вибіркові початкові та центральні моменти;

– оцінка коефіцієнта асиметрії;

– оцінка эксцесу;

– вибіркова мода;

– вибіркова медіана.

Далі необхідно провести аналіз щодо розподілу ймовірностей навантаження на центральний процесор та оперативну пам’ять. Необхідно визначити, чи є розподіл нормальним. Це дозволить зробити висновки стосовно спостережуваних випадкових величин.

Процес прогнозування полягає в обчисленні вектора розподілу ймовірностей. На вхід подається початковий вектор розподілу, після чого він множиться на стохастичну перехідну матрицю необхідну кількість разів. На виході отримуємо вектор розподілу ймовірностей, близький до стаціонарних ймовірностей, який описує ймовірності переходу системи у кожному зі станів.

Використовуючи отриманий вектор розподілу ймовірностей, можна прогнозувати навантаження на центральний процесор, оперативну пам'ять, проводити аналіз мережевого трафіку та приймати рішення щодо збільшення або зменшення числа віртуальних машин, зміни конфігурації сервісу тощо.

Нехай M – це стохастична матриця для ланцюга Маркова з n – кількістю станів, а V – початковий вектор розподілу. Тоді вектор розподілу ймовірностей:

$$V_{i+1} = V_i \bullet M \ ,$$

де V_i – вектор розподілу;

M – стохастична матриця.

Таким чином можна зробити прогноз обчислювальних ресурсів на кожний часовий крок і перевірити ступінь надійності прогнозу ресурсів.

Умова оптимальності прогнозу для стаціонарного стану

$$\left| p_i - v_{i,\,t} \right| \le \varepsilon = 0.001,$$

де p_i – стаціонарна ймовірність стану s_i;

$v_{i,\,t}$ – ймовірність вектора розподілу на момент t;

ε – точність обчислень.

Умова оптимальності прогнозу для нестаціонарного стану:

$$\left|v_{i,\,t-1} - v_{i,\,t}\right| \le \varepsilon = 0.001,$$

де $v_{i,\,t}$ – ймовірність вектора розподілу на момент t;

ε – точність обчислень.

Помилка прогнозу визначається Евклідовою метрикою:

$$d\left(\vec{x},\vec{y}\right) = \sqrt{\sum_{i=1}^{n}\left(x_i - y_i\right)^2},$$

де $d\left(\vec{x},\vec{y}\right)$ – помилка прогнозування;

$\vec{x}$ – спрогнозовані дані вектора розподілення навантаження на центральний процесор;

$\vec{y}$ – експерементальні дані стосовно навантаження на центральний процесор.

1.3.3 Постановка задачі дослідження

У роботі необхідно провести дослідження оцінки ресурсів для хмарних обчислювальних систем. Для цього використовується мобільна система моніторингу – система реєстрації та збору, зберігання та аналізу даних про роботу віртуальних машин у приватній хмарі, що дозволяє переглядати, аналізувати та управляти системою з мобільного пристрою.

Для мобільних пристроїв віртуалізована система може бути реалізована за допомогою SOA архітектури у приватній хмарі, яка, в свою чергу вимагає

використання ефективних алгоритмів. SOA архітектура дозволить зробити унікальний інтерфейс, з яким буде зручно взаємодіяти усім компонентам системи. Надалі це дозволить масштабувати систему та додавати нові додатки, наприклад, додаток для операційної системи iOS.

Дана робота спрямована на аналіз даних та можливий прогноз обчислювальних ресурсів, розробку алгоритмів для системи мобільного моніторингу, обґрунтування цих алгоритмів з точки зору теорії імовірностей та математичної статистики, реалізацію цих алгоритмів та перевірку отриманих результатів.

У ході виконання дипломної роботи необхідно:

– розглянути систему з точки зору математичної статистики;

– проаналізувати характеристики системи та зробити висновки;

– побудувати модель на базі ланцюга Маркова;

– побудувати діаграму зміни станів;

– виконати прогноз ресурсів та зробити висновки;

– реалізувати розроблені алгоритми у хмарному сервісі;

– провести тестування працездатності системи;

– зробити висновки з отриманих результатів та скласти план подальшої роботи.

2 АЛГОРИТМИ ТА МЕТОДИ МОБІЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ

2.1 Опис системи з точки зору математичної статистики

Основною перевагою мобільної системи моніторингу є можливість прогнозування обчислювальних ресурсів для віртуальної машини. Це дозволяє провайдеру хмарної інфраструктури адаптувати віртуальні ресурси за вимогами. Таким чином, модель має величезний вплив на вартість та ефективність використання обчислювальних ресурсів.

Для того, щоб оптимальним чином адаптувати систему за вимогами, необхідно спочатку описати її з точки зору математичної статистики та побудувати для неї математичну модель, яка дозволить зробити певні висновки стосовно обчислювальних ресурсів. Розглянемо систему з точки зору математичної статистики.

У розроблюваній системі піддамо аналізу кілька випадкових величин: навантаження на центральний процесор, використання оперативної пам'яті, завантаженість жорсткого диска та використання мережевого трафіку. Ці випадкові величини приймають значення від 0% до 100% та мають одиниці виміру:

– центральний процесор GHz;

– оперативна пам'ять Gb;

– жорсткий диск Gb;

– мережевий трафік Mbps.

Як приклад для побудови моделей використовується навантаження на центральний процесор та навантаження на оперативну пам'ять.

Всі дані для розрахунків взяті з лог-файлів у форматі JSON, згенерованих на віртуальній машині. Для генерування лог-файлів система використовує задані інтервали часу (хвилина, година, день, місяць, рік). Кожен інтервал включає в себе менші інтервали. Наприклад, інтервал «година» складається з 60 інтервалів «хвилина». Таким чином, значення випадкових величин для

інтервалу «година», дорівнюють усередненим значенням випадкових величин інтервалів «хвилина». Показання знімаються кожен раз через заданий інтервал часу. Як приклад, у розрахунках використовується інтервал «день».

Стан системи змінюється випадковим чином, отже, послідовність станів системи утворює випадковий процес. Цей випадковий процес є Марківським, тому що ймовірність будь-якого стану системи залежить тільки від її поточного стану, та не залежить від того, як система прийшла у цей стан.

Ланцюг Маркова характеризується наступними властивостями:

– вектор розподілу ймовірностей:

$$\left(p(s_1), p(s_2), \ldots, p(s_n)\right), \tag{2.1}$$

де $p(s_i)$ – ймовірність переходу системи у стан s_i;

n – кількість станів системи.

– матриця ймовірностей переходу:

$$M_{ij} = p\left(s_i \to s_j\right), \tag{2.2}$$

де M_{ij} – матриця ймовірностей переходу;

$p\left(s_i \to s_j\right)$ – умовна ймовірність переходу системи зі стану s_i у стан s_j.

Матриця M_{ij} використовується для опису переходів між станами ланцюга Маркова і є стохастичною матрицею, тобто:

$$\sum_j p\left(s_i \to s_j\right) = 1, \tag{2.3}$$

де $p\left(s_i \to s_j\right)$ – умовна ймовірність переходу системи зі стану s_i у стан s_j.

Переходи з одного стану в інший можливі у строго певні моменти часу t_i,

отже, розроблювану систему описує випадковий Марківський процес з дискретним часом.

Початковий розподіл ланцюга Маркова визначається вектором, що має наступний вигляд:

$$V = (v_1, v_2, \ldots, v_n), \tag{2.4}$$

$$p(s_0 = i) = v_i, \tag{2.5}$$

де V – початковий вектор розподілу;

v_i – ймовірність переходу системи у стан s_i;

n – кількість станів системи;

s_0 – початковий стан системи.

2.2 Дослідження характеристик системи та побудова статисти стичної моделі

Перед побудовою математичної моделі проведено групування отриманих даних. Як приклад, у розрахунках використовується навантаження на центральний процесор та навантаження на оперативну пам'ять.

Відсоток навантаження процесора змінюється від 0% до 100%, отже, можна розбити цей інтервал на менші інтервали. Визначимо кожен отриманий інтервал як стан моделі.

Нехай $s_1, s_2, \ldots, s_n$ – це стани системи. Візьмемо дані за 75 днів роботи системи та розіб'ємо їх на інтервали. У зібраних експериментальних даних навантаження процесора потрапляє в інтервали від 0% до 40%, тому що на протязі заданого проміжку часу навантаження не перевищувало 40%. Покладемо ширину інтервала $d=5$ та розіб'ємо інтервал на $n=8$ менших інтервалів:

$$\begin{cases} s_1 = x \in [0,5)\%, \\ s_2 = x \in [5,10)\%, \\ s_3 = x \in [10,15)\%, \\ s_4 = x \in [15,20)\%, \\ s_5 = x \in [20,25)\%, \\ s_6 = x \in [25,30)\%, \\ s_7 = x \in [30,35)\%, \\ s_8 = x \in [35,40)\%, \end{cases} \tag{2.6}$$

де $s_1, s_2, \ldots, s_n$ – стани системи у дискретні моменти часу $t_1, t_2, \ldots, t_n$;

x – значення навантаження на центральний процесор.

Дані, отримані за 75 днів роботи системи, представлені у вигляді лог–файлу у форматі JSON, який містить інформацію про навантаження на центральний процесор, оперативну пам'ять, жорсткий диск та мережевий трафік. Таким чином, усі розрахунки проводяться для інтервалу «день». Приклад такого файлу наведений на рисунку 2.1.

Всі дані за 75 днів роботи є узагальненням отриманих даних за менші інтервали часу. Для групування даних використовується алгоритм MapReduce. Наприклад, система збирає інформацію у лог-файли кожну годину, потім обчислює середні значення навантаження на систему та групує їх у лог-файл за день. Таким чином, система дозволяє регулювати точність обчислень у залежності від обраного інтервалу часу t.

Проведемо розрахунок ймовірності кожного стану. Розрахуємо ймовірність стану s_1. Під ймовірністю стану s_1 розуміється загальне число переходів $s_1 \to s_1, s_2, \ldots, s_n$ поділене на загальне число можливих подій [3]:

$$p(s_1) = \frac{m_1}{\sum m_i}, \tag{2.7}$$

де $p(s_1)$ – ймовірність стану s_1;

