МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 28.00.00.000 ПЗ

Група ІПм-24-2

Кукудяк Максим

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Кукудяк Максим Андрійович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Нейромережеві моделі та методи віртуалізації збоїв в хмарних обчисленнях

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Кукудяк М.А.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Михайлюк Ірина Романівна, к.п.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кукудяку Максиму Андрійовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Нейромережеві моделі та методи віртуалізації збоїв в хмарних обчисленнях”

керівник проекту (роботи) Михайлюк І.Р., к.п.н., доцент

затверджені наказом закладу вищої освіти від “ 05 ” листопада 2025 р. № 695/7

2. Строк подання студентом проекту (роботи) 15 грудня 2025 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні методи побудови та функціонування інформаційних нейромережевих технологій

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз нейромережевих моделей та методів визначення збоїв в хмарних обчисленнях
2. Представлення методів віртуалізації в хмарному середовищі з використанням нейромереж
3. Імплементація нейромережевих моделей віртуалізації подій та збоїв в хмарних платформах
4. Навчання нейромережі з використанням алгоритму зворотного поширення помилки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Категорії хмарних обчислень (рис. 1.1)
2. Парадигма хмарних обчислень (рис. 1.2)
3. Кластеризація хмарних послуг (рис. 1.3)
4. Модель нейронної мережі прямого поширення (рис. 1.4)
5. Нейрон у нейронній мережі прямого поширення (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2025	виконано
2	Аналіз нейромережевих моделей та методів визначення збоїв в хмарних обчисленнях	29.09.2025	виконано
3	Представлення методів віртуалізації в хмарному середовищі з використанням нейромереж	15.10.2025	виконано
4	Імплементація нейромережевих моделей віртуалізації подій та збоїв в хмарних платформах	08.11.2025	виконано
5	Навчання нейромережі з використанням алгоритму зворотного поширення помилки	20.11.2025	виконано
6	Оцінка ефективності прогнозування відмов	01.12.2025	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2025	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 77 с., 25 рис., 11 табл., 34 джерела.

Тема: Нейромережеві моделі та методи віртуалізації збоїв в хмарних обчисленнях

Мета магістерської роботи: розроблення та дослідження нейромережевої моделі і методів віртуалізації збоїв у хмарних обчислювальних системах з метою підвищення їх надійності, ефективності прогнозування відмов.

Об'єкт дослідження: процеси виникнення, прогнозування та віртуалізації збоїв у хмарних обчислювальних середовищах.

Предмет дослідження: нейромережеві моделі, алгоритми та методи аналізу системних подій, спрямовані на виявлення, прогнозування та віртуалізацію збоїв у хмарних платформах.

Результати дослідження

В роботі проведено навчання моделі із використанням алгоритму зворотного поширення помилки, виконано оптимізацію архітектури мережі, перевірку коректності результатів і валідацію на тестових даних;

Висновок

Доведено, що впровадження нейромережевого прогнозування у системи моніторингу дозволяє своєчасно виявляти потенційні збої, підвищуючи рівень надійності та скорочуючи час відновлення працездатності сервісів.

НЕЙРОННА МЕРЕЖА, ХМАРНІ ОБЧИСЛЕННЯ, ВІРТУАЛІЗАЦІЯ, ЗБОЇ, ПРОГНОЗУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ЖУРНАЛ ПОДІЙ, МАЙНІНГ ДАНИХ, НАДІЙНІСТЬ СИСТЕМИ.

ABSTRACT

Master Thesis: 77 pp., 25 fig., 11 tab., 34 sources.

Topic: Neural network models and methods of virtualization of failures in cloud computing

Purpose of the master's thesis: development and study of a neural network model and methods of virtualization of failures in cloud computing systems in order to increase their reliability, efficiency of failure prediction.

Object of research: processes of occurrence, prediction and virtualization of failures in cloud computing environments.

Subject of research: neural network models, algorithms and methods of analysis of system events aimed at detection, prediction and virtualization of failures in cloud platforms.

Research results

The work involved training the model using the error backpropagation algorithm, optimization of the network architecture, verification of the correctness of the results and validation on test data;

Conclusion

It is proven that the implementation of neural network prediction in monitoring systems allows for timely detection of potential failures, increasing the level of reliability and reducing the time to restore the operability of services.

NEURAL NETWORK, CLOUD COMPUTING, VIRTUALIZATION, FAILURES, FORECASTING, ARTIFICIAL INTELLIGENCE, EVENT LOG, DATA MINING, SYSTEM RELIABILITY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

VMM Virtual Machine Monitor - монітор віртуальних машин

IaaS - Infrastructure as a Service - Інфраструктура як послуга

OS - Operating System - Операційна система

RFC - Request for Comments - Запит на коментарі (стандарт)

evt - Event (file extension) - Подія (розширення файлу журналу Windows)

NN - Neural Network - Нейронна мережа

NAR - Nonlinear Autoregressive - Нелінійна авторегресія

NARX - Nonlinear Autoregressive with External Input,Нелінійна авторегресія із зовнішнім входом

LM - Levenberg–Marquardt

VM - Virtual Machine - Віртуальна машина

SRService - System Restore Service - Служба відновлення системи

ВСТУП

Актуальність теми.

Сучасний етап розвитку інформаційних технологій характеризується масштабною інтеграцією хмарних обчислень у всі сфери діяльності – від бізнесу до наукових досліджень та державного управління. Хмарні платформи стали ключовою інфраструктурою для зберігання, обробки й доступу до даних, що зумовлює підвищені вимоги до їхньої надійності, продуктивності та стійкості до відмов. У зв'язку з постійним зростанням обсягів даних, кількості користувачів і складності обчислювальних процесів, збої в роботі хмарних систем набувають системного характеру та можуть призводити до значних втрат.

Одним із перспективних напрямів підвищення надійності хмарних систем є використання методів штучного інтелекту, зокрема нейронних мереж, для прогнозування та віртуалізації подій відмов. Завдяки здатності нейромереж виявляти приховані закономірності у великих обсягах даних, з'являється можливість не лише прогнозувати потенційні збої, але й моделювати їхній вплив на інфраструктуру з метою запобігання деградації сервісів. Такий підхід формує основу інтелектуальних систем самовідновлення, що відповідає сучасній тенденції до побудови self-healing cloud systems.

Розробка та дослідження нейромережевих методів віртуалізації збоїв у хмарних обчисленнях є актуальною науково-технічною задачею, яка поєднує інструменти штучного інтелекту, теорії надійності, системного аналізу та обчислювальної інженерії.

Актуальність дослідження зумовлена необхідністю забезпечення безперервності функціонування хмарних обчислювальних систем в умовах зростання їх масштабності, динамічності та складності внутрішніх взаємозв'язків. Збої в роботі хмарних сервісів можуть мати критичні

наслідки — від простоїв корпоративних додатків до втрати даних користувачів.

Традиційні методи моніторингу й реагування на збої ґрунтуються на ручному аналізі журналів подій і застосуванні порогових моделей, що є неефективним у середовищах з високою динамікою змін. Натомість нейромережеві моделі, зокрема рекурентні та авторегресивні мережі, здатні відстежувати складні нелінійні залежності між параметрами системи та прогнозувати ймовірність виникнення відмов ще до їхнього фактичного прояву.

Крім того, розвиток технологій віртуалізації дозволяє створювати симуляційні моделі збоїв, які можуть бути використані для оцінювання стійкості системи без втручання у виробниче середовище. Це відкриває перспективу формування віртуальних двійників хмарних систем, які здатні відтворювати, аналізувати та запобігати збоям у режимі реального часу.

Таким чином, актуальність роботи визначається потребою у створенні інтегрованої нейромережевої моделі, що дозволяє автоматизувати процеси виявлення, прогнозування та віртуалізації збоїв у хмарних обчисленнях, забезпечуючи високу надійність, адаптивність і ефективність IT-інфраструктур.

Метою роботи є розроблення та дослідження нейромережевої моделі і методів віртуалізації збоїв у хмарних обчислювальних системах з метою підвищення їх надійності, ефективності прогнозування відмов.

Об’єктом дослідження є процеси виникнення, прогнозування та віртуалізації збоїв у хмарних обчислювальних середовищах.

Предметом дослідження є нейромережеві моделі, алгоритми та методи аналізу системних подій, спрямовані на виявлення, прогнозування та віртуалізацію збоїв у хмарних платформах.

Для досягнення поставленої мети у роботі розв’язуються такі **завдання**:

1. Провести аналіз архітектури, моделей обслуговування та показників надійності хмарних обчислень.

2. Дослідити сучасні підходи до прогнозування відмов із використанням методів штучного інтелекту та нейронних мереж.

3. Формалізувати процеси обробки журналів подій та застосувати методи частого майнінгу для виявлення закономірностей у послідовностях збоїв.

4. Побудувати архітектуру нейронної мережі типу NARX для прогнозування подій відмов у хмарному середовищі.

5. Реалізувати та навчити нейромережеву модель на основі отриманих даних, провести експериментальну перевірку її ефективності.

У роботі використано такі методи:

- методи нейромережевого моделювання (зокрема архітектура NARX) для прогнозування динаміки збоїв;

- методи частого майнінгу та аналізу послідовностей подій для формалізації закономірностей у журналах системних логів;

- методи машинного навчання та алгоритми зворотного поширення помилки для навчання моделей;

- імітаційне моделювання для відтворення поведінки користувачів і тестування стійкості хмарних платформ.

Наукова новизна отриманих результатів

Запропоновано нейромережеву модель віртуалізації збоїв у хмарних системах, що поєднує прогнозування подій відмов і їх симуляційне відтворення та удосконалено метод обробки системних журналів подій шляхом інтеграції частого майнінгу з нейромережевим аналізом часових послідовностей.

Практичне значення результатів

Розроблена нейромережева модель та методика віртуалізації збоїв можуть бути інтегровані у системи моніторингу хмарних дата-центрів,

програмні платформи IaaS-провайдерів та сервіси управління обчислювальними ресурсами.

Практичне застосування запропонованого підходу забезпечує:

- підвищення точності прогнозування та швидкості реагування на збої;
- скорочення часу відновлення сервісів після відмов;
- зниження ризику деградації продуктивності хмарних систем.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 77 сторінок, і містить 25 рисунків, 11 таблиць, список використаних джерел із 34 найменувань.

РОЗДІЛ 1. АНАЛІЗ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ТА МЕТОДІВ ВИЗНАЧЕННЯ ЗБОЇВ В ХМАРНИХ ОБЧИСЛЕННЯХ

1.1. Аналіз та прогнозування надійності хмарних систем на основі журналів помилок з використанням нейронних мереж

Хмарні обчислення становлять один із ключових напрямків сучасних передових технологічних трендів, що визначають розвиток багатьох галузевих аспектів. Наразі сфера хмарних обчислень є інтенсивним конкурентним середовищем для провідних технологічних корпорацій, перемога в якому потенційно забезпечує лідерство у наступному поколінні технологій.

З технічної точки зору, хмарні обчислення класифікуються на три основні категорії, кожна з яких надає користувачам відмінні важливі послуги:

- Інфраструктура як послуга (Infrastructure as a Service, IaaS),
- Програмне забезпечення як послуга (Software as a Service, SaaS),
- Платформа як послуга (Platform as a Service, PaaS).

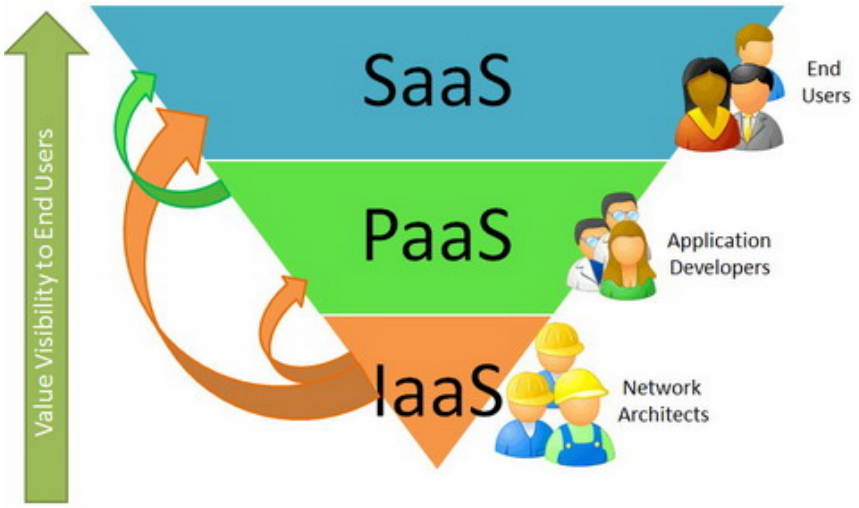

Рис. 1.1. Категорії хмарних обчислень

Стандартні метрики оцінки рівня надійності хмарних обчислень зазвичай ґрунтуються на двох підходах:

- Угоди про рівень обслуговування (Service Level Agreements, SLA,
- Якість обслуговування (Quality of Service, QoS).