m_1 – загальна кількість переходів $s_1 \to s_1, s_2, \ldots, s_n$.


```
{
  "unit": "day",
  "timestamp": 1399023949305,
  "data": [
    { "cpu": 7.04, "mem": 5.02, "disk": 26.01, "net": 10.2 }, { "cpu": 12.07, "mem": 8.13, "disk": 26.0, "net": 12.2 },
    { "cpu": 5.4, "mem": 7.1, "disk": 25.9, "net": 10.2 }, { "cpu": 4.94, "mem": 5.62, "disk": 25.9, "net": 10.0 },
    { "cpu": 6.67, "mem": 5.63, "disk": 25.9, "net": 10.2 }, { "cpu": 4.94, "mem": 5.5, "disk": 25.8, "net": 10.5 },
    { "cpu": 5.18, "mem": 6.1, "disk": 25.9, "net": 14.1 }, { "cpu": 5.16, "mem": 5.02, "disk": 25.9, "net": 15.2 },
    { "cpu": 4.79, "mem": 4.2, "disk": 25.6, "net": 17.3 }, { "cpu": 5.38, "mem": 7.12, "disk": 25.6, "net": 18.1 },
    { "cpu": 4.95, "mem": 7.2, "disk": 25.6, "net": 14.2 }, { "cpu": 5.42, "mem": 8.1, "disk": 25.6, "net": 13.2 },
    { "cpu": 6.4, "mem": 10.1, "disk": 25.7, "net": 20.2 }, { "cpu": 12.33, "mem": 15.02, "disk": 25.7, "net": 22.1 },
    { "cpu": 4.68, "mem": 14.8, "disk": 25.6, "net": 18.4 }, { "cpu": 6.22, "mem": 11.3, "disk": 25.6, "net": 14.7 },
    { "cpu": 12.66, "mem": 15.5, "disk": 25.9, "net": 10.2 }, { "cpu": 10.4, "mem": 15.6, "disk": 25.9, "net": 11.4 },
    { "cpu": 5.7, "mem": 3.1, "disk": 25.9, "net": 8.1 }, { "cpu": 6.1, "mem": 5.4, "disk": 25.7, "net": 11.2 },
    { "cpu": 20.2, "mem": 15.1, "disk": 26.4, "net": 33.2 }, { "cpu": 7.5, "mem": 16.0, "disk": 26.3, "net": 27.2 },
    { "cpu": 3.6, "mem": 6.3, "disk": 26.3, "net": 28.2 }, { "cpu": 2.9, "mem": 5.6, "disk": 26.4, "net": 22.5 },
    { "cpu": 2.3, "mem": 5.7, "disk": 26.4, "net": 22.2 }, { "cpu": 3.6, "mem": 5.8, "disk": 26.4, "net": 17.2 },
    { "cpu": 9.3, "mem": 5.02, "disk": 26.3, "net": 18.1 }, { "cpu": 14.8, "mem": 5.13, "disk": 26.4, "net": 18.0 },
    { "cpu": 12.3, "mem": 7.1, "disk": 26.4, "net": 19.1 }, { "cpu": 9.88, "mem": 8.8, "disk": 26.3, "net": 13.6 },
    { "cpu": 10.6, "mem": 9.9, "disk": 26.2, "net": 20.2 }, { "cpu": 30.1, "mem": 35.7, "disk": 26.6, "net": 68.2 },
    { "cpu": 15.71, "mem": 38.2, "disk": 26.8, "net": 66.1 }, { "cpu": 16.0, "mem": 33.2, "disk": 26.8, "net": 64.7 },
    { "cpu": 14.67, "mem": 35.02, "disk": 26.9, "net": 50.2 }, { "cpu": 17.38, "mem": 36.27, "disk": 27.0, "net": 55.1 },
    { "cpu": 16.0, "mem": 35.7, "disk": 27.1, "net": 66.2 }, { "cpu": 21.94, "mem": 37.02, "disk": 27.2, "net": 71.4 },
    { "cpu": 24.6, "mem": 35.9, "disk": 27.2, "net": 72.2 }, { "cpu": 19.0, "mem": 33.8, "disk": 27.4, "net": 68.5 },
    { "cpu": 25.0, "mem": 35.2, "disk": 27.4, "net": 48.4 }, { "cpu": 30.2, "mem": 36.1, "disk": 27.4, "net": 39.1 },
    { "cpu": 28.04, "mem": 36.6, "disk": 27.3, "net": 28.4 }, { "cpu": 13.2, "mem": 36.1, "disk": 27.3, "net": 28.0 },
    { "cpu": 19.66, "mem": 33.3, "disk": 27.3, "net": 26.3 }, { "cpu": 33.1, "mem": 34.2, "disk": 27.3, "net": 27.1 },
    { "cpu": 11.2, "mem": 27.4, "disk": 27.3, "net": 17.5 }, { "cpu": 14.17, "mem": 24.2, "disk": 27.2, "net": 16.2 },
    { "cpu": 5.6, "mem": 18.1, "disk": 27.2, "net": 17.1 }, { "cpu": 17.05, "mem": 16.1, "disk": 27.2, "net": 22.5 },
    { "cpu": 8.3, "mem": 13.4, "disk": 27.2, "net": 23.3 }, { "cpu": 5.1, "mem": 11.1, "disk": 27.2, "net": 22.6 },
    { "cpu": 13.16, "mem": 15.2, "disk": 27.2, "net": 23.5 }, { "cpu": 25.74, "mem": 17.83, "disk": 27.3, "net": 35.6 },
    { "cpu": 22.0, "mem": 16.6, "disk": 27.4, "net": 44.1 }, { "cpu": 23.2, "mem": 17.4, "disk": 27.5, "net": 55.4 },
    { "cpu": 22.42, "mem": 22.4, "disk": 27.5, "net": 35.4 }, { "cpu": 24.88, "mem": 25.02, "disk": 27.5, "net": 33.2 },
    { "cpu": 23.78, "mem": 25.42, "disk": 27.5, "net": 32.1 }, { "cpu": 22.51, "mem": 27.5, "disk": 27.5, "net": 24.3 },
    { "cpu": 35.0, "mem": 35.02, "disk": 23.2, "net": 10.2 }, { "cpu": 39.9, "mem": 37.02, "disk": 23.2, "net": 9.1 },
    { "cpu": 36.63, "mem": 35.1, "disk": 23.2, "net": 8.9 }, { "cpu": 34.6, "mem": 35.02, "disk": 23.2, "net": 7.2 },
    { "cpu": 33.9, "mem": 32.1, "disk": 23.2, "net": 11.1 }, { "cpu": 33.4, "mem": 31.1, "disk": 23.2, "net": 12.2 },
    { "cpu": 37.8, "mem": 39.9, "disk": 23.2, "net": 14.7 }, { "cpu": 29.7, "mem": 34.1, "disk": 23.2, "net": 15.0 },
    { "cpu": 27.7, "mem": 23.2, "disk": 23.2, "net": 14.9 }, { "cpu": 25.55, "mem": 25.02, "disk": 23.2, "net": 14.0 },
    { "cpu": 24.43, "mem": 15.2, "disk": 23.2, "net": 20.2 }, { "cpu": 29.0, "mem": 12.0, "disk": 23.2, "net": 21.6 },
    { "cpu": 20.0, "mem": 11.1, "disk": 23.2, "net": 22.1 }, { "cpu": 6.2, "mem": 7.1, "disk": 23.0, "net": 25.1 },
    { "cpu": 5.23, "mem": 15.02, "disk": 23.1, "net": 21.9 }
  ]
}
```


Рисунок 2.1 – Приклад лог-файлу з даними про використання ресурсів системи

Таким чином, для стана s_1 центрального процесора отримаємо ймовірність $p(s_1) = \frac{9}{75} = 0.12$. Аналогічним чином розрахуємо ймовірності для всіх станів центрального процесора та оперативної пам'яті. Результат розрахунків наведено в таблиці 2.1.

Аналогічно можна розрахувати вектор розподілу ймовірностей для навантаження на жорсткий диск та мережевий трафік.

Використовуючи результати обчислень, будуються гістограми для навантаження на центральний процесор та оперативну пам'ять (рис. 2.3 – 2.4).

На рисунках видно, що розподіл відрізняється від нормального розподілу.

Таблиця 2.1 – Результати обчислень ймовірностей для станів центрального процесора та оперативної пам'яті

Стан системи	Середина інтервала, u_i	Навантаження на центральний процесор		Навантаження на оперативну пам'ять	
		Число значень, m_i	Ймовірність попадання у інтервал, p_i	Число значень, m_i	Ймовірність попадання у інтервал, p_i
$s_1:[0,5)\%$	2.5	9	0.12	2	0.027
$s_2:[5,10)\%$	7.5	19	0.253	22	0.293
$s_3:[10,15)\%$	12.5	12	0.16	7	0.093
$s_4:[15,20)\%$	17.5	8	0.107	12	0.16
$s_5:[20,25)\%$	22.5	10	0.133	4	0.053
$s_6:[25,30)\%$	27.5	7	0.093	5	0.067
$s_7:[30,35)\%$	32.5	6	0.08	7	0.093
$s_8:[35,40)\%$	37.5	4	0.053	16	0.213

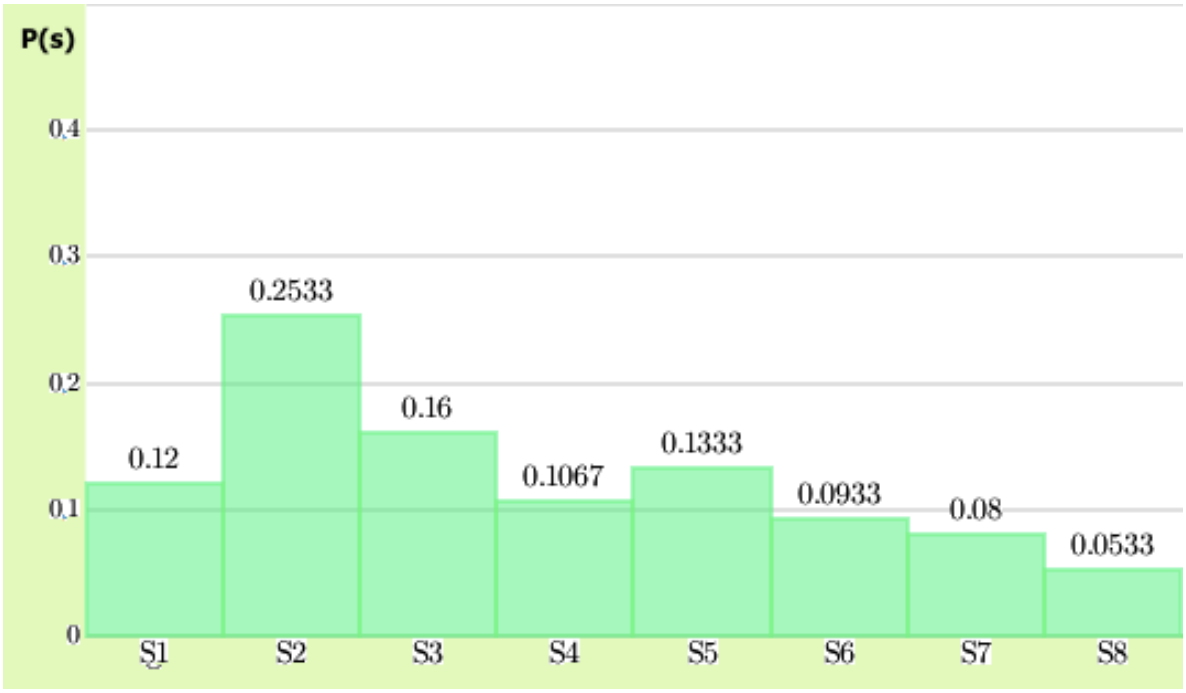


Рисунок 2.2 – Гістограма для навантаження на центральний процесор

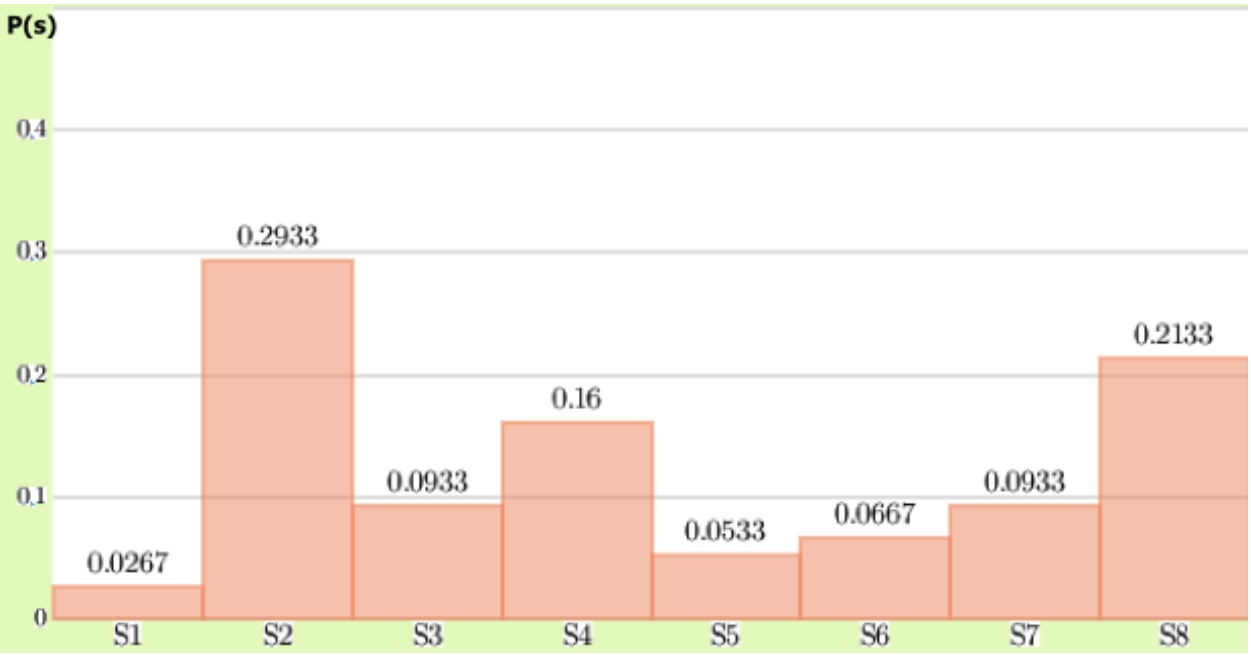


Рисунок 2.3 – Гістограма для навантаження на оперативну пам'ять

Тепер необхідно провести аналіз статистичних даних. Аналіз статистичних даних дозволить зрозуміти особливості розподілу навантаження на процесор та оперативну пам’ять та зробити висновки стосовно кореляції цих величин [4]. Для цього проведемо розрахунок наступних ймовірнісних характеристик системи:

– вибіркове емпіричне середнє значення;

– вибіркова дисперсія;

– вибіркове середнє квадратичне відхилення;

– коефіцієнт варіації;

– вибіркові початкові та центральні моменти;

– оцінка коефіцієнта асиметрії;

– оцінка эксцесу;

– вибіркова мода;

– вибіркова медіана.

Для обчислень будемо використовувати формули інтервального групування, тому що усі дані розподілені по інтервалах, кожен з яких описується відповідним станом.

Вибіркове емпіричне середнє – це арифметичне середнє спостережень. Воно характеризує наближення теоретичного середнього розподілу та обчислюється за формулою [5]:

$$\overline{x} = \sum_{i=1}^{n} p_i u_i, \tag{2.8}$$

де $\overline{x}$ – вибіркове середнє;

n – кількість станів системи;

p_i – ймовірність i–го стану;

u_i – середина i–го інтервала.

Після проведення розрахунків отримуємо значення вибіркового середнього для навантаження на центральний процесор $\overline{x} = 16.208$. Результати розрахунків говорять про те, що у середньому, навантаження на процесор складає 16.208%. Для навантаження на оперативну пам'ять вибіркове середнє складає $\overline{x} = 20.273$.