Дана магістерська робота зосереджується на аналізі журналів помилок систем IaaS як складової QoS в контексті надійності хмарних систем IaaS. Для уточнення, IaaS є еволюцією традиційної системи віртуалізації, де множина віртуальних машин (ВМ) з різними операційними системами (ОС) функціонує на одній фізичній машині (ФМ), що має достатню обчислювальну потужність. У контексті хмарних обчислень ФМ виконує роль хост-машини, а ВМ — роль гостьових машин.

У зв'язку з обмеженим доступом до реальної хмарної системи, це дослідження вивчатиме технічний рівень надійності хмарних систем IaaS за допомогою імітованої системи віртуалізації. Шляхом збору та аналізу журналів подій, які генеруються системою віртуалізації, можна отримати загальну оцінку технічного рівня надійності системи на основі частоти виникнення помилкових подій. Надалі ці події будуть використані у моделі нейронної мережі часового ряду для ідентифікації патернів відмов системи, а також для прогнозування наступної потенційної помилкової події у системі віртуалізації.

1.2. Аналіз хмарних обчислень як домінуючої парадигми IT-індустрії

Сьогодні хмарні обчислення (Cloud Computing) визнані оптимальним рішенням у сфері інформаційних технологій (IT). Вони демонструють значні переваги у порівнянні з традиційними моделями надання сервісів. Ці переваги включають впровадження моделі "плати за використання" (pay-as-you-go) та можливість миттєвого доступу до необхідних обчислювальних

ресурсів, що усуває часові та ресурсні витрати на побудову й конфігурацію власної інфраструктури.

Економічний ефект полягає не лише в економії часу та фінансових ресурсів, але й у подоланні обмежень апаратного забезпечення в IT-галузі завдяки гнучкості та масштабованості хмарних рішень. Користувачі отримують доступ до обчислювальних систем, обчислювальна потужність яких може бути в тисячі разів вищою, без необхідності капітальних інвестицій у їх створення. Крім того, перехід до хмарних технологій виключає необхідність для користувачів витрачати час на ремонт, технічне обслуговування та оновлення власних бізнес-орієнтованих обчислювальних систем. Такі операційні витрати, зазвичай, становлять до 30% від суми, витраченої на початкову побудову та налаштування системи.

1.2.1. Архітектура та економічна ефективність хмарних обчислень

Модель хмарних обчислень структурно охоплює постачальників хмарних послуг (Cloud Service Providers, CSP) та користувачів хмарних послуг. Постачальники хмарних послуг – це організації, що надають ці послуги кінцевим споживачам. Парадигма хмарних обчислень (візуалізована на рис. 1.2) передбачає, що користувачі використовують власні електронні пристрої для доступу до хмарного сховища, хмарного медіа-контенту або власних хмарних віртуальних машин.

Ключовою перевагою цієї моделі є мінімальні вимоги до специфікацій електронних пристроїв користувачів. Для забезпечення стабільного використання хмарних сервісів може бути достатнім лише наявність якісних дисплеїв та мережевих адаптерів середньої або високої швидкості.

Економічна доцільність хмарних обчислень порівняно з традиційними моделями, що базуються на дата-центрах, може бути формалізована наступною умовою [22]:

$$\text{UserHours}_{\text{хмарні}} \times (\text{Дохід} - \text{Cost}_{\text{хмарні}}) \geq \text{UserHours}_{\text{дата-центр}} \times \left(\text{Дохід} - \frac{\text{Cost}_{\text{дата-центр}}}{\text{Використання}}\right)$$

де UserHours позначає кількість годин споживання послуги користувачами, Дохід – загальний дохід, який генерується постачальниками послуг, а Cost – операційні витрати постачальників послуг на підтримку функціонування.

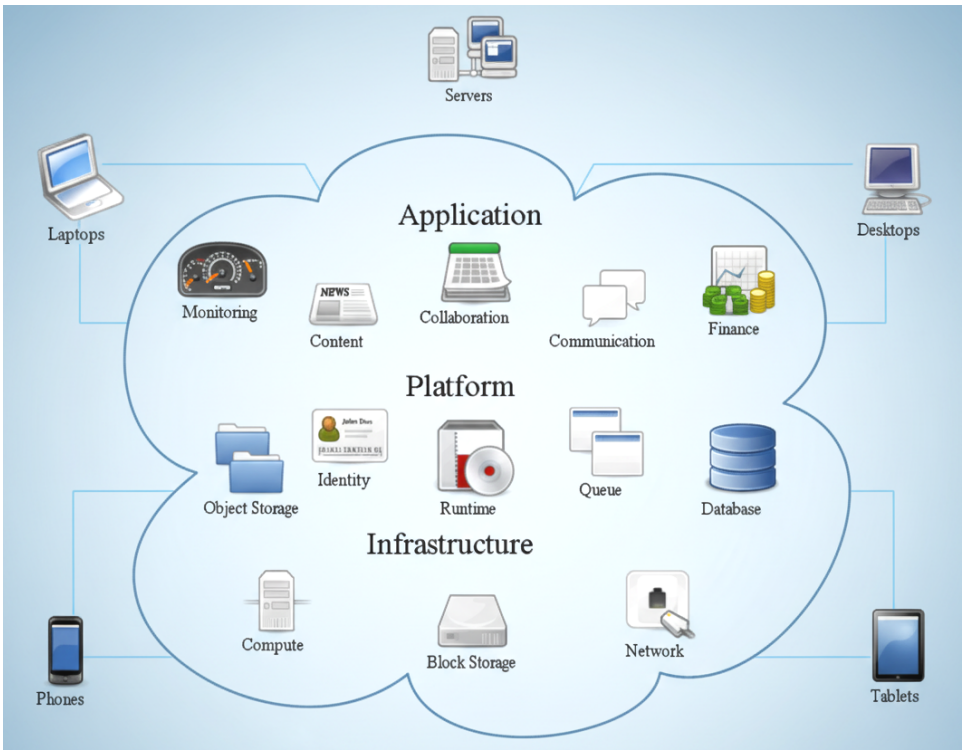

Рис. 1.2. Парадигма хмарних обчислень

1.2.2. Моделі обслуговування хмарних обчислень

Хмарні обчислення визначаються як надання програмного забезпечення як послуги через мережу Інтернет разом із необхідним апаратним та програмним забезпеченням для його функціонування. Відповідно до різних моделей обслуговування, хмарні обчислення класифікуються на три основні типи:

- Програмне забезпечення як послуга (Software as a Service, SaaS)
- Платформа як послуга (Platform as a Service, PaaS)
- Інфраструктура як послуга (Infrastructure as a Service, IaaS)

Рисунок 1.2 надає огляд послуг, що належать до цих трьох моделей. Серед них IaaS є найбазовішим шаром. IaaS — це модель, за якою

постачальники хмарних послуг надають апаратну обчислювальну потужність через Інтернет. Багато провідних технологічних компаній розглядають IaaS як свою ключову стратегію в хмарних обчисленнях (наприклад, Amazon EC2, VMware vCloud). У другому розділі цієї дисертації буде імітовано віртуалізоване середовище для проведення експериментів з оцінки надійності хмарних віртуальних машин IaaS.

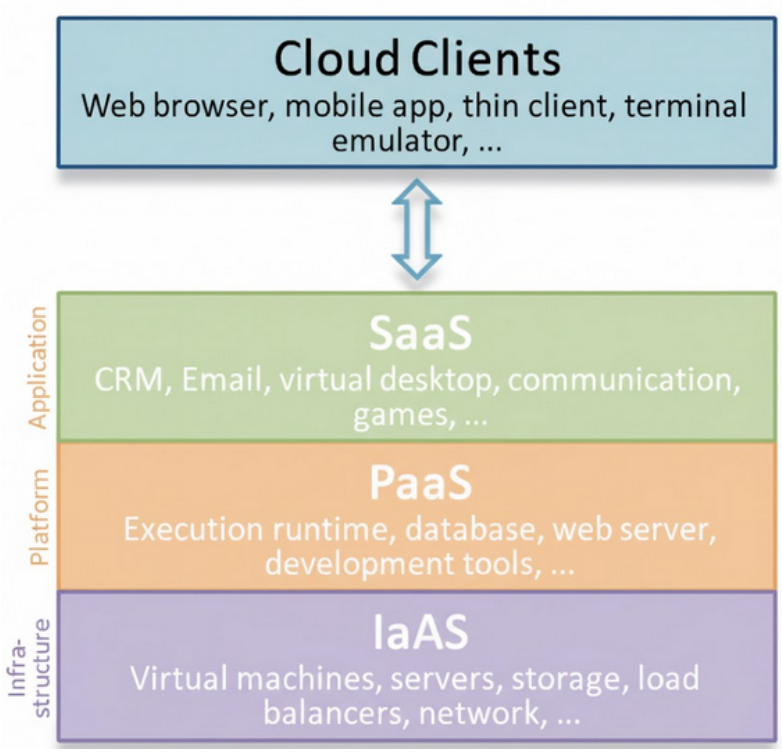

Рис. 1.3. Кластеризація хмарних послуг

1.2.3. Метрики надійності та якості обслуговування

Для верифікації рівня надійності IaaS застосовуються два основні підходи:

- Угоди про рівень обслуговування (Service Level Agreements, SLA) – це офіційний контракт, укладений між постачальником хмарних послуг та користувачами.

- Якість обслуговування (Quality of Service, QoS) – це гарантія з боку постачальників хмарних послуг щодо безперервності надання сервісів користувачам з якістю, що не поступається рівню, визначеному в SLA.

Рівень надійності (Reliability) відноситься до тривалості безперервного часу, протягом якого постачальники послуг зобов'язуються надавати стабільний сервіс користувачам. Він кількісно вимірюється кількістю збоїв, що виникають на віртуальних машинах користувачів [4].

Якщо P_R — ймовірність виникнення помилок, nf — кількість завдань, що зазнали збою, nt — загальна кількість завдань, які очікується виконувати нормально протягом того ж часу, а T — загальний час роботи хмарних послуг користувачами, надійність визначається як:

$$Reliability = P_R \text{ x } T = \left(1 - \frac{n_f}{n_t}\right) \text{ x } T$$

У таблиці 1.1 наведено деякі загальновживані метрики, які використовуються для оцінки SLA (Service-Level Agreement) хмарних послуг [3].

Таблиця 1.1.

Метрики оцінки SLA

SLA постачальника	**SLA користувача**
Пропускна здатність	Доступність
Безпека	Надійність
Білінг	Час відгуку
Моніторинг	Рівень обслуговування
Конфіденційність	Політика інтернаціоналізації
Компенсація	Міграція
Балансування навантаження	Служба підтримки
Відновлення	

1.3. Застосування штучного інтелекту та нейронних мереж у моделюванні процесів хмарних обчислень

Штучний інтелект (ШІ) (Artificial Intelligence, AI) привертає значну академічну увагу в галузі комп'ютерних наук. В загальному сенсі, ШІ являє собою сукупність методологій для проектування та програмування машинного інтелекту, що демонструє імітацію певного рівня людського інтелекту. Відповідно, ШІ надає машині здатність до когнітивних процесів і дій, аналогічних людським, у конкретних ситуаціях і за певних умов.

1.3.1. Нейронні мережі та механізми навчання

Нейронна мережа (Neural Network, NN), яку іноді розглядають як підмножину машинного навчання, є напрямком ШІ, що архітектурно натхненний моделлю нейронів біологічного мозку людини. Теоретична аналогія з біологічними нейронами полягає в тому, що знання набуваються шляхом навчання і зберігаються у вузлах з'єднань нейронів, які технічно відомі як ваги (weights).

Таким чином, фундаментальна мета нейронної мережі полягає в навчанні системи ШІ шляхом подачі їй визначеного вхідного набору даних. Після ітеративного процесу навчання система ШІ формує узагальнені знання про набір даних, що дозволяє їй демонструвати загальне розуміння щодо поданих даних.

Існують два основні парадигми навчання:

- Навчання з учителем (Supervised Learning) - користувачі надають системі як вхідний набір даних, так і відповідний цільовий набір даних (очікуваний вихід).

- Навчання без учителя (Unsupervised Learning або сліпе навчання) - користувачі надають системі лише вхідний набір даних, і машина повинна самостійно виявити правила або патерни в даних.

Нейронні мережі мають велику кількість архітектур, кожна з яких призначена для різних навчальних завдань. Нейронна мережа прямого поширення (Feedforward Neural Network, FNN) є однією з найбільш поширених і фундаментальних моделей нейронних мереж.

Основна концепція процесу навчання нейронної мережі ілюструється наступним чином: нехай X — це вхідний набір даних із N елементів, а T — цільовий набір даних із N елементів (N=1000 у прикладі). Перші M елементів (M=500) з X і T будуть подані нейронній мережі для фази навчання. Після досягнення оптимального результату навчання (завершення навчання) мережа приймає останні (N−M) елементів з X як вхідні дані і генерує набір Y із (N−M) елементів як прогнозований вихід. Цей набір Y порівнюється з останніми (N−M) елементами в T для верифікації точності (коректності) нейронної мережі.

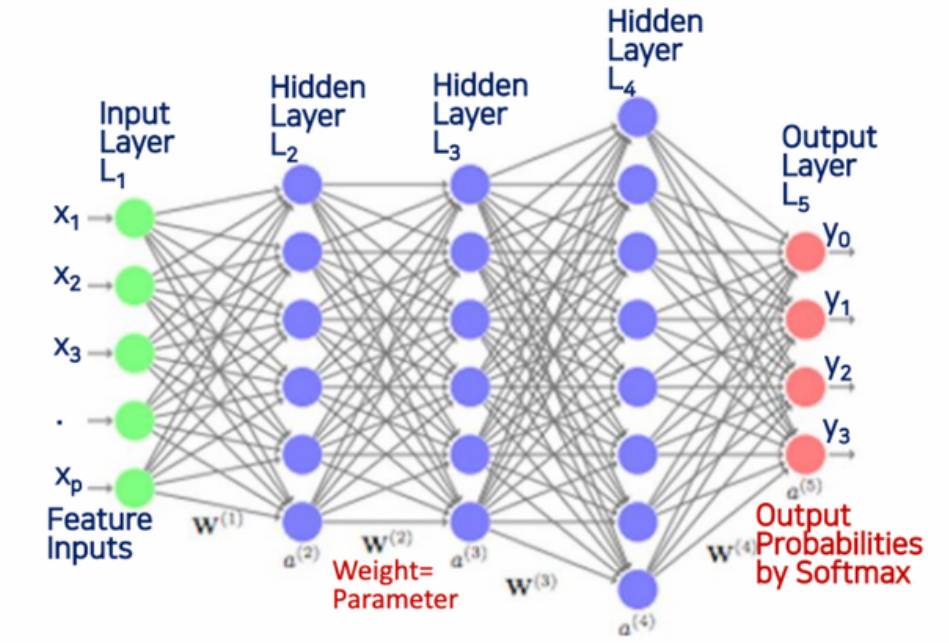

Рис. 1.4. Модель нейронної мережі прямого поширення

Рисунок 1.4 ілюструє базову модель нейронної мережі прямого поширення. Мережева модель складається з трьох типів шарів: одного вхідного шару, одного або більше прихованих шарів та одного вихідного шару. Кожен шар містить кілька вузлів, і кожен вузол спрямовано

сполучається з усіма суміжними вузлами з попереднього та наступного шарів [9]. Під час використання дані передаються від вузлів одного шару до вузлів наступного шару, доки не досягнуть вихідного шару, при цьому зворотний зв'язок між шарами відсутній.

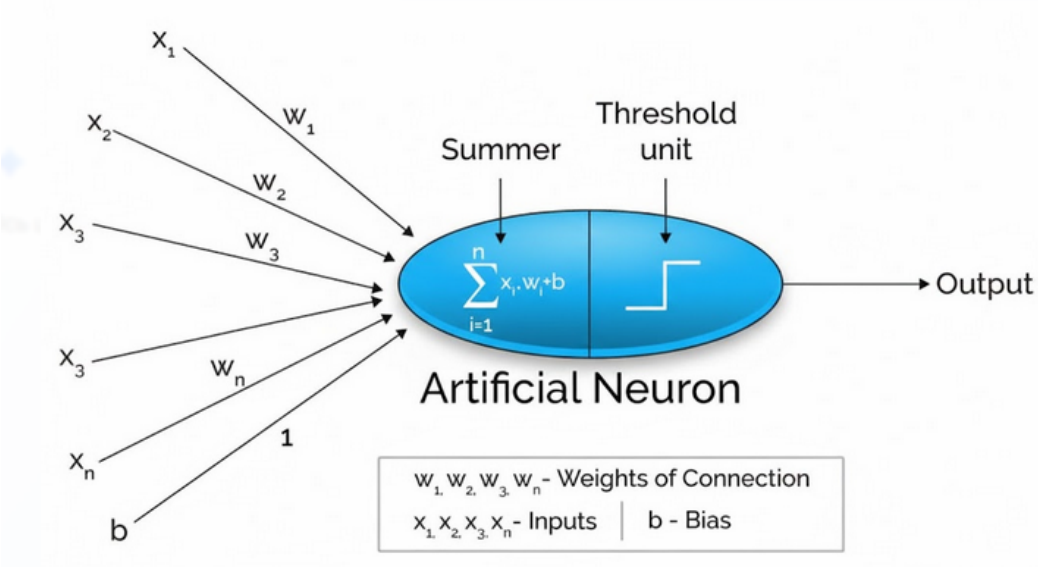

Рис. 1.5. Нейрон у нейронній мережі прямого поширення

Рисунок 1.5 візуалізує функціональну одиницю (нейрон) у нейронній мережі прямого поширення. Нехай x1,x2,…,xn — це вузли в попередньому шарі; w1,w2,…,wn — це ваги з'єднань, що ведуть від x1,…,xn до вузла xj у поточному шарі. Зсув b (bias) є додатковим вузлом у прихованому шарі, де xb +wb=1. Сумарний вхідний фактор a для нейрона j визначається як:

$$a = \sum_{i=1}^{n} (x_i \cdot w_i) + b$$

Вихід a_k від вузла j до вузла k у наступному шарі визначається як:

$$a_k = f(a) = \frac{1}{1 + e^{-a}}$$

де f — сигмоїдна функція активації (sigmoid activation function), яка є типовою функцією активації для нейронної мережі.

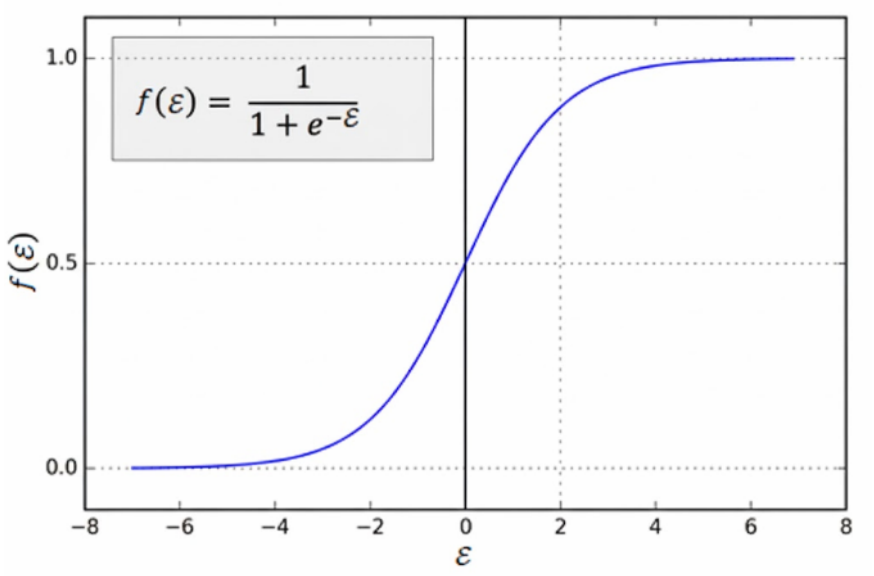

Рис. 1.6. Візуалізація сигмоїдної функції активації

Під час процесу навчання набір даних розділяється на два піднабори: перший використовується для навчання, а другий — для перевірки (валідації). Після завершення процесу навчання, результат перевірки кількісно оцінюється за значенням середньоквадратичної помилки (Mean Squared Error, MSE). Нехай Y={y1,y2,…,yn} — набір отриманих вихідних результатів, а T={T1,T2,…,Tn} — відповідний набір оригінальних цільових даних. Формула для MSE має вигляд:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - T_i)^2$$

Нейронні мережі прямого поширення можуть бути застосовані для широкого спектра завдань, включаючи апроксимацію функцій (підгонка кривої вхід-вихід), розпізнавання та класифікацію патернів, кластеризацію (у випадку навчання без учителя) та динамічне прогнозування часових рядів. У

контексті цієї роботи буде використано нейронну мережу часового ряду для прогнозування наступної помилкової події в імітованому хмарному середовищі IaaS.

1.4. Формалізація та застосування методів частого майнінгу для аналізу журналів системних подій хмарної платформи

З огляду на експоненціальне зростання обсягу та категоризації даних, що стало можливим завдяки розвитку технологій масового зберігання, термін Data Science набуває центрального значення в цій галузі. У загальному сенсі, Наука про дані охоплює сукупність методів і технік, спрямованих на використання наявних даних для надання відповідей на дослідницькі питання, які традиційно класифікуються за чотирма основними категоріями:

- Що сталося? (Звітність / Descriptive Analytics)
- Чому це сталося? (Діагностика / Diagnostic Analytics)
- Що станеться? (Прогнозування / Predictive Analytics)
- Що найкраще може статися? (Рекомендація / Prescriptive Analytics)

Наука про дані інтегрує міждисциплінарні галузі комп'ютерних наук, зокрема: майнінг даних (Data Mining), машинне навчання (Machine Learning), майнінг процесів (Process Mining), бази даних, великомасштабні розподілені обчислення, візуалізація та візуальна аналітика, а також аспекти соціальних наук, конфіденційності та безпеки. Багато з цих дисциплін застосовуються в рамках даного дослідження.

1.4.1. Частий майнінг епізодів

У певних типах наборів даних, зокрема в журналах системних подій, де записи упорядковані в часовій послідовності, епізод визначається як множина подій, що починається з певної початкової події активності та закінчується завершальною подією активності. Частота епізоду (support) вказує, як часто цей епізод виникає в заданій послідовності. Частий майнінг

епізодів (Frequent Episode Mining) — це техніка, призначена для ідентифікації всіх епізодів, частота яких перевищує або дорівнює попередньо визначеному користувачем порогу [21].

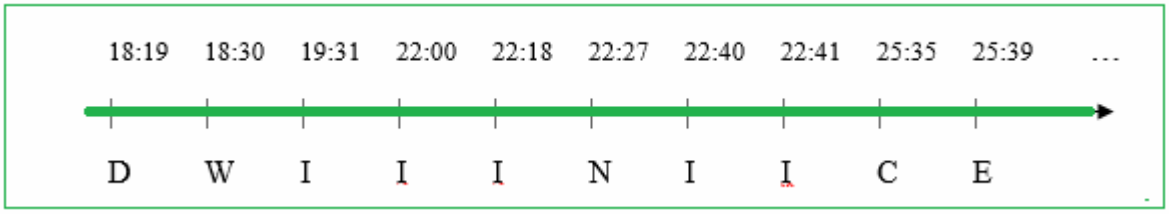

Рис. 1.7. Приклад послідовності подій

Нехай E={e1,e2,e3,…,em} буде набором елементарних подій, де ei={Ei1 ,Ei2,…,Ein,ti}. Послідовність подій S у часовому порядку позначається як S={(E1,t1),(E2,t2),…,(En,tn)}, де n≤m. У цьому визначенні Ei⊆E, a ti — це часова мітка події Ei, причому ti<tk⇔1≤i<k≤n. Наприклад, на рисунку 1.7 представлена послідовність

S=<({D},enoxa(18:19)),({W},enoxa(18:30)),(…),({C},enoxa(25:39))>.

Тут функція епоха(t) перетворює час t з формату мм:сс у формат епохи (epoch).

Епізод α (α=∅) — це впорядкована множина подій $e_1 \to e_2 \to \ldots \to e_i \to \ldots \to e_k$, де ei∈E та i∈[1,k], і ei передує ej⇔1≤i<j≤k. Довжина епізоду визначається кількістю подій у ньому; наприклад, α=I→I→C→E має довжину 4 і називається 4-епізодом.