Перейдемо до розрахунку вибіркової дисперсії. Вибіркова дисперсія є мірою розкиду випадкової величини, тобто її відхилення від математичного очікування, та обчислюється за формулою:

$$S^2 = \frac{1}{c-1} \sum_{i=1}^{n} m_i \left(u_i - \overline{x}\right)^2, \tag{2.9}$$

де S^2 – вибіркова дисперсія;

c – загальна кількість подій;

n – кількість станів системи;

m_i – кількість подій i–го стану;

u_i – середина інтервала s_i;

$\overline{x}$ – вибіркове середнє.

У результаті розрахунків отримуємо значення вибіркової дисперсії для навантаження на центральний процесор $S^2 = 108.172$. Для навантаження на оперативну пам'ять $S^2 = 146.109$, що говорить про більш високий рівень розкиду значень. S^2 є незміщеною оцінкою.

Проведемо розрахунок вибіркового середнього квадратичного відхилення. Вибіркове середнє квадратичне відхилення дозволяє визначити ступінь розсіювання значень відносно вибіркового середнього та визначається як корінь з вибіркової дисперсії:

$$S = \sqrt{\frac{1}{c-1}\sum_{i=1}^{n} m_i \left(u_i - \overline{x}\right)^2}, \qquad (2.10)$$

де S – вибіркове середнє квадратичне відхилення;

c – загальна кількість подій;

n – кількість станів системи;

m_i – кількість подій i–го стану;

u_i – середина інтервала s_i;

$\overline{x}$ – вибіркове середнє.

Після проведення розрахунків отримуємо значення вибіркового середнього квадратичного відхилення для навантаження на центральний процесор $S = 10.401$. Для навантаження на оперативну пам’ять вибіркове середнє квадратичне відхилення дорівнює $S = 12.088$.

Розрахуємо коефіцієнт варіації. Коефіцієнт варіації є мірою відносного розкиду випадкової величини і показує, яку долю середнього значення цієї величини складає її середній розкид. Таким чином, коефіцієнт варіації можна розрахувати за формулою:

$$V = S \diagup \overline{x}, \quad \overline{x} \neq 0, \qquad (2.11)$$

де V – коефіцієнт варіації;

S – вибіркове середнє квадратичне відхилення;

$\overline{x}$ – вибіркове середнє.

У результаті розрахунків отримуємо значення коефіцієнта варіації для навантаження на центральний процесор $V = 0.642.$ Для навантаження на оперативну пам'ять коефіцієнт варіації складає $V = 0.596$.

Вибіркові початкові та центральні моменти порядка l будемо визначати за формулами:

$$\alpha_l = \frac{1}{c}\sum_{i=1}^{n} m_i u_i^l, \tag{2.12}$$

$$\beta_l = \frac{1}{c}\sum_{i=1}^{n} m_i \left(u_i - \overline{x}\right)^l, \tag{2.13}$$

де α_l – вибірковий початковий момент порядка l;

β_l – вибірковий початковий момент порядка l;

c – загальна кількість подій;

n – кількість станів системи;

m_i – кількість подій i–го стану;

u_i – середина інтервала s_i;

$\overline{x}$ – вибіркове середнє.

Наведені вище формули можна використовувати при оцінюванні коефіцієнта асиметрії та ексцесу.

Коефіцієнт асиметрії характеризує симетричність розподілу випадкової величини відносно її вибіркового середнього. Оцінку коефіцієнта асиметрии можна отримати наступним чином:

$$A = \frac{\beta_3}{\left(\beta_2\right)^{3/2}}, \tag{2.14}$$

де A – коефіцієнт асиметрії;

β_l – вибірковий початковий момент порядка l.

Після проведення обчислень отримуємо оцінку коефіцієнта асиметрії для навантаження на центральний процесор що дорівнює $A = 0.494$. Для навантаження на оперативну пам'ять отримуємо значення $A = 0.277$. Можна зробити висновок, що розподіл має помірну асиметрію.

Оцінку ексцеса будемо шукати наступним чином:

$$E = \frac{\beta_4}{\beta_2^2} - 3, \tag{2.15}$$

де E – коефіцієнт ексцеса;

β_l – вибірковий початковий момент порядка l.

У результаті розрахунків отримуємо оцінку ексцеса для навантаження на центральний процесор $E = -0.889$. Для навантаження на оперативну пам'ять отримуємо коефіцієнт ексцеса $E = -1.466$. Отримані значення говорять про те, що розподіл має плоску вершину.

Оцінимо вибіркову моду розподілу. При використанні інтервального групування вибирається інтервал, якому відповідає найбільша частота. Нехай це k-й інтервал $(q_{k-1} - q_k)$, його частота дорівнює m_k, а його ширина дорівнює d, тоді:

$$M_0 = q_{k-1} + d\frac{m_k - m_{k-1}}{2m_k - m_{k-1} - m_{k+1}}, \tag{2.16}$$

де M_0 – вибіркова мода розподілу;

q_{k-1} – початок k–го інтервала;

d – ширина інтервала;

m_k – кількість значень у k–му інтервалі.

Ширина інтервала дорівнює $d = 5$, найбільша кількість відповідає другому інтервалу $k = 2$, отже, згідно з формулою 2.16 вибіркова мода навантаження на центральний процесор дорівнює $M_0 = 7.941$. Аналогічно, для навантаження на оперативну пам'ять мода дорівнює $M_0 = 7.857$. Вибіркова мода представляє собою значення, що найбільш часто зустрічається у вибіркових спостереженнях.

Перед розрахунком медіани необхідно визначити медіанний інтервал, номер якого визначається завдяки нерівностям:

$$\sum_{i<k} m_i \le \frac{c}{2}, \quad \sum_{i \le k} m_i > \frac{c}{2}, \tag{2.17}$$

де k – номер медіанного інтервала;

m_i – кількість значень у i –му інтервалі;

c – загальна кількість подій.

Проведемо розрахунок вибіркової медіани. Вибіркова медіана це величина, що знаходиться в середині ряду величин, розташованих у зростаючому або спадному порядку. Таким чином, медіану оцінюють так:

$$M_e = q_{k-1} + d\left(\frac{\frac{c}{2} - \sum_{i<k} m_i}{m_k}\right), \tag{2.18}$$

де M_e – вибіркова медіана;

k – номер медіанного інтервала;

q_{k-1} – початок медіанного інтервала;

d – ширина інтервала;

c – загальна кількість подій;

m_k – кількість значень у медіанному інтервалі.

Таким чином, значення медіани $M_e = 14.375$ для навантаження на центральний процесор та $M_e = 18.125$ для навантаження на оперативну пам'ять.

Результати розрахунків характеристик наведені у таблиці 2.2.

Таблиця 2.2 – Результати розрахунків характеристик системи

Параметр вибіркового розподілу	Навантаження на центральний процесор	Навантаження на оперативну пам'ять
Вибіркове середнє	16.208	20.273
Вибіркова дисперсія	108.172	146.109
Вибіркове середнє квадратичне відхилення	10.401	12.088
Коефіцієнт варіації	0.642	0.596
Оцінка асиметрії	0.494	0.277
Оцінка ексцеса	–0.889	–1.466
Вибіркова мода	7.941	7.857
Вибіркова медіана	14.375	18.125

Проведемо аналіз щодо розподілу ймовірностей навантаження на центральний процесор та оперативну пам’ять.

Необхідно визначити, чи є розподіл нормальним. Це дозволить зробити висновки стосовно спостережуваних випадкових величин [6]. Перевірка характеристик системи на нормальність наведена у таблиці 2.3.

Аналіз показує, що розподіл навантаження на центральний процесор та навантаження на оперативну пам'ять не задовольняє умовам нормальності. Тому у розрахунках не можливо буде використовувати деякі критерії, наприклад критерії Стьюдента, тому що він потребує щоб випадкова величина була розподілена нормально.

У складній системі важливо знати, які фактори найбільш тісно взаємодіють один з одним. Розуміння деякого взаємозв'язку може дозволити використовувати інформацію щодо однієї змінної для передбачення значень іншої змінної.

Таблиця 2.3 – Перевірка результатів розрахунків на нормальність розподілу

Критерій	Нормальний розподіл	Розподіл навантаження на центральний процесор	Розподіл навантаження на оперативну пам'ять
Кількість значень в інтервалах $\bar{x} \pm S$, $\bar{x} \pm 2S$, $\bar{x} \pm 3S$	68%, 95%, 100%	57.3%, 97.3%, 100%	44%, 100%, 100%
Величина коефіцієнта варіації	$V < 0.33$	0.642	0.596
Оцінка ексцеса та асиметрії	$E = 0,\ A = 0$	$E = -0.889,$ $A = 0.494$	$E = -1.466,$ $A = 0.277$
Різниця між вибірковим середнім та вибірковою медіаною	$\lvert x - M_e \rvert \approx 0$	1.833	2.148

Для цього знайдемо вибірковий коефіцієнт кореляції між навантаженням на центральний процесор та навантаженням на оперативну пам’ять. Вибірковий коефіцієнт кореляції служить мірою сили та напрямку лінійного зв'язку між двома змінними та визначається за формулою [7]:

$$r = \frac{\sum u_i v_i - n\overline{xy}}{cS_x S_y}, \tag{2.19}$$

де r – вибірковий коефіцієнт кореляції;

u_i – середина інтервала навантаження на процесор;

v_i – середина інтервала навантаження на оперативну пам’ять;

$\bar{x}$ – вибіркове середнє навантаження на процесор;

$\bar{y}$ – вибіркове середнє навантаження на оперативну пам’ять;

S_x – вибіркове середнє квадратичне відхилення навантаження на центральний процесор;

S_y – вибіркове середнє квадратичне відхилення навантаження на оперативну

пам’ять.

У результаті розрахунків отримали значення кореляції $r = 0.782$, що говорить про високий рівень зв'язку між навантаженням на центральний процесор та навантаженням на оперативну пам'ять.

Побудуємо статистичну модель прогнозування. У статистичних моделях залежність майбутнього значення від минулого задається у вигляді деякого рівняння. До статистичних моделей належать моделі, засновані на лінійній та нелінійній регресії. Опишемо кореляційний зв'язок між змінними, використовуючи рівняння регресії:

$$y - \overline{y} = r \frac{S_y}{S_x} \left(x - \overline{x} \right), \tag{2.20}$$

де y – значення навантаження на оперативну пам’ять;

x – значення навантаження на центральний процесор;

r – вибірковий коефіцієнт кореляції;

$\overline{x}$ – вибіркове середнє навантаження на процесор;

$\overline{y}$ – вибіркове середнє навантаження на оперативну пам’ять;

S_x – вибіркове середнє квадратичне відхилення навантаження на центральний процесор;

S_y – вибіркове середнє квадратичне відхилення навантаження на оперативну пам’ять.

Регресійний аналіз — розділ математичної статистики, присвячений методам аналізу залежності однієї величини від іншої. На відміну від кореляційного аналізу не з'ясовує чи істотний зв'язок, а займається пошуком моделі цього зв'язку, вираженої у функції регресії.

Підставивши значення у формулу 2.20, отримаємо наступне рівняння регресії:

$$y = 0.909x + 5.541, \quad (2.21)$$

де y – значення навантаження на оперативну пам’ять;

x – значення навантаження на центральний процесор.

Представимо результати регресіонного аналізу у вигляді діаграми розсіювання. На підставі регресіонного анализу хмара точок апроксимується рівнянням регресії. На графіку видно, що залежність задана рівнянням прямої, отже має місце лінійна регресійна модель (рис. 2.4).

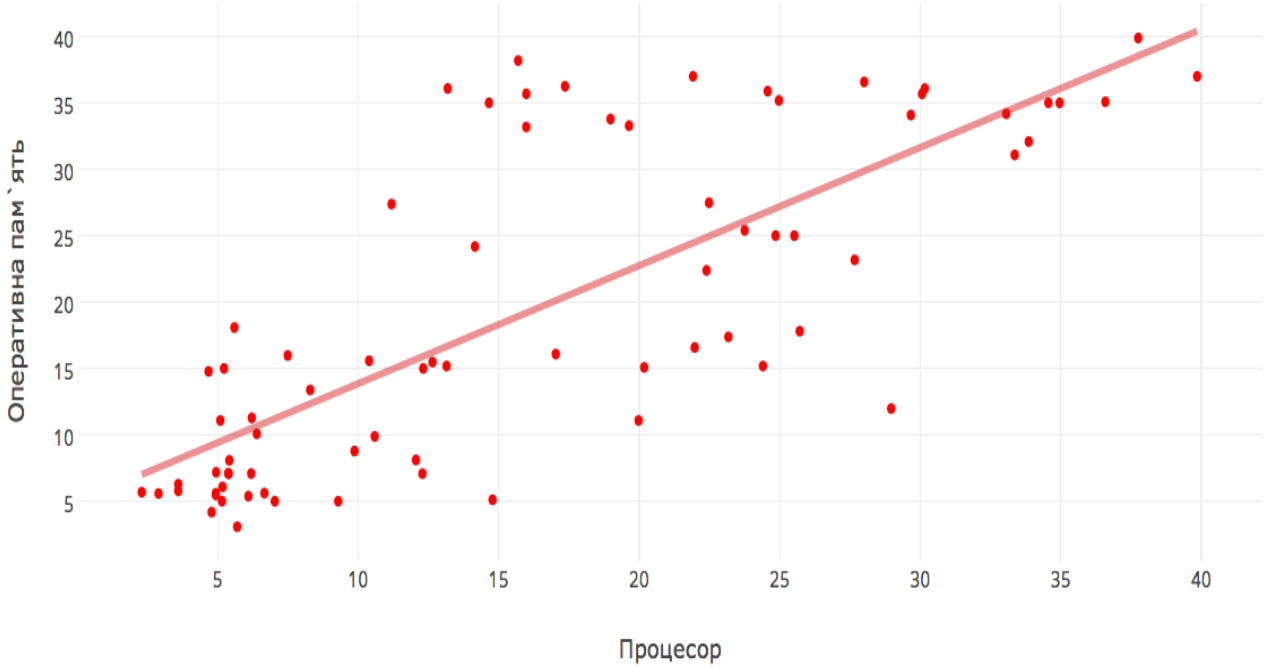


Рисунок 2.4 – Діаграма розсіювання

В результаті регресійного аналізу була встановлена залежність між ресурсами процесора та ресурсами оперативної пам'яті $r = 78.2\%$. Згідно зі шкалою Чеддока зв'язок між величинами можна охарактеризувати як сильний. Цю інформацію можна використовувати при прогнозуванні обчислювальних ресурсів.