Для двох заданих епізодів α=e1→…→en та β=e1′→…→ek′, β є під-епізодом α (позначення β≤α) тоді і лише тоді, якщо в часовому порядку для кожного ej′∈[1,k] існує подія ep [1,n] у тому ж часовому порядку, така що ej′ =ep. Наприклад, якщо β=I→E (2-епізод) і α=I→C→E (3-епізод), то β≤α, але β′=E→I не є під-епізодом α.

Виникнення епізоду α=e1→…→ek, позначене як [t1…tk], є дійсним тоді і лише тоді, якщо (1) для всіх i∈[1,k], ti⊆ei; (2) t1<t2<…<tk; та (3) tk−t1 <δ, де δ — максимальне вікно виникнення, попередньо визначене користувачем. На рисунку 1.7, [епоха(18:19), епоха(18:30), епоха(19:31)] є виникненням епізоду D→W→I.

Два виникнення [t1…tk] та [t1′…tk′] епізоду α вважаються еквівалентними, якщо t1=t1′ та tk=tk′; у такому випадку вони враховуються як одне виникнення. Виникнення α позначається як (α,[t1,tk]), а [t1,tk] називається вікном виникнення α. Наприклад, виникнення [епоха(22:27),епоха(22:40),епоха(25:35)] та [епоха (22:27), епоха(22:41), епоха (25:35)] епізоду α=N→I→C є еквівалентними.

Виникнення [t1,t2] є мінімальним виникненням, якщо не існує іншого виникнення [t1′,t2′], яке б йому підпорядковувалося, тобто t1<t1′ та t2′<t2. Множина всіх різних мінімальних виникнень епізоду α позначається як moSet(α). Підтримка (Support) епізоду α, позначена як sp(α), визначається як кількість елементів у moSet(α). Наприклад, на рис. 1.7, при δ=2, moSet(I→I)={[епоха(19:31),епоха(22:00)],[епоха(22:00),епоха(22:18)],[епоха(22:40),епоха(22:41)]}, отже sp(I→I)=3. Епізод α вважається частим, якщо sp(α)≥min_sup, де min_sup — мінімальний поріг підтримки, попередньо визначений користувачем.

1.4.2. Журнали подій та майнінг процесів

Техніки майнінгу процесів (Process Mining) застосовують дані системних транзакцій або журнали бізнес-подій для виявлення різних моделей бізнес-процесів. Застосування цієї техніки часто розкриває моделі, існування яких може бути невідомим учасникам галузі/бізнесу. Крім того, майнінг процесів використовується для моніторингу та оптимізації існуючих бізнес-процесів шляхом аналізу відповідних журналів подій.

Журнал подій L формалізується як кортеж

$$L = (E, AN, AI, C, act, type, time, res, case, name)$$

У цьому кортежі E, AN, AI, C являють собою набори подій, назв активностей, екземплярів активностей та випадків (case) відповідно, причому E={e1,e2,…,en}. Нехай UET={Інформація, Попередження, Помилка,…} буде списком типів подій, а Ures = {Dhcp, IPSec, HTTP, DCOM, W32Time, Setup,…} — списком системних ресурсів. Функції $EET \in E \rightarrow UET$ та $res \in E \rightarrow Ures$ відображають події на їхні типи та відповідні системні ресурси. Функції $EAE \in AI$ та AAE та $EAC \in C$ відображають екземпляри активностей на випадки та назви активностей.

Таблиця 1.2 надає спрощений фрагмент журналу подій, що імітує хмарну обчислювальну систему, де кожен рядок представляє екземпляр активності. Наприклад, перший процес (case ID = 1) починається з активності "WG- Warning " (ресурс "Dhcp") і закінчується активністю "WG- Error" (ресурс "Setup"). Початкові та кінцеві точки одного процесу встановлюються користувачем, і це правило застосовується до всього набору даних журналу подій.

Таблиця 1.2.

Фрагмент журналу подій імітованої хмарної обчислювальної системи

Case ID (Ідентифікатор випадку)	**Resource (Ресурс)**	**Activity (Активність)**	**Time Generated (Час генерації)**
1	Dhcp	WG-Warning	11/10/2025 00:18:19
1	IPSec	WG-Information	11/10/2025 00:18:19
1	Workstation	WG-Information	11/10/2025 00:18:30
1	HTTP	WG-Information	11/10/2025 00:19:31
1	Setup	WG-Information	11/10/2025 00:22:00
1	Setup	WG-Error	11/10/2025 00:22:00
2	DCOM	WG-Information	11/10/2025 00:22:18
2	EventLog	WG-Information	11/10/2025 00:22:18
2	EventLog	WG-Information	11/10/2025 00:22:18

Case ID (Ідентифікатор випадку)	Resource (Ресурс)	Activity (Активність)	Time Generated (Час генерації)
2	IPSec	WG-Information	11/10/2025 00:22:27
2	SRService	WG-Information	11/10/2025 00:22:27
2	W32Time	WG-Information	11/10/2025 00:22:27
2	DCOM	WG-Error	11/10/2025 00:22:27
...	...	...	...

Майнінг процесів поділяється на три основні категорії, що візуалізовані на рис. 1.8:

1. Відкриття процесів (Discovery).

Найбільш корисний для встановлення нової моделі процесу на основі журналу подій.

2. Перевірка відповідності (Conformance Checking).

Використовується для порівняння та моніторингу існуючої моделі процесу з фактичними журналами, щоб перевірити дотримання визначених правил та ідентифікувати відхилення.

3. Аналіз продуктивності (Enhancement / Performance Analysis).

Використовує узгодження між моделлю процесу та продуктивністю системи в реальному часі (отримане в результаті перевірки відповідності) для виявлення вузьких місць системи.

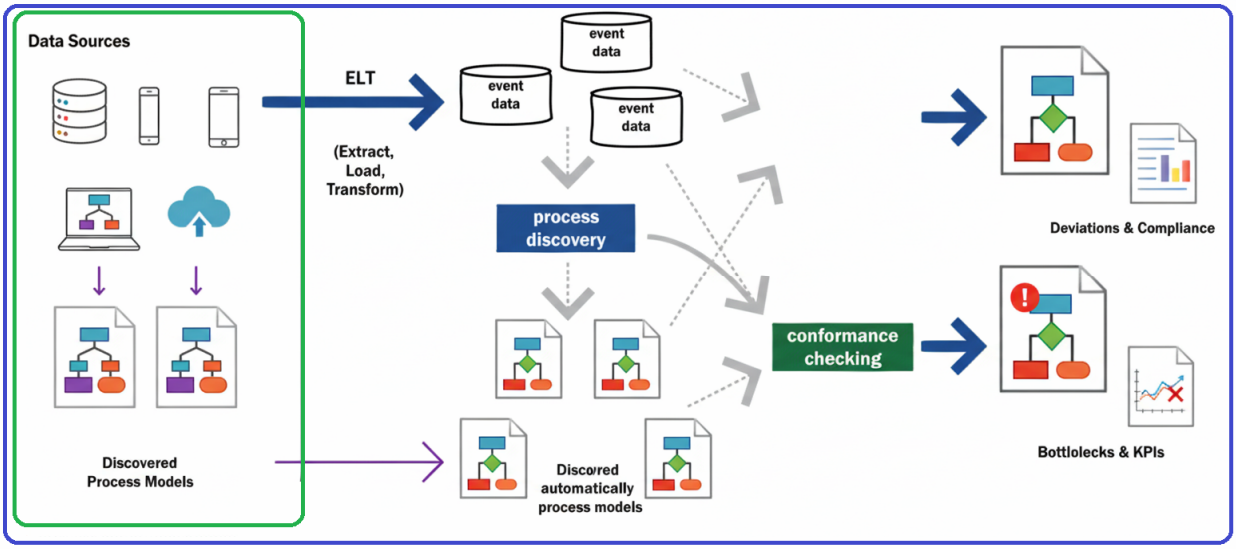

Рис. 1.8. Типи технік майнінгу процесів

Висновки до розділу

У першому розділі здійснено системний аналіз сучасних підходів до прогнозування збоїв у хмарних обчисленнях, виявлено основні закономірності формування помилок і особливості їхнього відображення в журналах подій. Доведено, що традиційні статистичні методи аналізу журналів подій не забезпечують необхідної точності прогнозування у складних розподілених системах через нелінійність, стохастичність і високий рівень взаємозалежності параметрів. Обґрунтовано доцільність використання штучних нейронних мереж (зокрема моделей глибинного навчання) для аналізу журналів системних подій. Визначено, що поєднання нейромережевих алгоритмів із методами частого майнінгу дозволяє виявляти приховані залежності між подіями збоїв та часовими інтервалами їхнього виникнення. Розроблено узагальнену схему інтеграції частого майнінгу епізодів із процесами навчання нейронних моделей, що формує основу для подальшого моделювання процесів віртуалізації відмов.

РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ МЕТОДІВ ВІРТУАЛІЗАЦІЇ В ХМАРНОМУ СЕРЕДОВИЩІ З ВИКОРИСТАННЯМ НЕЙРОМЕРЕЖ

2.1. Підготовка емпіричних даних для навчання нейронної мережі

Для досягнення цілей навчання нейронної мережі (НМ) критично важливим є забезпечення високоякісного підготовленого набору даних, агрегованого з реальної операційної хмарної системи. Проте, зважаючи на відсутність доступу до журналів системних подій провайдерів хмарних послуг (ПХП), було розроблено альтернативне методологічне рішення.

З технічної точки зору, хмарний сервер функціонально еквівалентний стандартному серверу віртуалізації, доступ до якого користувачі здійснюють через мережу Інтернет, незалежно від його фізичного розміщення. Відповідно, існує можливість дослідження продуктивності хмарної системи шляхом використання аналогічної системи віртуалізації, за умови, що мережеві латентності та пов'язані проблеми у цьому експериментальному середовищі ідеалізовано встановлюються на нуль.

Ключовою складністю цього підходу є необхідність не лише аналогічної конфігурації системи, але й симуляції віртуального користувача або автоматизованих макросів, що імітують типову активність користувачів на гостьових машинах (віртуальних машинах, ВМ). Це є визначальним чинником для генерації репрезентативних журналів системних подій для емпіричних експериментів.

2.2. Віртуалізація в інфраструктурі як послуга (IaaS)

У моделі IaaS користувачам надається гостьова машина, що включає виділені віртуалізовані ресурси: сховище, центральний процесор (CPU, потенційно графічний процесор – GPU), пам'ять, мережеві інтерфейси та операційну систему (ОС). Це надає користувачам можливість виконувати всі

операції, які теоретично можливі на їхньому власному персональному комп'ютері (ПК).

Використання IaaS значно скорочує час, необхідний для конфігурації та налагодження обладнання. Крім того, це особливо вигідно для суб'єктів, яким потрібна обчислювальна потужність лише на короткий проміжок часу, оскільки IaaS пропонує високу гнучкість масштабування орендованих ресурсів. Наприклад, користувач Amazon EC2, що використовує дві гостьові машини (з одноядерним CPU 3,4 ГГц та 2 ГБ пам'яті кожна), може миттєво масштабувати їх до 100 машин з ідентичними характеристиками, якщо йому потрібна така обчислювальна потужність на наступні 60 секунд (або довше). Зауважимо, що Amazon EC2 стягне з такого користувача додаткову плату лише за одну хвилину використання ресурсів, що у 50 разів перевищує початковий обсяг, оскільки їхня послуга оплачується з посекундною тарифікацією, де 60 секунд є мінімальним порогом.