2.3 Дослідження моделі прогнозування

Досліджувана модель прогнозування ресурсів заснована на використанні часових рядів. Подібні математичні моделі прагнуть знайти залежність майбутнього значення від минулого всередині самого процесу і на цій залежності обчислити прогноз. Ці моделі універсальні для різних предментних областей, тобто їх загальний вигляд не змінюється від природи часового ряду.

Перейдемо до моделювання навантаження на центральний процесор з використанням ланцюга Маркова. Модель на базі ланцюга Маркова є структурною моделлю, тому що в таких моделях залежність майбутнього значення від минулого задається у вигляді певної структури і правил переходу по ній [8–9].

Побудуємо перехідну матрицю для ланцюга Маркова. Для цього розрахуємо ймовірність переходу зі стану s_i в s_j. Ймовірність переходу зі стану s_i в s_j дорівнює кількості переходіи зі стану s_i в s_j, поділенному на загальну кількість успішних переходів зі стану s_i в будь-який інший стан. Таким чином, ймовірність перехода визначається за формулою:

$$p\left(s_i \to s_j\right) = P[X_{t+1} = s_j \,|\, X_t = s_i], \tag{2.22}$$

де $p\left(s_i \to s_j\right)$ – умовна ймовірність перехода зі стану s_i в s_j;

X_t – значення випадкової величини у дискретний момент часу t.

Використовуючи формулу 2.2, розраховуємо ймовірності переходу для усіх станів системи. З отриманих результатів будуємо матриці переходів для навантаження на центральний процесор та навантаження на оперативну пам'ять (табл. 2.4, 2.5).

Тепер можна перейти до побудови діаграми перехідних станів для використання ресурсів процесора.

Таблиця 2.4 – Матриця переходів для навантаження на центральний процесор

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1	0.333	0.667	0	0	0	0	0	0
s_2	0.263	0.316	0.316	0.105	0	0	0	0
s_3	0.083	0.333	0.25	0.167	0	0.083	0.083	0
s_4	0	0.25	0.125	0.25	0.125	0.125	0.125	0
s_5	0	0.1	0	0.1	0.6	0.1	0	0.1
s_6	0	0	0.143	0	0.429	0.286	0.143	0
s_7	0	0	0.167	0.167	0	0.167	0.333	0.167
s_8	0	0	0	0	0	0.25	0.25	0.5

Таблица 2.5 – Матриця переходів для навантаження на оперативну пам'ять

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1	0	1	0	0	0	0	0	0
s_2	0.045	0.773	0.045	0.091	0	0	0	0.045
s_3	0	0.143	0.429	0.429	0	0	0	0
s_4	0.083	0.083	0.25	0.5	0.083	0	0	0
s_5	0	0	0	0.25	0.25	0.5	0	0
s_6	0	0	0	0.2	0.2	0.4	0	0.2
s_7	0	0	0	0	0.143	0.143	0.286	0.429
s_8	0	0	0	0	0	0	0.313	0.688

Система, що описана Марківським ланцюгом, може бути представлена діаграмою перехідних станів. Діаграма перехідних станів це діаграма, яка показує усі можливості переходу системи з одного стану в інший. Можливість пере-

ходу зі стану s_i у стан s_j зображується у вигляді зв'язку між станами s_i та s_j із зазначенням спрямування у бік s_j.

Інформація зі стохастичної матриці показана у вигляді діаграми перехідних станів на рисунку 2.5.

Діаграма показує 8 станів системи та ймовірності переходу між цими станами. Важливо, щоб система була сильно пов'язана. Сильно пов'язаний стан системи забезпечує стабільність системи прогнозування, заснованої на ланцюзі Маркова.

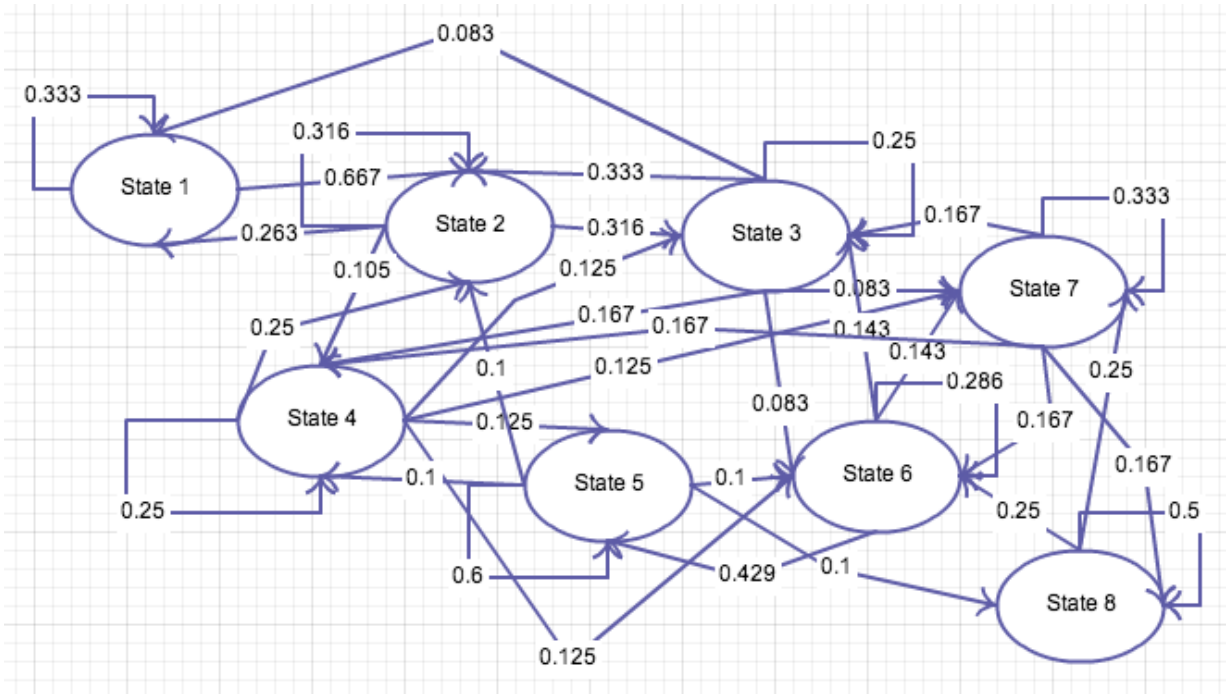


Рисунок 2.5 – Діаграма перехідних станів для навантаження на центральний процесор

2.4 Перевірка системи на стаціонарність

Стаціонарність процесу – це властивість не змінювати свої ймовірнісні характеристики з часом. На практиці стаціонарність дозволяє припустити, що для процесу характеристики будь-якої випадкової вибірки та генеральної суку-

пності збігаються.

За результатами розрахунків перехідних матриць для навантаження на центральний процесор та оперативну пам’ять можна зробити висновок, що в обох випадках Марківський ланцюг є неоднорідним, тому що перехідні ймовірності змінюються на кожному кроці, тобто:

$$p\left(s_i \to s_j\right)_{t+1} \neq p\left(s_i \to s_j\right)_t, \tag{2.23}$$

де $p\left(s_i \to s_j\right)$ – умовна ймовірність переходу зі стану s_i в s_j;

t – дискретний момент часу.

Також Марківські ланцюги не є розкладними, оскільки вони не містять безповоротних станів. Безповоротним називається стан, з якого не можна перейти ні в який інший стан, тобто:

$$p\left(s_i \to s_j\right) = 0, \quad i \neq j, \tag{2.24}$$

де $p\left(s_i \to s_j\right)$ – умовна ймовірність переходу зі стану s_i в s_j.

Водночас, Марківські ланцюги є сильно зв'язаними, оскільки умовою сильної зв'язності є можливість переходу з будь–якого стану s_i у будь-який стан s_j за кінцеве число кроків. Тобто, досліджувані ланцюги Маркова є ергодичними. Для ергодичних ланцюгів при достатньо великому часі функціонування $\left(t \to \infty\right)$ наступає стаціонарний режим, завдяки якому ймовірності p_i стану системи не залежать від часу та не залежать від розподілу ймовірностей у початковий момент часу, тобто $p_i = const$ [5].

Кожна компонента p_i вектора таких стаціонарних ймовірностей характеризує середню долю часу, протягом якого система знаходиться в аналізованому стані s_i за час спостереження.

Для визначення стаціонарних ймовірностей p_i знаходження системи у стані s_i скористаємося наступною формулою для складання системи рівнянь:

$$p_i = \sum_{j=1}^{n} p_j * p\left(s_j \to s_i\right),$$
$$\sum p_i = 1, \qquad (2.25)$$

де p_i, p_j – стаціонарні ймовірності для станів s_i та s_j відповідно;

$p\left(s_j \to s_i\right)$ – ймовірність переходу системи зі стану s_j у стан s_i;

n – кількість станів системи.

Таким чином, використовуючи формулу 2.25 для навантаження на центральний процесор, отримаємо наступну систему рівнянь:

$$\begin{cases} p_1 = 0.333p_1 + 0.263p_2 + 0.083p_3, \\ p_2 = 0.667p_1 + 0.316p_2 + 0.333p_3 + 0.25p_4 + 0.1p_5, \\ p_3 = 0.316p_2 + 0.25p_3 + 0.125p_4 + 0.143p_6 + 0.167p_7, \\ p_4 = 0.105p_2 + 0.167p_3 + 0.25p_4 + 0.1p_5 + 0.167p_7, \\ p_5 = 0.125p_4 + 0.6p_5 + 0.429p_6, \\ p_6 = 0.083p_3 + 0.125p_4 + 0.1p_5 + 0.286p_6 + 0.167p_7 + 0.25p_8, \\ p_7 = 0.083p_3 + 0.125p_4 + 0.143p_6 + 0.333p_7 + 0.25p_8, \\ p_8 = 0.1p_5 + 0.167p_7 + 0.5p_8, \\ p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 = 1, \end{cases} \qquad (2.26)$$

де p_i – стаціонарна ймовірність i–го стану системи.

Рішення системи рівнянь будемо шукати методом Гаусса. Метод Гаусса це класичний метод рішення системи лінійних рівнянь, суть якого полягає у послідовному виключенні змінних та приведення системи до трикутного виду. Аналогічним способом, для навантаження на оперативну пам'ять отримаємо наступну систему рівнянь:

$$\begin{cases} p_1 = 0.045p_2 + 0.083p_4, \\ p_2 = p_1 + 0.773p_2 + 0.143p_3 + 0.083p_4, \\ p_3 = 0.045p_2 + 0.429p_3 + 0.25p_4, \\ p_4 = 0.091p_2 + 0.429p_3 + 0.5p_4 + 0.25p_5 + 0.2p_6, \\ p_5 = 0.083_4 + 0.25p_5 + 0.2p_6 + 0.143p_7, \\ p_6 = 0.5p_5 + 0.4p_6 + 0.143p_7, \\ p_7 = 0.288p_7 + 0.313p_8, \\ p_8 = 0.045p_2 + 0.2p_6 + 0.429p_7 + 0.687p_8, \\ p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 = 1, \end{cases} \quad (2.27)$$

де p_i – стаціонарна ймовірність i –го стану системи.

Результати рішення систем рівнянь приведені у таблиці 2.6.

Таблиця 2.6 – Результати рішення систем рівнянь

Стаціонарна ймовірність	Навантаженя на центральний процесор	Навантаження на оперативну пам'ять
p_1	0.12	0.028
p_2	0.253	0.258
p_3	0.16	0.105
p_4	0.107	0.192
p_5	0.133	0.056
p_6	0.093	0.068
p_7	0.08	0.089
p_8	0.053	0.203

Розраховані ймовірності будуть використовуватися при прогнозуванні майбутнього стану системи. Чим ближче вектор розподілу ймовірностей прогнозу до стаціонаціонарних ймовірностей, тим надійніше виконаний прогноз обчислювальних ресурсів.

2.5 Прогнозування навантаження на систему

Виконаємо розрахунок вектора розподілу системи. Вектор розподілу – це вектор з одним невід'ємним значенням для кожного стану системи.

Нехай M – це стохастична матриця для ланцюга Маркова з $n=8$ кількістю станів, а V це початковий вектор розподілу. Використовуючи формули (2.4–2.5), можна визначити початковий вектор V_1. Після одного кроку розподіл змінюється на $V_2 = V_1 * M$, де M – це стохастична матриця. Таким чином, можна зробити прогноз обчислювальних ресурсів на кожний часовий крок, у даному випадку день.

Розрахуємо початковий вектор розподілу для *n* станів використовування ресурсів процесора (табл. 2.7).

Таблиця 2.7 – Результати розрахунків вектора розподілу для навантаження на центральний процесор та оперативну пам'ять

Стан системи	Навантаження на центральний процесор	Навантаження на оперативну пам'ять
v_1	0.1198	0.0273
v_2	0.2532	0.2549
v_3	0.1601	0.1034
v_4	0.1067	0.1902
v_5	0.1335	0.056
v_6	0.0933	0.0677
v_7	0.0799	0.0885
v_8	0.0534	0.2019

Отримані результати близькі до стаціонарних ймовірностей (табл 2.6). Це

говорить про високу надійність прогнозу обчислювальних ресурсів [10]. Перевіримо ступінь надійності прогнозу ресурсів.

Для цього, рохрахуємо оптімальне число днів для складання прогнозу за формулою:

$$\left| p_i - v_{i,\,t} \right| \le \varepsilon = 0.001, \tag{2.28}$$

де p_i – стаціонарна ймовірність стану s_i;

$v_{i,\,t}$ – ймовірність вектора розподілу на момент t;

ε – точність обчислень.

У результаті обчислень для навантаження на центральний процесор отримуємо $t = 23.$ Таким чином, 23 днів достатньо для складання оптимального прогнозу навантаження на центральний процесор бо за цей час система переходить у стаціонарний стан.

Для навантаження на оперативну пам'ять $t \to \infty$. Це говорить про те, що система не переходить в стаціонарний стан. У цьому випадку будемо використовувати формулу:

$$\left| v_{i,\,t-1} - v_{i,\,t} \right| \le \varepsilon = 0.001, \tag{2.29}$$

де $v_{i,\,t}$ – ймовірність вектора розподілу на момент t;

ε – точність обчислень.

Отримаємо значення $t = 31.$ Це означає, що інформації за 31 день роботи системи достатньо для складання прогнозу, однак він буде менш точним, завдяки тому, що система не перебуває у стаціонарному стані. Далі в цьому розділі буде проведений порівняльний аналіз прогнозу для процесора та прогнозу для оперативної пам'яті.

Проведемо розрахунок помилки прогнозування вектора розподілу. Для цього будемо використовувати Евклидову метрику *n*-мірного простору. Евкли-

дова метрика є геометричною відстанню між двома точками у багатовимірному просторі, який обчислюється за теоремою Піфагора [11–12].

Таким чином, помилка розраховується між даними, що були спрогнозовані, та реальними даними за наступною формулою:

$$d(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{2.30}$$

де $d(\vec{x},\vec{y})$ – помилка прогнозування;

$\vec{x}$ – спрогнозовані дані вектора розподілення навантаження на центральний процесор;

$\vec{y}$ – експерементальні дані стосовно навантаження на центральний процесор.

Результати прогнозування зображені на рисунку 2.6.

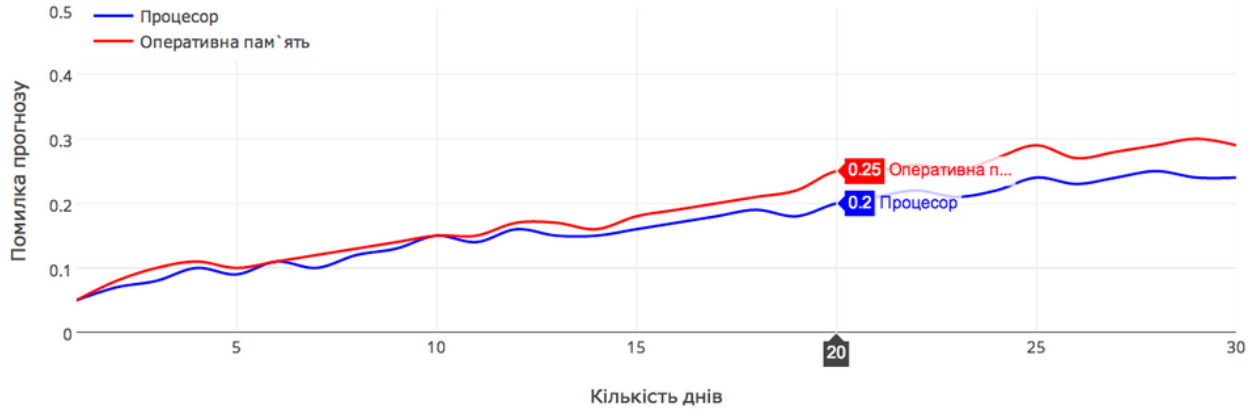


Рисунок 2.6 – Помилка прогнозу для навантаження на процессор та оперативну пам’ять

На графіку видно, що ймовірність помилки у прогнозі ресурсів через 20 днів становить 20% для навантаження на центральний процесор та 25% для навантаження на оперативну пам’ять. Прогноз ресурсів центрального процесора надійніший завдяки стаціонарності стану [13–16].

Також, можна помітити, що значення помилки прогнозування швидше йде на підвищення в перших днях прогнозу, а потім показник стабілізується і

практично не змінюється з часом завдяки стаціонарності стану.

2.6 Алгоритм MapReduce

Мобільна система моніторингу працює з великим об'ємом даних, що впливає на швидкість обробки інформації. Для того, щоб сервіс встигав обробляти велику кількість лог-файлів у реальному часі, необхідно використовувати ефективний алгоритм розподілених обчислень. Для таких цілей підходить алгоритм MapReduce.

Основний виграш у швидкості роботи даного алгоритму полягає у тому, що з’являється можливість паралельної обробки даних на декількох комп'ютерах одночасно. Мобільна система моніторингу досить об'ємна, тому, має сенс розділити роботу сервісу на кілька комп'ютерів, з метою підвищення швидкості роботи системи.

Алгоритм MapReduce – це модель розподілених обчислень, яка використовується для паралельних обчислень над великими обсягами даних. Цей алгоритм дозволяє підготувати та структурувати дані перед інтелектуальною обробкою. Наприклад, щоб зібрати інформацію щодо роботи системи за місяць, алгоритм MapReduce дозволяє структурувати інформацію за кожен день, та представити її у зручному вигляді для подальшої обробки. Розглянемо принцип роботи цього алгоритму.

Робота алгоритма MapReduce складається з двух кроків: Map та Reduce.

На Map кроці відбувається попередня обробка лог-файлів. Для цього, головний комп'ютер отримує вхідні дані від агента, потім розділяє їх на частини та передає іншим комп'ютерам задля попередньої обробки. Операція Map встановлює пари «ключ–значення», після чого ці пари передаються функції Reduce у сгрупованому по ключу вигляді.

На Reduce кроці, відбувається згортка попередньо оброблених даних. Головний вузел отримує відповіді від робочих вузлів та на їх основі формує ре-

зультат – рішення поставленої задачі. Операція Reduce формує одну пару «ключ–значення».

Перевага алгоритму полягає в тому, що він дозволяє паралельно проводити операції попередньої обробки та згортки за рахунок того, що результат роботи попередньої обробки та згортки має однаковий формат. Таким чином, згортка може виконуватись багаторазово [17]. У зв'язку з цим, на крок Reduce накладаються наступні вимоги:

– формат значення повинен збігатися із форматом значення що поступає на вхід функції Map;

– повинно виконуватися рівняння

$$r\left(k,\left[s_1,r\left(k,\left[s_2,s_3\right]\right)\right]\right)=r\left(k,\left[s_1,s_2,s_3\right]\right), \tag{2.31}$$

де r – функція Reduce;

k – ключ пари;

s_1, s_2, s_3 – стани системи.

Це рівняння говорить про те, що:

– порядок обробки та кількість переданих значень не повинні впливати на результат;

– повторне використання шагу Reduce не повинно впливати на результат.

Також алгоритм має високий рівень відмовостійкості. Відмовостійкість досягається шляхом перезапуску в інших вузлах задач, які виконувалися на вузлах, де відбулися відмови.

3 АНАЛІЗ ФУНКЦІЙ МОБІЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ

3.1 Аналіз функцій аналогічних систем

Перед розробкою системи моніторингу був проведений аналіз функцій аналогічних систем. Аналіз функцій аналогічних систем дозволить зрозуміти, які функції необхідно реалізувати в системі, щоб вона була конкурентоспроможною (табл. 3.1).

Таблиця 3.1 – Аналіз функцій аналогічних систем

Функція	Розроблювана система	Zabbix	OpenNMS
Побудова діаграм	+	+	+
Прогнозування подій	+	+	–
Агент	+	+	+
SNMP	+	+	+
syslog	+	+	–
WMI	+	–	–
NetFlow	+	–	–
Аналіз трафіка	+	–	–
Тригери, тривоги	+	+	+
Плагіни	–	+	+
Доступ через Web	+	+	+
Управління доступом	+	+	–
Мобільний додаток	+	+	+

Наведений перелік функцій допоможе визначити, яким чином побудувати архітектуру мобільної системи моніторингу. Кожна функція визначається рівнем важливості, складності та ризику. Це дозволить визначити пріоритет під час програмної реалізації системи.

3.2 Функції системи

Після проведення повного аналізу можливих функцій системи, було складено перелік усіх функцій системи моніторингу, що входять в рамки дослідження. Важливість, складність та ризик визначаються за шкалою від 1 до 5, де 5 означає найвищий рівень, а 1 – найнижчий. Цей перелік наведено у таблиці 3.2.

Таблиця 3.2 – Функції системи та їх атрибути

Функція	Опис	Важливість	Складність	Ризик
1	2	3	4	5
Побудова діаграм	Можливість перегляду результатів прогнозування у вигляді діаграм	5	4	2
Прогнозування подій	Можливість прогнозування подій за результатами аналізу зібраних даних	5	5	5
Агент	Наявність ПЗ, що дозволяє отримувати дані щодо обчислювальних ресурсів	5	5	5
SNMP	Підтримка протоколу SNMP	3	4	2
syslog	Підтримка стандарту syslog	3	3	1
WMI	Підтримка технології WMI	3	3	1

Кінець таблиці 3.2

1	2	3	4	5
NetFlow	Підтримка протоколу NetFlow	3	3	1
Аналіз трафіка	Можливість аналізу мережевого трафіку	5	5	3
Тригери, тривоги	Можливість виклику тривоги у разі наявності проблем	4	3	3
Доступ через Web	Можливість роботи із системою через Web	4	4	3
Управління доступом	Наявність системи акаунтів для розмежування доступу	4	4	5
Мобільний додаток	Можливість роботи із системою через мобільний додаток на базі ОС Android	5	4	3

Виходячи з даних у таблиці 3.2, можна зробити висновок, що найбільш важливими функціями є прогнозування подій, побудова діаграм та реалізація програм для роботи з даними (агент, мобільний додаток). Решта функцій системи, такі як підтримка різних протоколів, є додатковою функціональністю і мають нижчий пріоритет.

4 АРХІТЕКТУРА РОЗРОБЛЮВАНОЇ СИСТЕМИ

У системі реалізовані дві ролі: звичайний користувач та адміністратор. Звичайний користувач має право авторизуватися у системі та переглядати стан віртуальних машин до яких він має доступ. Адміністратор системи має ті ж права що і звичайний користувач, а також можливість зміни конфігурації сервісу та налаштування доступу до системи для звичайних користувачів.

У розроблювану систему входять:

– мобільний додаток на платформі Android. Мобільний додаток дозволяє користувачеві переглядати дані щодо роботи системи за обраний проміжок часу. Користувач може побачити яке було споживання обчислювальних ресурсів (центральний процесор, оперативна пам'ять, жорсткий диск, мережевий трафік) у заданий момент часу. Також, користувач може переглядати результати аналізу даних у вигляді діаграм, графіків і таблиць. Якщо користувач є адміністратором, то він також має можливість конфігурування налаштувань сервісу, з метою оптимізації роботи системи;

– вебсайт на платформі J2EE. Вебсайт призначений для тих же цілей, що й мобільний додаток. У відмінності від мобільного додатку, має кілька додаткових функцій, таких як конфігурування сервісу через XML файли, а також управління системою акаунтів;

– агент системи моніторингу. Ця частина системи є додатком, що запускає демон–процес, що працює у фоновому режимі без прямої взаємодії з користувачем. Попередньо налаштований агент збирає інформацію щодо роботи віртуальної машини у встановлені адміністратором моменти часу;

– хмарний сервіс для обробки даних. Є основним компонентом системи. Структурує та аналізує дані, які надходять від агента після чого виконує прогноз обчислювальних ресурсів. Після завершення процесу прогнозування повідомляє цю інформацію іншим компонентам системи. Містить перелік налаштувань, які дозволяють налаштувати роботу системи необхідним чином.