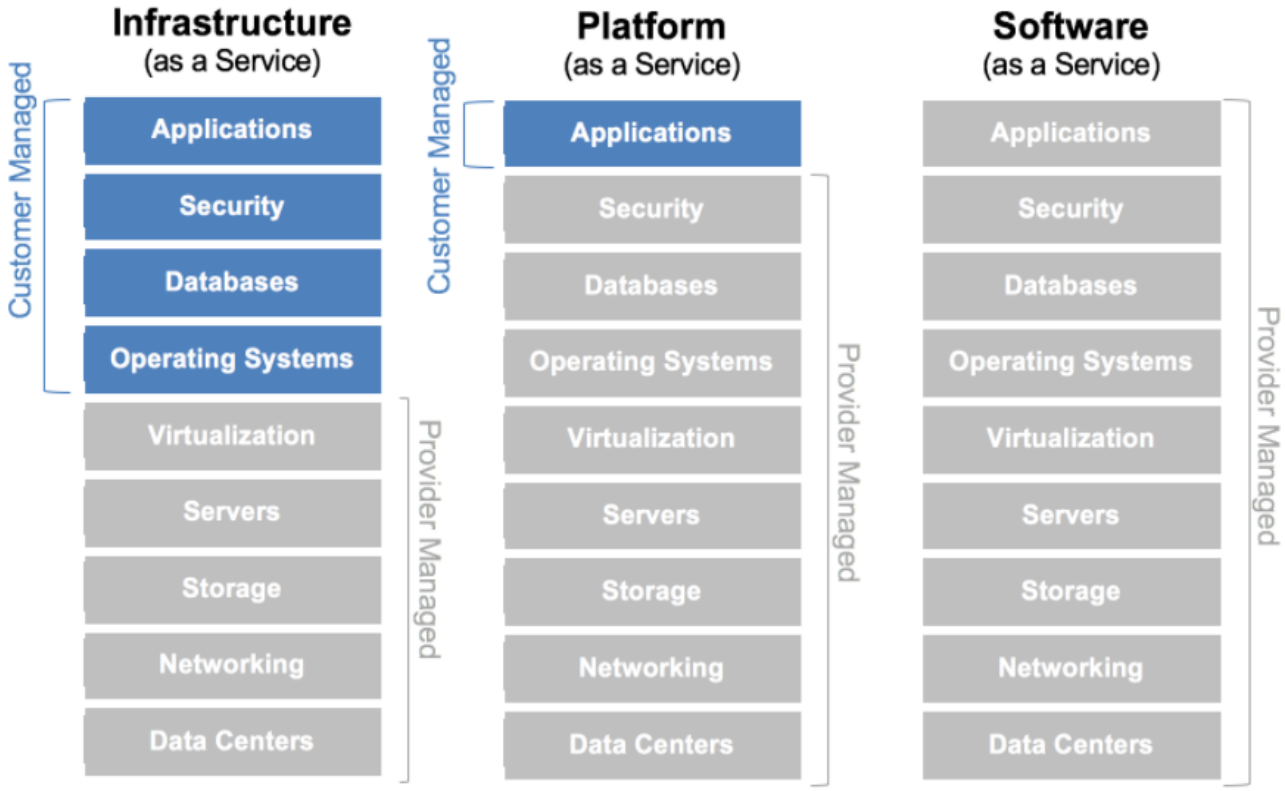

Рис. 2.1. Керувані користувачем компоненти в IaaS

Спрощена архітектура IaaS є фундаментальною моделлю розгортання, яка демонструє, як обчислювальні ресурси віртуалізуються та надаються користувачам на базі фізичного обладнання. Ця архітектура, розгорнута на фізичних серверах, функціонально ідентична архітектурі системи віртуалізації, що є ключовою технологією, яка лежить в основі IaaS.

Кожна гостьова машина, що надається кінцевому користувачеві IaaS, є динамічною комбінацією та конфігурацією ресурсів, виділених з цих трьох вузлів. Таким чином, ця архітектура забезпечує необхідну гнучкість та масштабованість, дозволяючи користувачам створювати та змінювати свої ВМ відповідно до індивідуальних потреб.

Рисунок 2.2 демонструє спрощену архітектуру IaaS. Ця інфраструктура розгортається на фізичних серверах і інтегрує три основні компоненти: мережевий вузол, обчислювальний вузол та вузол зберігання. Кожна гостьова машина, що надається користувачеві IaaS, є певною комбінацією цих трьох вузлів.

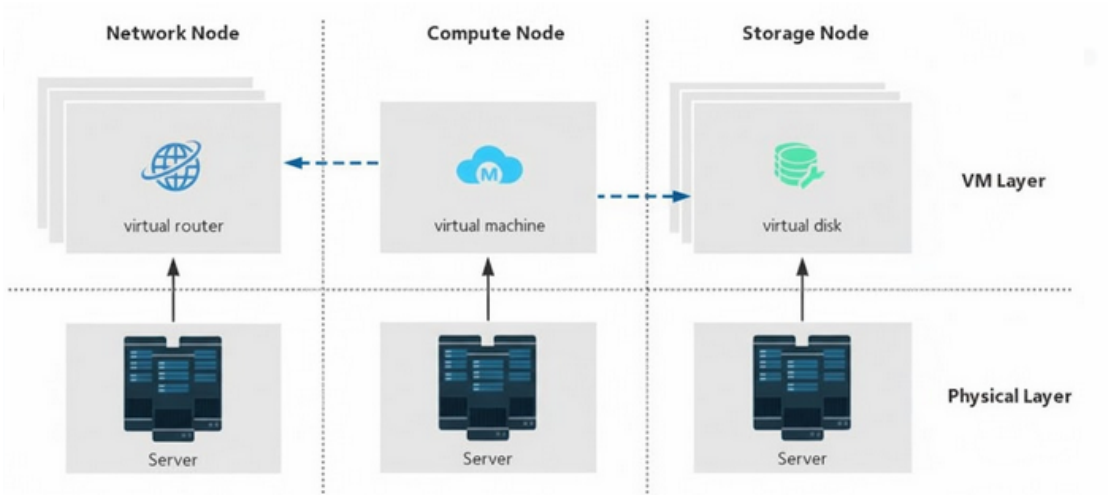

Рис. 2.2. Спрощена архітектура IaaS

Ця архітектура суттєво схожа на загальну архітектуру системи віртуалізації, представлену на рис. 2.3. Це підкреслює, що віртуалізація є ключовою технологією, що лежить в основі IaaS, оскільки гостьові машини в

IaaS є фактично віртуальними машинами, які функціонують на фізичних серверах провайдера.

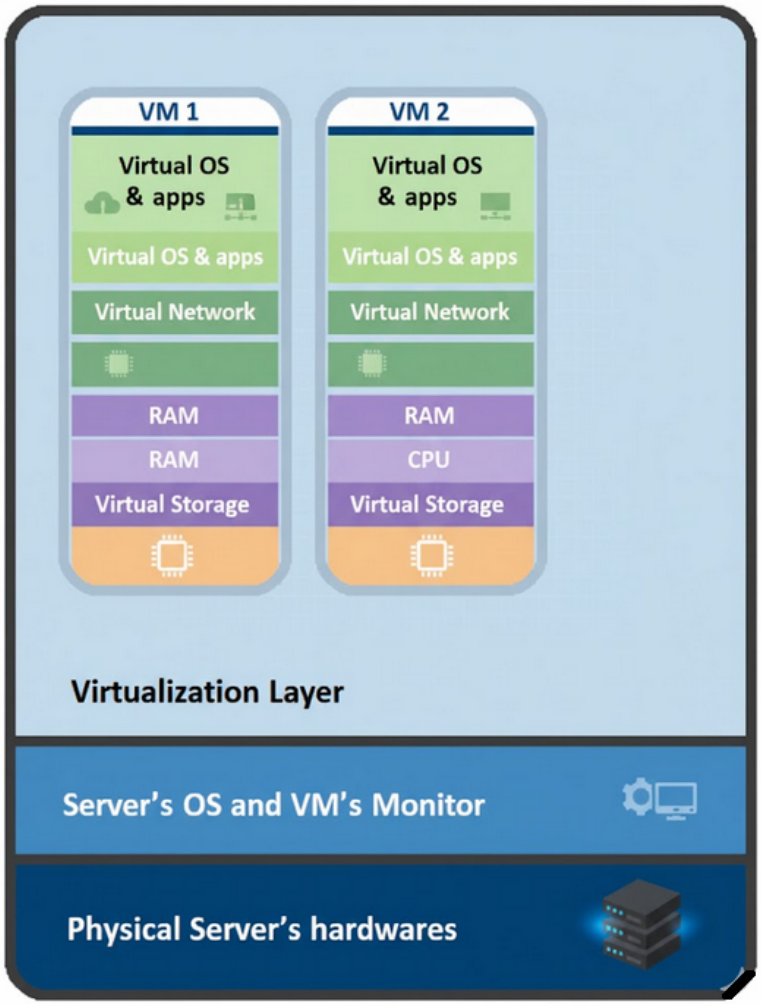

Рис. 2.3. Загальна архітектура віртуалізації

Віртуалізація — це метод розподілу ресурсів одного мейнфрейма, що функціонує на єдиній апаратній платформі, на менші, незалежні логічні екземпляри. Цей підхід забезпечує підвищення ефективності та зниження операційних зусиль з технічного обслуговування. У системі віртуалізації кожен користувач може обрати власну ОС, яка працює на його ВМ. Кілька ВМ можуть співіснувати на одному фізичному комп'ютері, причому кожна ВМ теоретично володіє всіма функціями реального комп'ютера. Хостова ОС є єдиним елементом, який безпосередньо взаємодіє з фізичним комп'ютерним апаратним забезпеченням.

Дві суттєві переваги віртуалізації включають:

1. Спільне використання ресурсів.

Одне фізичне апаратне забезпечення спільно використовується множиною ВМ, що максимізує ефективність використання апаратних ресурсів.

2. Ізоляція.

Всі ВМ у межах однієї системи віртуалізації ізольовані одна від одної, що робить їх невидимими одна для одної. Таким чином, програми, що виконуються на одній ВМ, не мають доступу до програм, які працюють на інших ВМ.

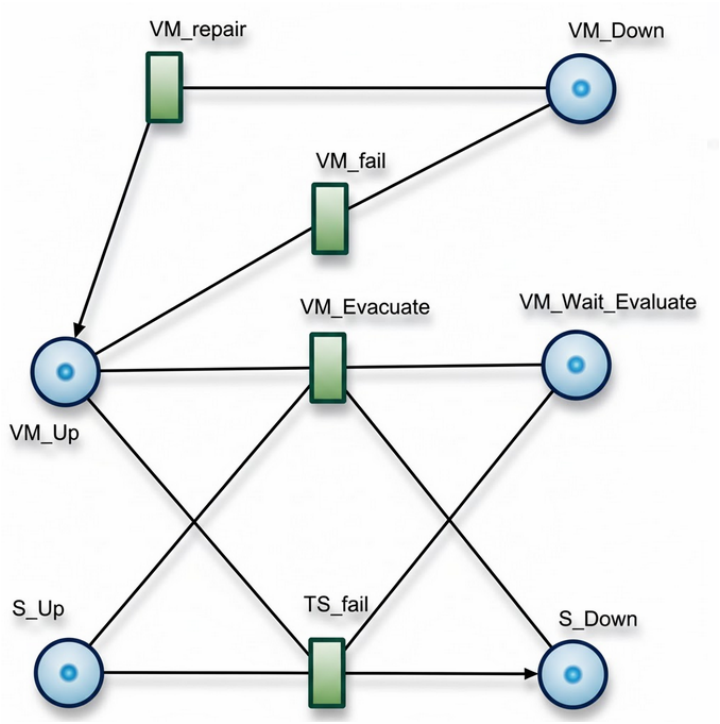

Рис. 2.4. Модель надійності серверів віртуалізації та віртуальних машин

Рис. 2.4 ілюструє модель надійності системи віртуалізації, що складається з кількох фізичних серверів та ВМ. Збої TS_fail та VM_fail являють собою помилкові події, що виникають відповідно на серверах та у ВМ. Ці збої можуть переривати безперервність використання ВМ з боку

користувача, оскільки системі потрібен додатковий час для відновлення або повного перезавантаження, що негативно впливає на надійність послуги.

У сегменті персональних комп'ютерів технологія віртуалізації, інтегрована в процесори, називається AMD-V для процесорів AMD та VT-x для процесорів Intel. Тільки процесори, які підтримують ці функції, можуть розміщувати на собі ВМ. Ці функції зазвичай вмикаються/вимикаються у вкладці налаштувань CPU в BIOS. Крім того, для забезпечення функціонування високопродуктивних обчислювальних (HPC) ВМ, процесори також мають підтримувати функцію Input-Output Memory Management Unit (IOMMU). Ця функція активується/деактивується у налаштуваннях BIOS, як правило, у вкладці Northbridge.

2.3. Архітектурний аналіз моніторів віртуальних машин та методика розгортання імітаційного середовища IaaS

У середовищі віртуалізації повна ізоляція віртуальних машин (ВМ) одна від одної забезпечується монітором віртуальних машин (VMM). VMM функціонує як інтерфейс між ВМ та фізичним апаратним забезпеченням, контролюючи доступ та використання ВМ цих апаратних ресурсів. VMM керує видимими ресурсами, які включають інформацію про стан ВМ, використання CPU, RAM та базу даних. Важливо зазначити, що сучасні процесори також мають невидимі ресурси (наприклад, спільна шина, спільна пам'ять, спільний кеш), які можуть значно впливати на поведінку системи за різних умов навантаження.