4.1 Архітектура сервісу у приватній хмарі

Сервіс системи призначений для обробки отриманих агентом даних та прогнозування ресурсів. Найбільш важливими властивостями такого сервісу є гнучкість та масштабованість. Розглянемо функції сервісу більш детально на діаграмі варіантів використання (рис. 4.1) [18].

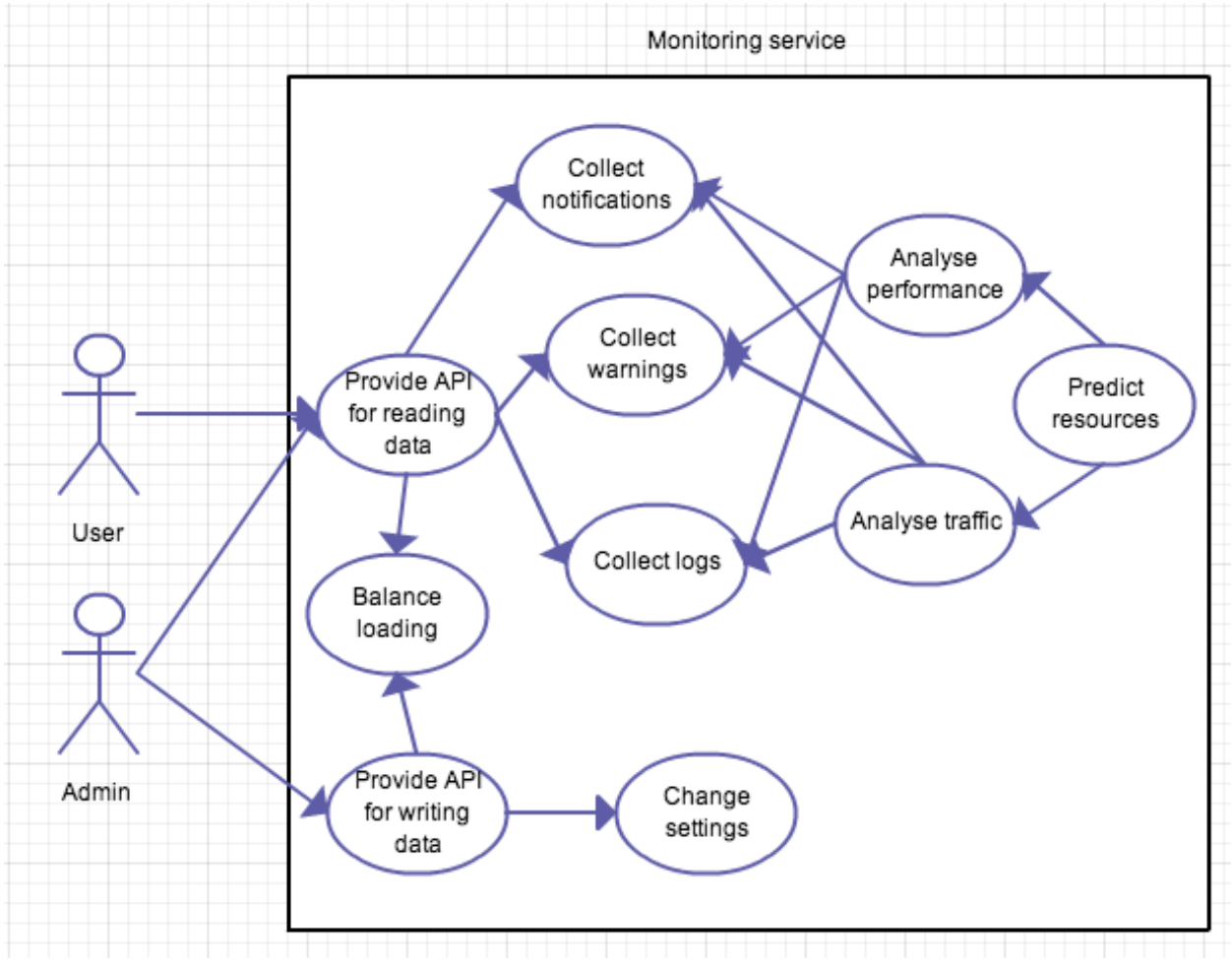


Рисунок 4.1 – Діаграма варіантів використання сервісу

На сервері знаходиться база даних користувачів, яка керується адміністратором системи через вебсайт. Адміністратор може переглядати, додавати, редагувати та видаляти користувачів системи. Також адміністратор може настроювати рівні доступу кожному користувачеві системи.

Перед звертанням до сервісу користувач повинен увійти в систему, вико-

ристовуючи свій аккаунт. Процес авторизації виконується за допомогою протоколу OAuth 2.0. Цей протокол дозволяє отримати доступ до ресурсів без необхідності передавати логін і пароль. Після проходження процесу авторизації в систему користувач отримує ключ доступу до системи, який дозволить йому використовувати API функції сервісу. Час роботи ключа доступу встановлюється адміністратором в налаштуваннях сервісу.

У цілях безпеки процес авторизації використовує протокол SSL. Цей криптографічний протокол забезпечує безпеку зв'язку тим, що він використовує асиметричну криптографію для аутентифікації ключів обміну, симетричне шифрування для збереження конфіденційності та коди аутентифікації повідомлень для цілісності повідомлень.

API функції сервісу розділені на два типи: функції читання та функції запису даних. До функцій читання належать функції збору лог-файлів, нагадувань, тривог та попереджень, функції збору інформації щодо результатів аналізу, а також прогнозування ресурсів. До функцій запису відносяться функції, що дозволяють адміністратору змінювати конфігурацію сервісу, наприклад, налаштування віртуальних машин, налаштування процесу збору та аналізу даних тощо.

Для підвищення ефективності роботи сервісу використовується балансування навантаження за допомогою алгоритму Round Robin, тим самим знижуючи навантаження на пристрій користувача. Цей алгоритм гарантовано забезпечує системну стійкість, з огляду на те, що при використанні цього алгоритму неможлива ситуація перевантаження окремих вузлів системи. Алгоритм Round Robin є одним з найбільш простих та ефективних алгоритмів балансування завдяки тому, що він забезпечує рівномірний розподіл запитів по серверам. Цей алгоритм дуже широко використовується при складанні розкладів у багатозадачних системах з поділом часу [19].

Суть алгоритму полягає у тому, що всі виконувані процеси поміщаються в одну циклічну чергу. У процесі роботи планувальник процесора по черзі виділяє кожному процесу квант часу. В системі всі процеси мають однаковий

пріоритет, тому у всіх завдань однаковий квант часу [20].

Сервіс містить однозначно визначений інтерфейс для роботи з хмарою. Через цей інтерфейс мобільний додаток може взаємодіяти з хмарою, запитувати інформацію та керувати її ресурсами. Усі запити на сервер та відповіді з сервера мають формат JSON, тому що цей формат вважається найбільш зручним для передачі подібної інформації за рахунок того, що вимагає значно менше пам'яті порівняно з XML. Також цей формат більш зручний при обробці, бо існують бібліотеки, такі як Google gson, що забезпечують максимально швидку обробку даних у цьому форматі.

Дані щодо роботи системи у форматі JSON надходять від агента, який збирає інформацію з віртуальних машин. Для зберігання даних у системі використовується NoSQL система управління базами даних MongoDB. Ця система є документо-орієнтованою та має JSON-подібну схему зберігання даних. Таким чином, процес збереження даних полегшується завдяки тому, що дані, що надійшли від агента, вже мають необхідний формат. Також, важливою перевагою MongoDB є можливість роботи у зв'язку з парадигмою MapReduce.

Для обробки інформації потрібна велика кількість серверних машин, тому що кількість даних дуже велика. Перед обробкою необхідно попередньо структурувати інформацію з метою зниження навантаження на обчислення.

Для цього перед інтелектуальною обробкою даних сервіс використовує алгоритм MapReduce для структурування інформації.

Завдяки зручності, реалізація алгоритму виконана на стороні MongoDB з використанням мови JavaScript. На кроці Map алгоритм за допомогою функції emit створює пари «ключ–значення», що дозволяє створити зручну структуру.

Потім пари «ключ–значення» передаються на Reduce крок, на якому алгоритм виконує згортку даних в одну пару «ключ–значення». Це дозволяє зменшити число даних для зручності подальшої обробки.

Після виконання Reduce кроку на виході виходить пара «ключ–значення», яка містить сумарне навантаження на обчислювальні ресурси системи, а також число спостережень. Таким чином, можна виконати фінальне обчислення.

Після виконання останнього кроку запускається паралельне обчислення, яке дозволяє структурувати дані та підготувати їх перед інтелектуальною обробкою.

Програмна реалізація прогнозування обчислювальних ресурсів виконана з використанням мови Java. Після обробки та структурування даних, отриманих від агента, сервіс звертається до бази даних за необхідною інформацією. При зверненні до бази вказуються дані, які необхідно проаналізувати сервісу, а також проміжок часу, за який необхідно виконувати аналіз.

Починається прогнозування з побудови матриці переходів, використовуючи отримані від агента дані. Дані надходять з бази у форматі JSON, після чого циклічно обробляються.

Після заверешення процесу побудови перехідної матриці система знає перехідні ймовірності станів, які їй знадобляться для здійснення прогнозу обчислювальних ресурсів. Виконавши цей крок, вона переходить до прогнозування обчислювальних ресурсів.

Процес прогнозування полягає в обчисленні вектора розподілу ймовірностей. На вхід подається початковий вектор розподілу, після чого він множиться на стохастичну перехідну матрицю необхідну кількість разів. На виході отримуємо вектор розподілу ймовірностей, близький до стаціонарних ймовірностей, який описує ймовірності переходу системи у кожному зі станів.

Використовуючи отриманий вектор розподілу ймовірностей, можна прогнозувати навантаження на центральний процесор, оперативну пам'ять, проводити аналіз мережевого трафіку та приймати рішення щодо збільшення або зменшення числа віртуальних машин, зміни конфігурації сервісу тощо.

Для аналізу мережевого трафіку сервіс використовує SNMP та HTTP колектори, які збирають інформацію про події, що відбулися. Після аналізу цих даних сервіс відправляє результати на мобільний додаток та вебсайт.

Всі складні обчислення виконуються тільки на стороні сервісу, що надає додаткову гнучкість та масштабованість системі. Таким чином, сервіс надає обчислювальні потужності для мобільного додатку, знижуючи навантаження на

пристрій користувача, та, тим самим, підвищуючи швидкість роботи системи. Користувачам надається повний контроль над обчислювальними ресурсами, а також доступна середа для роботи.

У разі підвищеного навантаження на систему сервіс може сповіщати власника через мобільний додаток та вебсайт залежно від конфігурації сервісу. Сервіс містить колектор оповіщень, в який поміщаються повідомлення під час роботи системи. Потім, залежно від налаштувань сервісу, необхідні оповіщення відправляються у вигляді Push-запитів клієнта на пристрій. Технологія Push стала дуже популярною на сьогоднішній день та є дуже зручним засобом для "просування" оновлення або повідомлення на пристрій користувача. Ця технологія ідеально підходить для розсилки повідомлень та розповсюдження контенту.

Також сервіс підтримує можливість відправки e-mail або SMS- повідомлення з важливою інформацією для клієнта, наприклад, інформацію про надмірну перевантаженість віртуальних машин, інформацію про збої або мережеві атаки, інформацію про проблемний сервер тощо.

4.2 Архітектура агента

Агент системи призначений для збору інформації щодо використання фізичних та віртуальних ресурсів. Розглянемо функції агента більш детально на діаграмі варіантів використання (рис 4.2).

Агент встановлюється адміністратором системи на фізичну або віртуальну машину для збору інформації щодо використання ресурсів. Цей додаток збирає інформацію про навантаження процесора, пам'яті та мережевого трафіку та зберігає її у лог-файли. Збір інформації здійснюється завдяки демон-процесу, який працює у фоновому режимі без участі користувача. Вихідний код програми агента написаний на мові C. Програма є кросплатформенною та може працювати в таких системах як Unix, Linux, Mac OS X та Microsoft Windows.

Після читання інформації ці дані зберігаються у базу даних. На кожному

часовому інтервалі проводиться обчислення середнього навантаження на процесор, після чого дані зберігаються в базу.

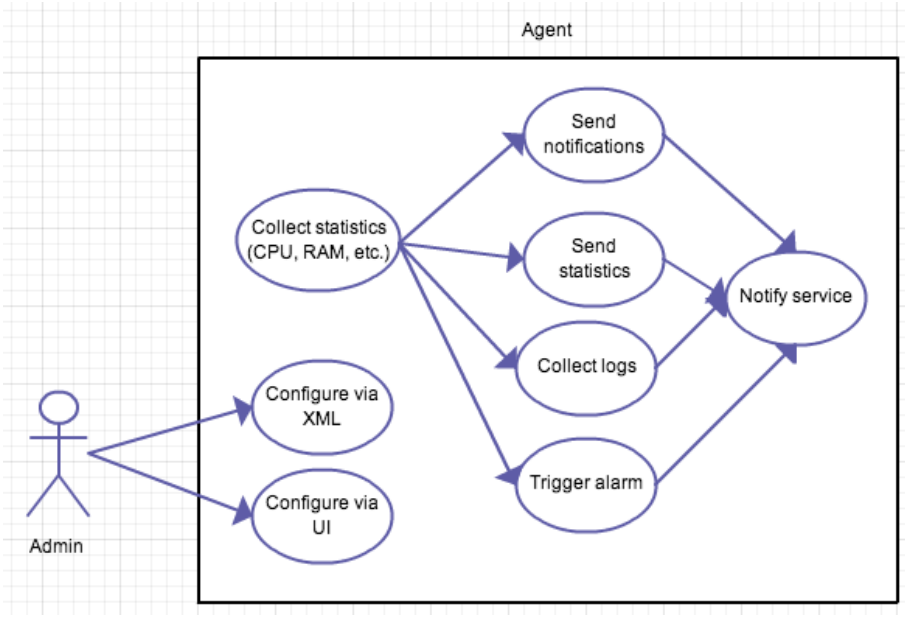


Рисунок 4.2 – Діаграма варіантів використання агента

Під час збору інформації агент зберігає дані у вигляді JSON файлів, після чого відправляє цю інформацію в базу даних, використовуючи API сервісу. Відправка даних вимагає наявність ключа доступу до API сервісу, тому агент вимагає авторизації з боку користувача для подальшого використання програми.

При установці агента на машину адміністратор налаштовує програму для оптимальної роботи за допомогою візуального інтерфейсу або XML файлів. Адміністратор може конфігурувати: інтервал часу, через який агент буде збирати дані, налаштування системи сповіщень, а також налаштування автозапуску.

У систему сповіщень входять різні види повідомлень, тривоги, попередження про збої, нагадування, інформація про неполадки, інформація про атаки. Всі оповіщення мають один інтерфейс, що полегшує їх обробку, яка виконується сервісом. Після обробки цих оповіщень сервіс надсилає Push повідомлення до мобільного додатку.

4.3 Архітектура мобільного додатку

Мобільний додаток на базі ОС Android дозволяє користувачеві переглядати інформацію щодо використання ресурсів системи [21]. Для отримання даних щодо використання ресурсів додатку необхідно зробити запит до API функцій сервісу в узгодженому форматі. Перед відправленням запиту до API функцій необхідно авторизуватися, використовуючи протокол OAuth 2.0.

Після відправлення запиту на мобільний пристрій приходить відповідь від сервісу з набором даних у форматі JSON. На мобільному пристрої дані зберігаються у локальну базу даних SQLite. Потім ці дані відображаються у вигляді таблиць, діаграм та графіків (рис. 4.3).

Мобільний додаток дозволяє адміністратору міняти конфігурацію сервісу. Під регулюванням налаштувань сервісу мається на увазі можливість збільшення або зменшення кількості віртуальних машин, управління балансуванням навантаження, можливість аналізування навантаження системи, налаштування критеріїв аналізу, а також прогнозування ресурсів хмарної системи.

Розглянемо функції мобільного додатку більш детально на діаграмі варіантів використання (рис. 4.4).

Мобільний додаток містить свою систему оповіщень. До неї відносятся різні види повідомлень, тривоги, попередження про збої, нагадування, інформація про неполадки, інформація про атаки. Залежно від отриманої інформації сервіс пропонує користувачеві змінити ті чи інші конфігурації, щоб оптимізувати роботу системи та уникнути подальших проблем, наприклад, встановити захист від мережевих атак та збоїв. Також ці настройки можуть змінюватись користувачем вручну.

Більшість оповіщень приходить на мобільний пристрій у вигляді Push повідомлень. Це дозволяє швидко та зручно доставити повідомлення користувачу на пристрій. Другим можливим варіантом реалізації оповіщень може бути технологія «опитування» polling, яка полягає у тому, що мобільний додаток сам відправляє запити сервісу в очікуванні нових даних. Однак, це призвело б до під-

вищення навантаження на мережевий трафік та різкого збільшення споживання енергії акумулятора, що безумовно погано позначається на часі роботи пристрою.

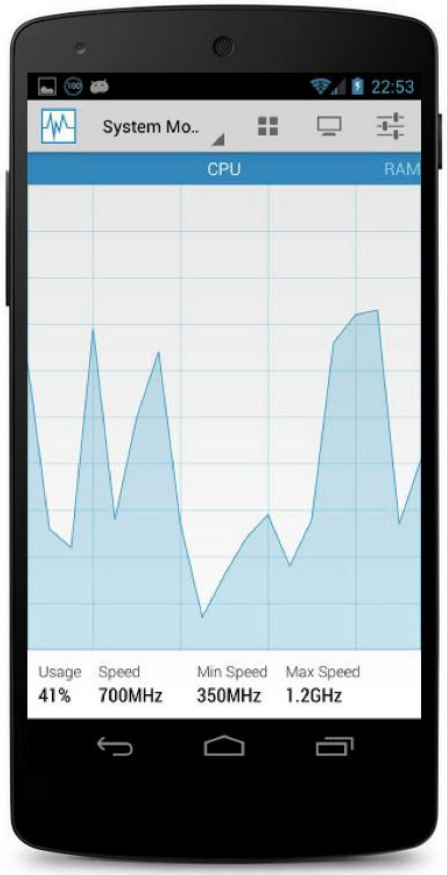


Рисунок 4.3 – Перегляд інформації щодо споживання обчислювальних ресурсів системи

Ще однією перевагою мобільного пристрою є те, що він може приймати оповіщення у вигляді SMS та email. Це дозволяє тримати користувача в курсі найважливіших подій, що відбуваються у його приватній хмарі.

4.4 Вебсайт системи моніторингу

Вебсайт фактично дублює функціональність мобільного додатку та є доповненням до системи з метою підвищення зручності роботи, тому діаграма ва-

ріантів використання майже ідентична діаграмі для мобільного пристрою. Використовуючи створену систему акаунтів, він дозволяє переглядати інформацію та конфігурувати роботу сервісу.

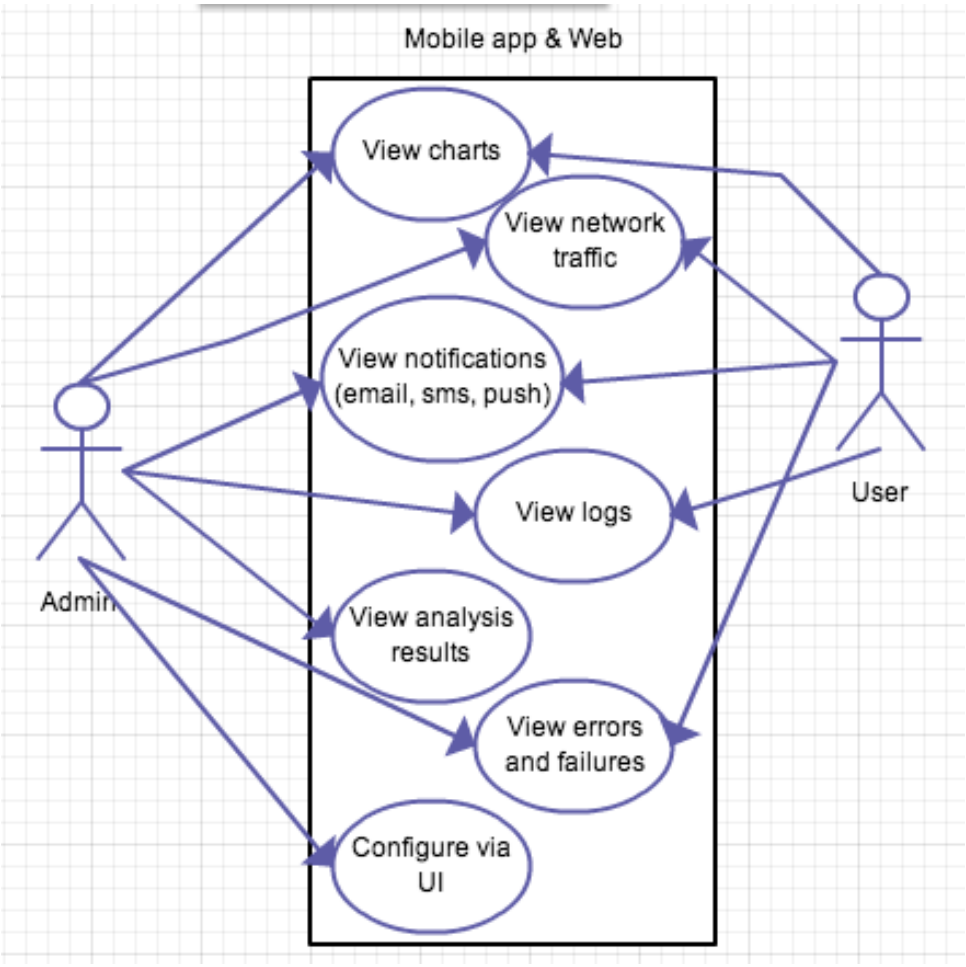
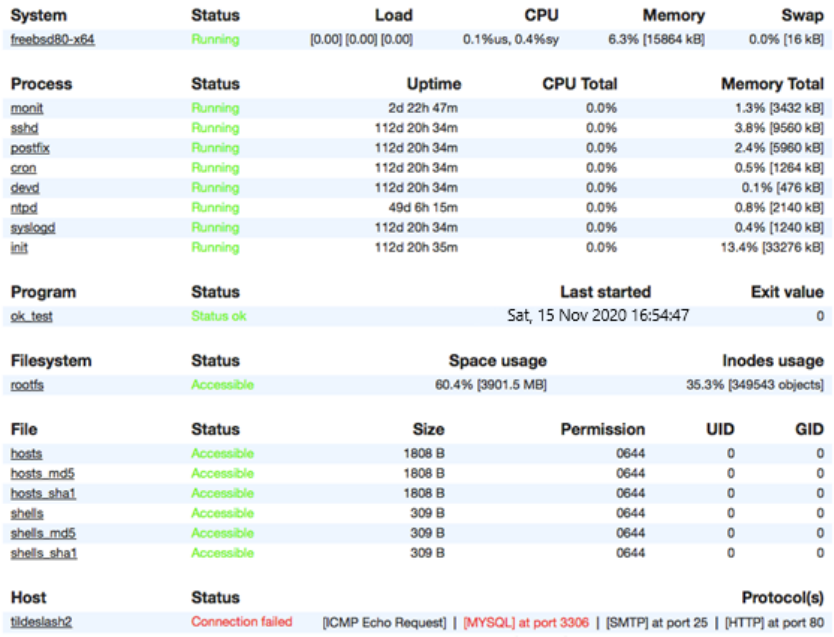


Рисунок 4.4 – Діаграма варіантів використання мобільного додатку

На відміну від мобільного додатку вебсайт надає можливість конфігурування системи акаунтів, а також можливість налаштування роботи сервісу через XML файли.

Після проходження процесу авторизації вебсайт дозволяє переглядати інформацію щодо результатів аналізу даних (рис. 4.5).

Система оповіщень на вебсайті працює аналогічним чином, що і в мобільному додатку, з тією лише різницею, що у користувача з'являється можливість скласти звіт про роботу системи.



System	Status	Load	CPU	Memory	Swap
freebsd80-x64	Running	[0.00] [0.00] [0.00]	0.1%us, 0.4%sy	6.3% [15864 kB]	0.0% [16 kB]

Process	Status	Uptime	CPU Total	Memory Total
monit	Running	2d 22h 47m	0.0%	1.3% [3432 kB]
sshd	Running	112d 20h 34m	0.0%	3.8% [9560 kB]
postfix	Running	112d 20h 34m	0.0%	2.4% [5960 kB]
cron	Running	112d 20h 34m	0.0%	0.5% [1264 kB]
devd	Running	112d 20h 34m	0.0%	0.1% [476 kB]
ntpd	Running	49d 6h 15m	0.0%	0.8% [2140 kB]
syslogd	Running	112d 20h 34m	0.0%	0.4% [1240 kB]
init	Running	112d 20h 35m	0.0%	13.4% [33276 kB]

Program	Status	Last started	Exit value
ok_test	Status ok	Sat, 15 Nov 2020 16:54:47	0

Filesystem	Status	Space usage	Inodes usage
rootfs	Accessible	60.4% [3901.5 MB]	35.3% [349543 objects]

File	Status	Size	Permission	UID	GID
hosts	Accessible	1808 B	0644	0	0
hosts_md5	Accessible	1808 B	0644	0	0
hosts_sha1	Accessible	1808 B	0644	0	0
shells	Accessible	309 B	0644	0	0
shells_md5	Accessible	309 B	0644	0	0
shells_sha1	Accessible	309 B	0644	0	0

Host	Status	Protocol(s)
tildeslash2	Connection failed	[ICMP Echo Request] | [MYSQL] at port 3306 | [SMTP] at port 25 | [HTTP] at port 80

Рисунок 4.5 – Перегляд інформації про роботу системи

4.5 Тестування працездатності системи

У процесі реалізації системи було проведено тестування її працездатності.