2.3.1 Класифікація та архітектура гіпервізорів

VMM або гіпервізор, класифікується на два основні типи:

- Гіпервізор типу 1 (Bare-Metal Hypervisor) - функціонує безпосередньо на фізичному апаратному забезпеченні та виконує роль повноцінної еквівалентної операційної системи (ОС).

- Гіпервізор типу 2 (Hosted Hypervisor) - працює як додаток на хостовій ОС і поводиться подібно до будь-якої іншої окремої програми.

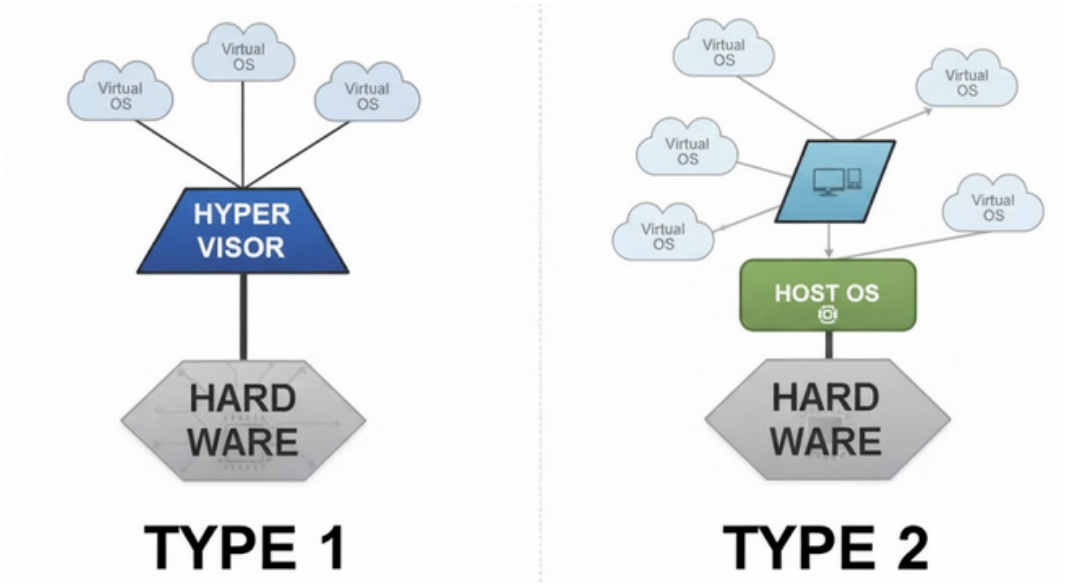

Рис. 2.5. Гіпервізор Типу 1 та Типу 2

У цьому дослідженні використовуються технології віртуалізації Xen Project (гіпервізор типу 1) та VirtualBox (гіпервізор типу 2), оскільки вони є найбільш типовими технологіями, що застосовуються провідними постачальниками хмарних послуг. Xen Project використовується, зокрема, Amazon EC2 — найбільшим провайдером хмарних послуг у галузі. Архітектура VirtualBox, у свою чергу, дуже схожа на архітектуру VMware vCloud, який є провідним постачальником, що використовує технологію гіпервізора типу 2.

2.3.2. Гіпервізор Xen

Гіпервізор Xen — це відкритий гіпервізор типу 1, який широко застосовується у високотехнологічних секторах, таких як віртуалізація серверів/робочих станцій, IaaS, вбудовані системи та апаратні пристрої. Проект Xen є найпоширенішою системою віртуалізації серед сучасних провайдерів хмарних послуг.

Xen підтримує два типи гостьових ВМ:

- Паравіртуалізовані гостьові ВМ (Para-virtualized, PV Guest) - техніка віртуалізації, представлена проектом Xen, яка не вимагає апаратних розширень віртуалізації (AMD-V або VT-x) від фізичного CPU.

- Апаратно-допоміжні віртуальні машини (Hardware-assisted Virtual Machine, HVM Guests) - традиційна техніка віртуалізації, яка вимагає апаратних розширень віртуалізації (AMD-V або VT-x) від фізичного CPU.

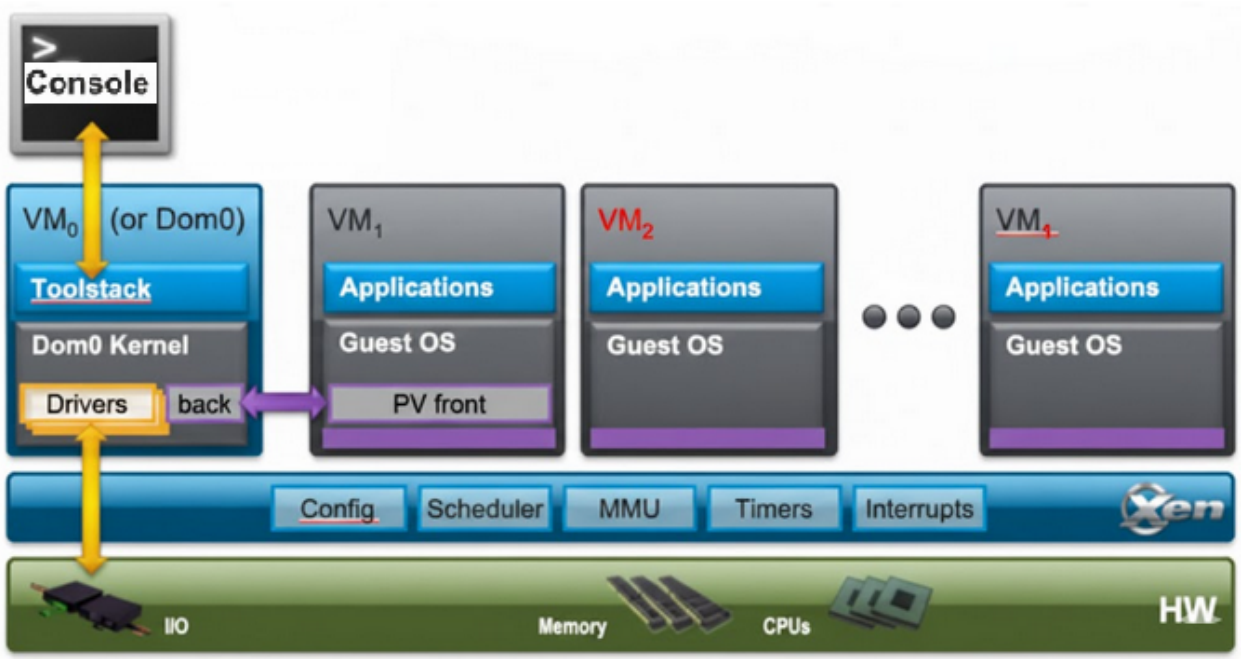

Рис. 2.6. Архітектура проекту Xen

Згідно з рис. 2.6, гіпервізор Xen працює безпосередньо на фізичному апаратному забезпеченні та керує CPU та пам'яттю, але не функціями вводу-виводу. Функції вводу-виводу обробляються доменом 0 (Dom0), який є спеціалізованою ВМ, що завжди функціонує на хості Xen за замовчуванням. Dom0 є найбільш критичною ВМ у системі віртуалізації Xen, оскільки це єдина ВМ, яка може безпосередньо отримувати доступ до апаратного забезпечення. Dom0 містить набір інструментів для керування створенням, видаленням та конфігурацією ВМ, доступ до яких користувачі можуть отримати через інтерфейс командного рядка (CLI), включений до проекту

Xen. Інші ВМ, окрім Dom0, називаються гостьовими доменами або непривілейованими доменами (DomU).

Переваги використання гіпервізора Xen включають: сумісність з більшістю ОС на базі Linux, NetBSD та Solaris; наявність драйверів пристроїв, що працюють всередині ВМ, що забезпечує ізоляцію драйверів (у разі збою драйвера, ця ВМ може перезавантажитися, не впливаючи на інші ВМ); а також повну підтримку PV гостьових ВМ, що дозволяє PV-екземплярам працювати з такою ж швидкістю, як і HVM-екземпляри, при нормальному обчислювальному навантаженні.

Середовище віртуалізації гіпервізора Xen, використане в цій роботі, включає одну фізичну хост-машину (ОС: Ubuntu 14.04 LTS) та дві віртуальні машини (ВМ 1: Ubuntu 14.04 LTS та ВМ 2: Windows).

Таблиця 2.1.

Специфікації віртуальних машин

Компонент	**Хост-машина**	**ВМ 1**	**ВМ 2**
ОЗУ (RAM)	16 ГБ	4 Гб	4 ГБ
CPU	AMD FX8350, 4.0 ГГц, 8 ядер	AMD FX8350, 4.0 ГГц, 1 ядро	AMD FX8350, 4.0 ГГц, 1 ядро
Жорсткий диск	50 ГБ	12 ГБ	9 ГБ
ОС	Ubuntu 14.04 LTS x64	Ubuntu 14.04 LTS x64	Windows 10 x64

Реалізація середовища віртуалізації Xen Project здійснювалася за наступними послідовними кроками:

Крок 1: Встановлення хостової ОС. Встановлення Ubuntu на фізичну машину з використанням конфігурації менеджера логічних томів (LVM).

Крок 2: Встановлення гіпервізора. Встановлення Xen Project на хостову ОС за допомогою команди: sudo apt-get install xen-hypervisor-amd64

Крок 3: Перевірка встановлення. Перевірка успішності встановлення гіпервізора Xen.

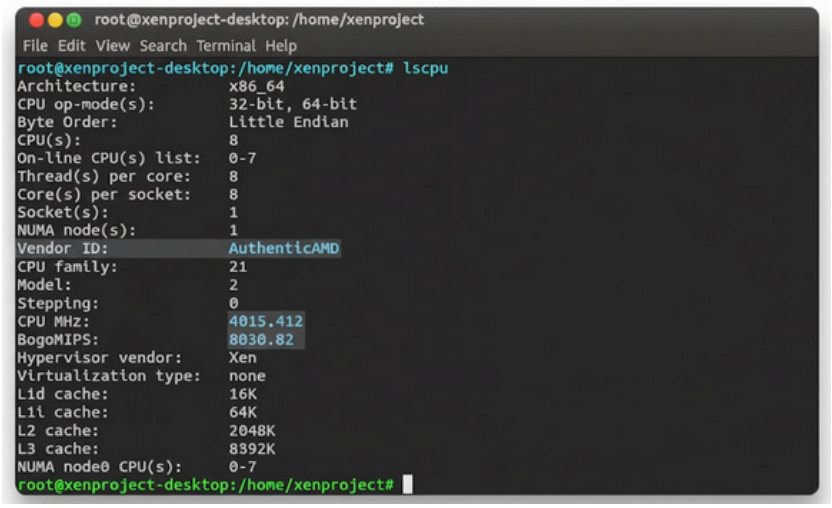

Рис. 2.7. Список специфікацій CPU

На рисунку 2.7 вказано, що постачальник гіпервізора — Xen, що підтверджує роботу Dom0. Звертається увага на те, що гіпервізор Xen приховує прапор віртуалізації, змушуючи ВМ ідентифікувати себе як фізичні машини, тому тип віртуалізації не вказується.

Крок 4: Підтвердження підтримки апаратного забезпечення. Перевірка підтримки апаратної технології віртуалізації за допомогою команди xl dmesg (яка працює лише з гіпервізором Xen).

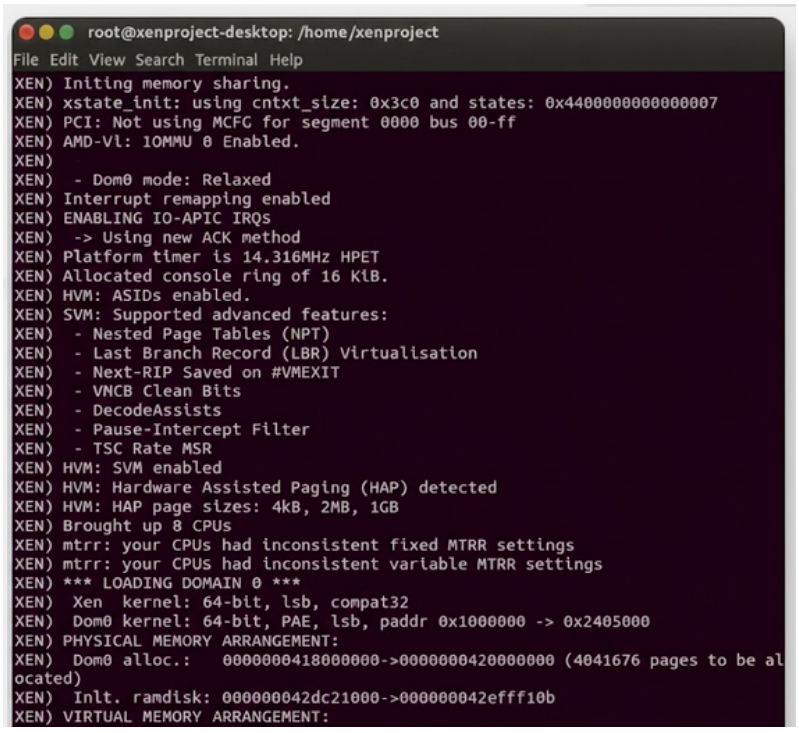

Рис. 2.8. Архітектура Xen та ядро у файлі журналу dmesg

Рис. 2.8 підтверджує, що віртуалізація вводу-виводу, SVM, AMD-Vi та IOMMU увімкнені, що є необхідними прапорами для роботи ВМ. Також показано, що Dom0 виділено 8 віртуальних CPU з фізичного апаратного забезпечення.

Крок 5: Налаштування мережевого інтерфейсу. Конфігурація мережевого інтерфейсу в /etc/network/interfaces.

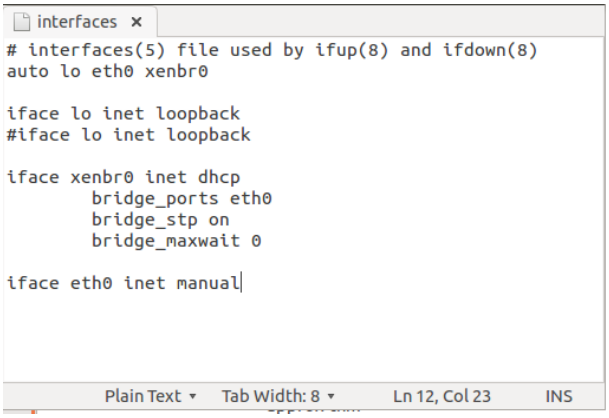

Рис. 2.9. Налаштування мережевого інтерфейсу в Xen Virtualization

Крок 6: Налаштування ВМ 1 (Ubuntu). Створення логічного тому (LV) для зберігання віртуального жорсткого диска: sudo lvcreate -L 12G -n ubuntu-hvm /dev/ubuntu-vg Створення конфігураційного файлу гостьової машини (/etc/xen/ubuntu-hvm.cfg).

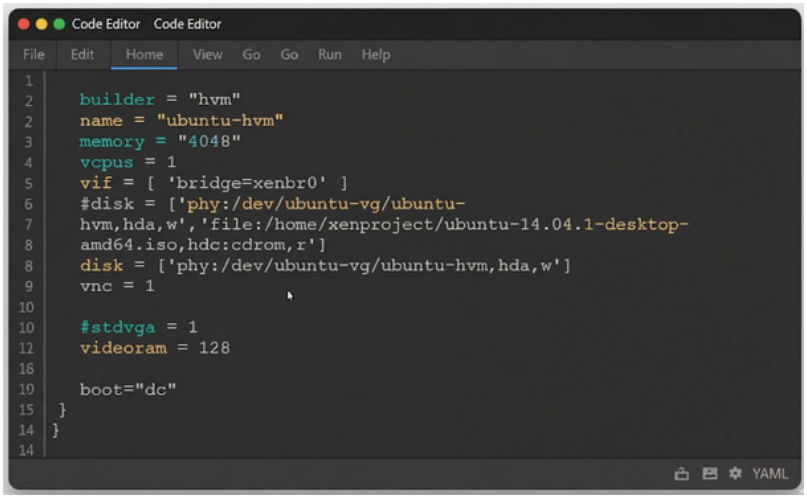

Рис. 2.10. Конфігураційний файл віртуальної машини Ubuntu

У цьому файлі гостьова ВМ має ім'я ubuntu-hvm, тип HVM, виділено 4048 МБ віртуальної пам'яті та 1 віртуальний CPU. Шлях до завантажувального диска змінюється з ISO-файлу (для встановлення) на каталог логічного тому (після встановлення).

Крок 7: Налаштування ВМ 2 (Windows). Аналогічно, створення LV та конфігураційного файлу.

Крок 8: Перевірка логічних томів. Використання команд pvdisplay, vgdisplay та lvdisplay для перевірки логічних томів (LV) у розділі LVM хоста.

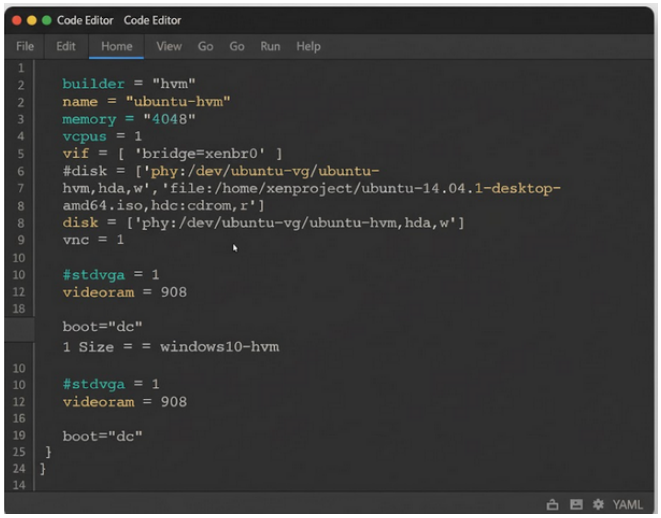

Рис. 2.11. Відображення логічних томів ВМ у розділі LVM хоста

Рис. 2.11 підтверджує, що LV ubuntu-hvm (12 ГБ) та windows10-hvm (9 ГБ) розташовані під групою томів ubuntu-vg системи Xen.

Крок 9: Створення та запуск ВМ. Створення ВМ на гіпервізорі Xen за допомогою команди create.

Крок 10: Знищення ВМ. Для вимкнення гіпервізора Xen необхідно спочатку знищити всі ВМ за допомогою команди xl destroy.

2.3.3. Використання Oracle VirtualBox

Oracle VirtualBox — це безкоштовне програмне забезпечення для віртуалізації типу 2 з відкритим кодом, яке вимагає підтримки апаратної віртуалізації (VT-x або AMD-V).

Завдяки інтуїтивно зрозумілому графічному інтерфейсу користувача (GUI), VirtualBox став одним із найпоширеніших середовищ віртуалізації для повсякденного використання. Його GUI надає базові функції для керування ВМ (створення, запуск, призупинення, зупинка). VirtualBox підтримує широкий спектр гостьових ОС (Windows, Linux, MacOS, OpenSolaris, FreeBSD) та три основні формати віртуальних жорстких дисків: VDI (стандарт VirtualBox), VMDK (VMware) та VHD (Windows Virtual PC).

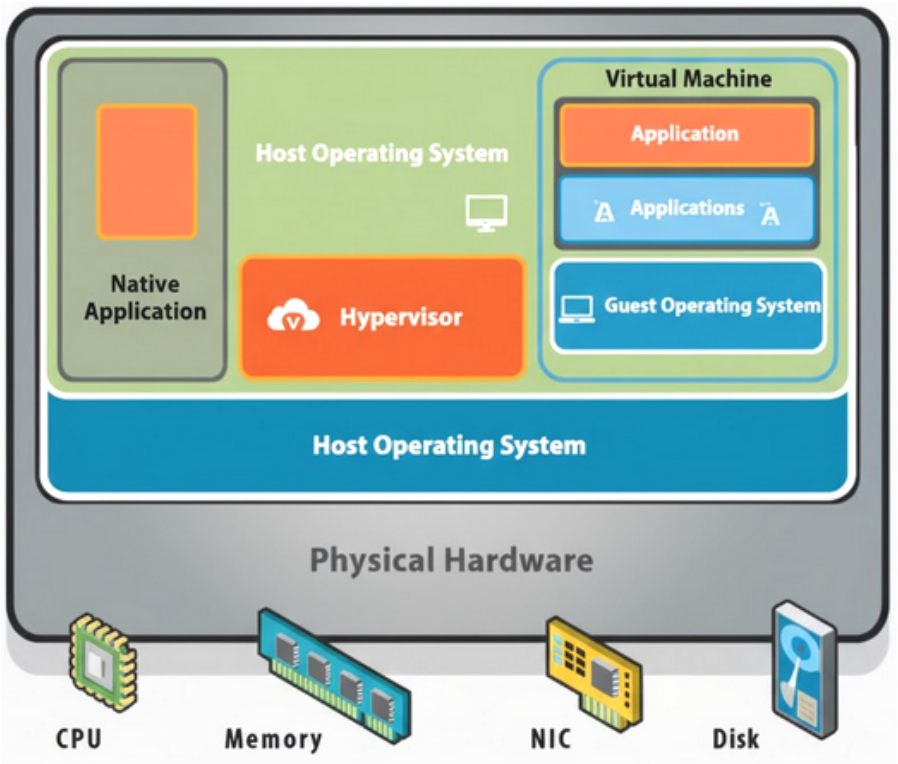

Рис. 2.12. Архітектура VirtualBox та VMware

Рисунок 2.12 демонструє, що VirtualBox та VMware мають ідентичну архітектуру, де гіпервізор розглядається як програма, що працює на хостовій ОС. Оскільки ці інструменти є загальнодоступними та популярними серед звичайних користувачів, детальний опис процесу створення віртуалізованого середовища VirtualBox у цій дисертації опускається. Система віртуалізації, створена за допомогою VirtualBox, матиме ті ж самі специфікації, що зазначені у таблиці 2.1.

2.4. Методологічні аспекти збору та обробки журналів подій про збої для емпіричних досліджень надійності системи

У контексті науки про дані (Data Science) емпіричні дані, зібрані під час функціонування системи, надають критично важливу інформацію щодо фактичних або потенційних помилкових/відмовних поведінок. Ця інформація використовується для перевірки гіпотез, сформульованих в аналітичних моделях, та для діагностики і прогнозування збоїв.

Реєстрація (Logging) являє собою процес фіксації системних та мережевих активностей і підтримання записаних даних у хронологічному порядку. Реєстраційні дані (або просто "реєстрація") є необхідною передумовою для розуміння діяльності складних систем, особливо тих, що мають мінімальну взаємодію з користувачами, як-от сервери.

Журнали подій (Event Logs) — це набір записів, що генеруються протягом сеансу виконання системи. Вони слугують як аудиторський слід, який у подальшому використовується для аналізу системної активності та діагностування проблем у разі їх виникнення.

У сфері обчислювальної техніки, подія визначається як дія або випадок, що виникає синхронно з програмним забезпеченням реєстрації подій. Події охоплюють як нормальну, так і помилкову/відмовну активність. Комп'ютерні події можуть бути згенеровані або ініційовані системою, користувачами чи іншими механізмами.

Традиційно журнали системних подій стандартизуються відповідно до специфікації syslog, як проілюстровано у таблиці 2.2.

Таблиця 2.2.

Фрагмент прикладного набору традиційного журналу подій

Час	Джерело	Рівень
5:57:25 PM	udisksd	LG-notice
5:57:25 PM	NetworkManager	LH-info
5:57:32 PM	NetworkManager	LG-info

Час	Джерело	Рівень
5:57:50 PM	kernel	LG-warning
5:57:50 PM	kernel	LG-notice
5:57:52 PM	dbus	LG-notice
5:57:52 PM	kernel	LG-warning
5:50:03 PM	kernel	LH-debug
5:50:15 PM	IPSec	WG-Information
5:50:35 PM	pulseaudio	LG-err
...	...	...

Проте, в рамках даної роботи журнали систем розглядаються з альтернативної перспективи, подібно до підходу Flow-net Logging, що візуально демонструється на рис. 2.13. Перевага цього підходу полягає в тому, що він включає реєстрацію конкретних взаємозв'язків між подіями, що забезпечує високу ефективність для зворотної діагностики (back-tracing) у разі виникнення помилок.