Розглянемо систему з точки зору покрокового використання. Для цього повністю опишемо процес, як користувач отримує інформацію на екрані свого пристрою.

Починається процес з того, що користувач підключається до одного з серверів системи. Після того, як користувач увійшов у систему, додаток отримує інформацію з бази даних станів про те, до якого з серверів необхідно підключати клієнта. Так забезпечується балансування навантаження алгоритмом Round Robin.

Потім користувач авторизується у клієнтському додатку. Для цього він використовує свій акаунт, що створив для нього адміністратор. Після завершення процесу авторизації користувач отримує ключ доступу, який дозволяє йому використовувати API функції сервісу для прогнозування

обчислювальних ресурсів.

Водночас агент збирає інформацію щодо використання ресурсів віртуальної машини та зберігає цю інформацію в базу даних, використовуючи API функції сервісу.

Коли клієнт звертається до сервісу за вимогою зробити прогноз обчислювальних ресурсів, сервіс бере дані з бази та виконує аналіз і прогнозування ресурсів згідно моделі.

Проведемо аналіз якості прогнозування обчислювальних ресурсів системи. Для цього проведемо тестування процесу роботи системи без використання прогнозу ресурсів, а також тестування процесу роботи з використанням прогнозу та адаптацією системи залежно від результатів прогнозу.

Створимо тестовий процес обробки зображень, який запускається на віртуальних машинах системи кожен день у певний час. Кожен день системі необхідно обробити велику кількість зображень, причому число зображень заздалегідь невідомо. Для наочності тестування побудуємо процес так, що ближче до кінця місяця кількість зображень збільшується, таким чином віртуальним машинам необхідно виконувати більше роботи .

Для теста без використання прогнозування ресурсів кількість віртуальних машин фіксована та дорівнює 10. Для теста, що використовує модель прогнозування, початкове число віртуальних машин також дорівнює 10, але воно може збільшуватися або зменшуватися в залежності від результатів прогнозу навантаження. Наприклад, якщо згідно з прогнозом навантаження зросте, то необхідно збільшити число віртуальних машин та віддати частину роботи на нову машину, щоб знизити навантаження на вже працюючі машини.

На рисунку 4.6 наведено результат тестування для навантаження на центральний процесор.

На графіку видно, що на 20 день роботи системи навантаження на центральний процесор складає 18% для системи, що використовує прогнозування ресурсів, та 35% для системи, що не використовує прогнозування обчис-

лювальних ресурсів. Таким чином, навантаження відрізняється майже в 2 рази.

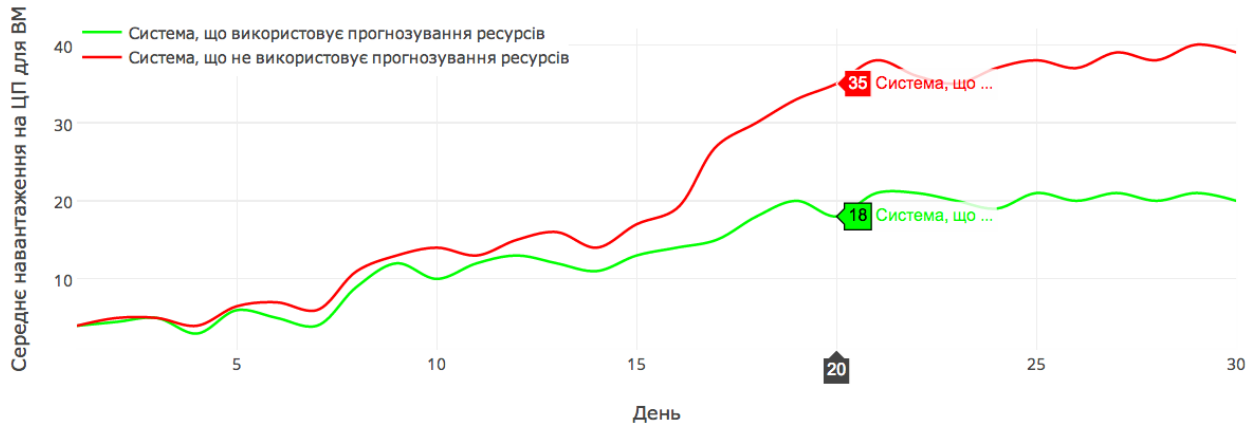


Рисунок 4.6 – Результат тестування роботи системи

Виходячи з результатів тестування можна зробити висновок про те, що прогнозування обчислювальних ресурсів для мобільної системи моніторингу є критично важливим, тому що воно дозволяє істотно оптимізувати роботу всієї системи.

Клієнтський додаток вимагає наявність стабільного підключення до мережі Інтернет. Також клієнтський додаток вимагає пристрій (телефон, планшет) з тактовою частотою процесора не менше 1 GHz, оперативною пам'яттю не менше 512 Mb та кількістю пам'яті на жорсткому диску не менше 10 Mb. На пристрої повинна бути встановлена операційна система Android 7.0 або більш нова її версія.

Сервіс повинен забезпечувати нормальну роботу для мобільних пристроїв, які мають з'єднання з мережею Інтернет зі швидкістю від 1 Мб/с. Сервіс не залежить від операційної системи, тому він може бути встановлений на будь-яку платформу. Це робить систему гнучкою та масштабованою.

Сервіс вимагає сервер з тактовою частотою процесора не менше 3 GHz, оперативною пам'яттю не менше 16 Gb та жорстким диском не менше 1 Tb. На сервері має бути встановлено таке програмне забезпечення: JDK 8u9, Tomcat 8.0, MongoDB 4.4.1 [22 – 26].

ВИСНОВКИ

У роботі була розглянута система мобільного моніторінгу з точки зору математичної статистики. Була розроблена математична модель прогнозування на базі ланцюга Маркова, розраховані та проаналізовані усі необхідні ймовірнісні характеристики такої системи.

За результатами прогнозування обчислювальних ресурсів адміністратор системи, що має доступ до віртуальних машин, зможе прийняти рішення на основі цих даних. Передбачений автоматичний режим роботи системи, в якому система сама приймає рішення щодо оптимізації налаштувань.

Після побудови моделі системи та проведення всіх необхідних розрахунків проведена програмна реалізація системи. Розроблений клієнтський додаток на платформі Android, сервіс, що надає інтерфейс для регулювання стану приватної хмари, агент, що збирає інформацію з віртуальних машин, а також вебсайт для просмотру результатів аналізу та прогнозування. Розроблені алгоритми реалізовані у хмарному сервісі. Також розробка додатків значно підвищила навички програмування.

Відзначимо, що система аналізує дані за принципом чорний ящик, тим самим не знаючи деталей щодо потужності апаратних засобів комп'ютера. Це робить модель незалежною від апаратних засобів та їх виробника.

Для перевірки результатів роботи системи розроблені програми були протестовані. Тестування було проведено з точки зору перевірки системи, а також з точки зору покрокового використання. За результатами тестування визначено, що розроблені алгоритми та методи мають критичну важливість для систем моніторингу, тому що вони значно покращують швидкість роботи системи.

Основною перевагою системи є можливість прогнозування обчислювальних ресурсів для віртуальних машин. Це дозволяє провайдеру хмарної інфраструктури адаптувати віртуальні ресурси за вимогами. Таким

чином, модель має величезний вплив на вартість та ефективність використання обчислювальних ресурсів.

Розроблені алгоритми мають високу цінність для систем, які використовують архітектуру SOA та потребують оптимізації часу обробки запитів. Подальша робота над цими алгоритмами має високу актуальність, тому що SOA архітектура набуває все більшої популярності.

Система може використовуватися для моніторингу IT інфраструктури, обробки різних типів мережевих повідомлень та подій, прогнозу цих подій, моделювання продуктивності системи, стохастичного моделювання та комп'ютерної безпеки (виявлення вторгнень та запобігання цим вторгненням).

Така система може використовуватися у різних сферах, де виникає потреба керувати великою кількістю віртуальних машин для виконання великої кількості складних обчислень. Наприклад, у розподіленій системі, що виконує обробку зображень.

Дана система також може використовуватися у хмарах Amazon, Azure та Oracle завдяки її гнучкості та масштабованості.

У якості подальшої роботи над системою планується досліджувати інші моделі прогнозування та провести їх тестування. Також планується реалізувати систему для операційної системи iOS.

ПЕРЕЛІК ПОСИЛАНЬ

1. Erl T. Service–Oriented Architecture. Concepts, technology and design. Boston : Pearson Education, 2009. 727 p.

2. Krafzig D., Banke K., Slama D. Enterprise SOA: Service–Oriented Architecture Best Practices. USA : Prentice Hall, 2004. 408 p.

3. Гмурман В. Е. Теория вероятностей и математическая статистика. Москва : Высш.шк., 2003. 479 с.

4. Боровков А. А. Математическая статистика. Новосибирск : Наука, 1997. 772 с.

5. Боровков А. А. Математическая статистика: оценка параметров, проверка гипотез. Москва : Физматлит, 1984. 472 с.

6. Пугачев В. С. Теория вероятностей и математическая статистика. Москва : Физматлит, 2002. 496 с.

7. Кельберт М. Я., Сухов Ю. М. Вероятность и статистика в примерах и задачах. Том 2. Марковские цепи как отправная точка теории случайных процессов и их приложения. Москва : Издательство МЦНМО, 2009. – 533 с.

8. Whittaker J., Thomason J. A. A Markov Chain Model for Statistical Software Testing. // IEEE Transactions on Software Engineering, 1994. Vol. 20, Issue 10. Pp. 812–824.

9. Кемени Дж., Снелл Дж. Конечные цепи Маркова. Москва : Наука, 1970. 272 с.

10. Портенко Н. И., Скороход А. В., Шуренков В. М. Марковские процессы. Москва : ВИНИТИ, 1989. 248 с.

11. Майн Х., Осаки Х. Марковские процессы принятия решений. Москва : Наука, 1977. 178 с.

12. Стратонович Р. Условные Марковские процессы и их применение к теории оптимального управления. Москва : Издательство МГУ, 1965. 321 с.

13. Pinnow A., Osterburg S. A capacity supply model for virtualized servers // Informatica Economica, 2009. Vol. 13. Pp. 96–105.

14. Тихонов Э. Е. Прогнозирование в условиях рынка. Невинномысск : Логос, 2006. 221 с.

15. Armstrong J. S., Brodie R. J. Forecasting for Marketing // Quantitative Methods in Marketing. 1999. Pp. 92 – 119.

16. Jingfei Y. M. Power System Short–term Load Forecasting . Darmstadt : Elektrotechnik und Informationstechnik der Technischen Universitat, 2006. 139 p.

17. Барсегян А., Куприянов М., Степаненко В. Методы и модели анализа данных: OLAP и Data Mining. Санкт-Петербург : БХВ, 2004. 235 с.

18. Фаулер М., Скотт К. UML. Основы. Киев : Символ–Плюс, 2002. 192 с.

19. Вишневский В. М. Теоретические основы проектирования компьютерных сетей. Москва : Техносфера, 2003. 512 с.

20. Христанков А. С. GRID-Технологии. Модели и алгоритмы распределения нагрузки. Алгоритмы на основе сетей СМО // Информационные технологии и вычислительные системы. 2009. №3. С. 48-57.

21. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. Санкт-Петербург : Питер, 2017. 736 с.

22. Гамма З., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно–ориентированного проектирования. Паттерны проектирования. Санкт-Петербург : Питер, 2001. 368 с.

23. Шилдт Г. Полный справочник по Java. 7–е издание. Москва : Издательский дом «Вильямс», 2007. 1024 с.

24. Fitzgerald M. Introducing regular expressions. New York : Springer–Verlag, 2012. 136 p.

25. Chacon S. Pro Git. New York : Springer–Verlag, 2009. 265 p.

26. Murphy M. L. The busy coder’s guide to advanced android development. USA, 2012. 792 p.

27. Кононюк А.Е. Основы теории облачных технологий. Киев : «Освіта України», 2018. 710 с.

28. Бондар Є. С., Глибовець М. М., Гороховський С. С. Хмарні обчислення та їх застосування // Вісник КНУ ім. Т. Шевченка. 2011. Вип. № 1. С. 74-82.

29. Федоров А. Г. Windows Azure: облачная платформа Microsoft. URL : http://kak.znate.ru/docs/index-61012.html (дата звернення: 28.10.2020).

30. The NIST Definition of Cloud Computing. URL : http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (дата звернення: 28.10.2020).

31. Шаньгин В. Ф. Информационная безопасность компьютерных систем и сетей. Москва : ФОРУМ, 2013. 416 с.

32. Хмарні обчислення. URL : http://uk.wikipedia.org/wiki (дата звернення: 28.10.2020).

33. Переваги та недоліки використання хмарних технологій підприємствами України. URL : http://www.bsfa.edu.ua/files/konf2013/62.pdf (дата звернення: 28.10.2020).

34. Benefits of cloud computing. URL : http://www.verio.com/resource-center/articles/cloud-computing-benefits (дата звернення: 28.10.2020).

35. Специфіка інформаційних систем на основі технології cloud computing. URL : http://archive.nbuv.gov.ua/portal/natural/vcndtu/2011_53/29.htm (дата звернення: 28.10.2020).