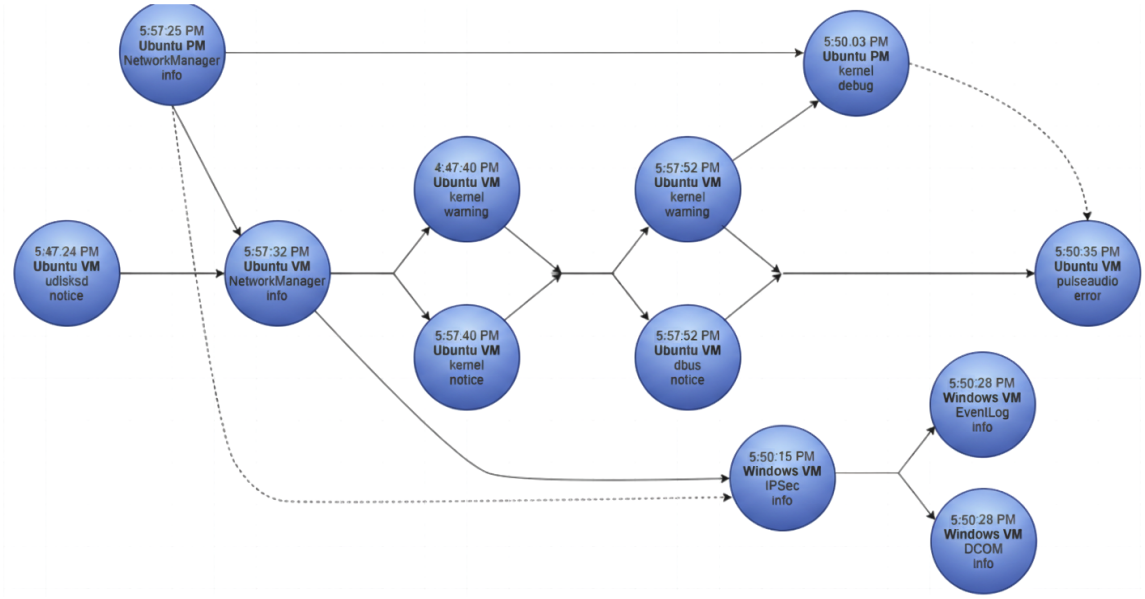

Рис. 2.13. Підхід Flow-net Logs

2.4.1. Журнали подій Windows

Журнали подій Windows надають стандартизований, централізований механізм для програмного забезпечення та системних додатків для реєстрації важливих програмних та апаратних подій. Операційна система Windows,

додатки та інші системні служби записують ці важливі події, як-от умови низького рівня пам'яті або надмірні спроби доступу до диска. У подальшому, у разі виникнення помилки чи збою, системний адміністратор використовує ці події для ідентифікації першопричини збою, спроби відновлення втрачених даних та запобігання повторенню помилки [7].

У ОС Windows реєстрація подій реалізована як системна служба, що працює у фоновому режимі та очікує ініціюючих активностей для початку запису. Служба журналу подій Windows зберігає зареєстровані події у файлах із розширенням .evt. Існують три типи файлів журналу: журнал додатків, журнал безпеки та системний журнал. Для цілей дослідження доступності та надійності системи ця дисертація зосереджується виключно на системному журналі.

Системні журнали подій класифікуються відповідно до п'яти рівнів серйозності, як описано у таблиці 2.3.

Таблиця 2.3.

Рівні серйозності журналу подій Windows

Рівень серйозності	**Опис**	**Приклад**
Помилка (Error)	Подія, що вказує на **значну проблему**, наприклад, втрату функціональності або даних.	Невдача завантаження служби під час запуску системи реєструється як помилка.
Попередження (Warning)	Подія, яка не є критичною негайно, але **вказує на можливу майбутню проблему**.	Низький рівень дискового простору реєструється як попередження. Якщо додаток може відновитися без втрати функціональності, подія класифікується як попередження.
Інформація (Information)	Подія, що описує **успішну роботу** додатка, драйвера або служби.	Успішне завантаження мережевого драйвера може бути зареєстровано як інформаційна подія.
Успішний аудит (Success Audit)	Подія, що фіксує **успішну спробу аудиту безпеки**.	Успішна спроба користувача увійти в систему реєструється як успішний аудит.
Невдалий аудит (Failure Audit)	Подія, що фіксує **невдалу спробу аудиту безпеки**.	Невдала спроба користувача отримати доступ до мережевого диска реєструється як невдалий аудит.

Структура файлу журналу подій, розроблена Microsoft, включає дві основні структури:

```
_EVENTLOGHEADER


_EVENTLOGRECORD 1
_EVENTLOGRECORD 2
...
_EVENTLOGRECORD N
<BLANK SPACE>
```

Заголовок файлу журналу (EVENTLOGHEADER): Містить загальну інформацію про файл журналу.

```
typedef struct _EVENTLOGHEADER {
  ULONG HeaderSize;
  ULONG Signature;
  ULONG MajorVersion;
  ULONG MinorVersion;
  ULONG StartOffset;
  ULONG EndOffset;
  ULONG CurrentRecordNumber;
  ULONG OldestRecordNumber;
  ULONG MaxSize;
  ULONG Flags;
  ULONG Retention;
  ULONG EndHeaderSize;
} EVENTLOGHEADER, *PEVENTLOGHEADER;
```

Запис події (EVENTLOGRECORD): Містить детальну інформацію про кожен окремий запис події. Один файл .evt може містити N записів EVENTLOGRECORD, де N залежить від попередньо визначеного користувачем розміру файлу.

```
typedef struct _EVENTLOGRECORD {
  DWORD Length;
  DWORD Reserved;
  DWORD RecordNumber;
  DWORD TimeGenerated;
  DWORD TimeWritten;
  DWORD EventID;
  WORD  EventType;
  WORD  NumStrings;
  WORD  EventCategory;
  WORD  ReservedFlags;
  DWORD ClosingRecordNumber;
  DWORD StringOffset;
  DWORD UserSidLength;
  DWORD UserSidOffset;
  DWORD DataLength;
  DWORD DataOffset;
} EVENTLOGRECORD, *PEVENTLOGRECORD;
```

Незважаючи на те, що Windows надає вбудоване програмне забезпечення переглядач подій (Event Viewer) для перегляду журналів, багато значень атрибутів у EVENTLOGRECORD не відображаються у цьому інструменті. Після аналізу інших доступних інструментів (таких як Wevtutil або LogParser), було встановлено, що жоден з них не надає повної інформації про один EVENTLOGRECORD. Тому для цілей даного дослідження було розроблено власний конвертер evt2txt.cpp (вихідний код буде надано у Додатку А), призначений для перетворення формату файлу .evt у текстовий формат .txt за допомогою мови програмування C++.

Нижче наведено зразок журналу подій у текстовому форматі після конвертації:

```
HeaderSize: 48,
Signature: 1699505740,
MajorVersion: 1,
MinorVersion: 1,
StartOffset: 48,
EndOffset: 32436,
CurrentRecordNumber: 142,
OldestRecordNumber: 1,
MaxSize: 32476,
Flags: 0,
Retention: 0,
EndHeaderSize: 48,
Event Record ID: 0,
Length: 192,
Reserved: 1699505740,
RecordNumber: 1,
TimeGenerated In epoch: 1464232167,
TimeGenerated In local datetime: Wed May 25 20:09:27 2025,
TimeWritten In epoch: 1464232167,
TimeWritten In local datetime: Wed May 25 20:09:27 2025,
EventID in hexadecimal: 2147489657,
EventID:6009,
EventType: 4,
NumStrings: 4,
EventCategory: 0,
ReservedFlags: 0,
```

...

Згідно з інформацією заголовка, цей файл містить 142 записані події, із загальним розміром 32 476 байт. Перша зареєстрована подія (RecordNumber: 1) була записана 25 травня 2025 року о 20:09:27 службою EventLog, а її

значення EventType дорівнює 4. Відповідно до таблиці 2.4, це означає, що рівень серйозності цієї події — "Інформація".

Таблиця 2.4.

Довідкова таблиця значень EventType з MSDN

Значення EventType	**Рівень серйозності**
0	Успіх
1	Помилка
2	Попередження
4	Інформація
8	Успішний аудит
16	Невдалий аудит

2.4.2. Реєстрація системних подій у Linux

Операційна система Linux використовує системні демони syslogd (або syslog-ng) як служби реєстрації системних подій. Демон syslogd складається з двох основних системних компонентів: klogd та syslogd. Klogd відповідає за управління всіма процесами реєстрації подій, що походять від ядра (kernel), тоді як syslogd керує процесом реєстрації подій від прикладних програм (applications). Крім того, сторонні додатки можуть генерувати власні незалежні журнали.

Процес реєстрації подій у Linux відповідає стандарту syslog і фіксується за часом генерації. Отже, ця система зазвичай позбавлена здатності зберігати прямі взаємозв'язки (relationship) між окремими подіями.

У процесі реєстрації подій Linux інформація про ініційовану подію спочатку зберігається у буфері, після чого демон реєстрації подій періодично записує цю подію у файл журналу. Відповідно, часова мітка (timestamp) події не обов'язково відображає точний час фактичного виникнення події; вона позначає час, коли подія була згенерована та записана у файл.

Рівні серйозності подій у системі Ubuntu класифікуються відповідно до таблиці 2.5.

Таблиця 2.5.

Рівні серйозності в Ubuntu

Рівень серйозності	Ключове слово	Опис
Emergency	emerg	Система непридатна для використання.
Alert	alert	Необхідна негайна дія. Умова, яку слід негайно виправити (наприклад, пошкоджена системна база даних).
Critical	crit	Критичні умови, такі як апаратні помилки пристроїв.
Error	err	Умови помилки.
Warning	warning	Умови попередження.
Notice	notice	Нормальні, але значущі умови. Умови, які не є помилковими, але можуть вимагати спеціальної обробки.
Informational	info	Інформаційні повідомлення.
Debug	debug	Повідомлення рівня налагодження. Містять інформацію, корисну, як правило, лише при налагодженні програми.

Структура події в Ubuntu може варіюватися залежно від формату, визначеного користувачем у конфігураційному файлі /etc/rsyslog.conf. Нижче наведено приклад модифікованого шаблону та відповідний вихідний формат запису:

Лістинг 2.1. Модифікований шаблон (strtpl)

```
$template strtpl,"PRI: %pri-text%,\nSeverity: %syslogseveritytext%,\
nFacility: %syslogfacility-text%,\nTimeGenerated:
%timegenerated%,\nTimeReported: %timereported%,\nHostname:
%HOSTNAME%,\nProgramName: %programname%,\nFromHost: %fromhost%,\nTAG:
%syslogtag%,\nInfoUnitType: %iut%,\nMsg: %msg%,\nRawMsg:
%rawmsg%\nProtocolVersion: %protocol-version%,\nStructuredData:
%structured-data%,\nAppName: %app-name%,\nProcid: %procid%,\nMsgid:
%msgid%,\nInputname: %inputname%\n\n"
```

Лістинг 2.2. Фрагмент вихідного запису події

```
PRI: daemon.info,
Severity: info,
Facility: daemon,
TimeGenerated: May 26 03:43:40,
TimeReported: May 26 03:43:40,
Hostname: UbuntuHost,
ProgramName: NetworkManager,
FromHost: UbuntuHost,
TAG: NetworkManager[698]:,
InfoUnitType: 1,
```

Параметри, включені у наведений вище запис події Linux, є функціонально аналогічними параметрам, що містяться у записах подій Windows. Отже, подальший детальний розгляд цієї структури в даному контексті не є необхідним.

Висновки до розділу

Другий розділ присвячено дослідженню принципів віртуалізації у хмарному середовищі та методам реалізації інфраструктури як послуги (IaaS). Проведено детальний аналіз архітектурних рішень моніторів віртуальних машин (VMM) і гіпервізорів, таких як Xen та Oracle VirtualBox, визначено їхні технічні особливості, рівні абстракції та вплив на продуктивність хмарних систем.

Запропоновано методику побудови імітаційного середовища для дослідження процесів віртуалізації збоїв, що дає змогу контролювати взаємодію між компонентами віртуальних машин і системними подіями. Розроблено процедуру збору емпіричних даних із системних журналів Windows та Linux, адаптовану для подальшого використання в процесі навчання нейронних мереж.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ТА МЕТОДІВ ВІРТУАЛІЗАЦІЇ ПОДІЙ ТА ЗБОЇВ В ХМАРНИХ ПЛАТФОРМАХ

3.1. Симуляція поведінки типових користувачів для генерації емпіричних даних

Для забезпечення репрезентативності та валідності зібраних даних системних подій, необхідних для навчання нейронної мережі, критично важливим є відтворення реальних умов експлуатації. Це досягається шляхом симуляції поведінки типових користувачів (автокористувачів) у віртуалізованому середовищі. Метою є отримання реалістичних журналів подій, які відображатимуть типові сценарії використання системи.

З цією метою було розроблено спрощений алгоритм імітації, який моделює стандартні дії користувачів. Ці дії охоплюють:

- Запуск та взаємодія з програмним забезпеченням;
- Операції з файловою системою (створення, читання, модифікація);
- Використання мережевих ресурсів та комунікаційні активності.

Зазначені дії автоматично виконуються за допомогою скриптових механізмів, запущених на гостьових віртуальних машинах.

3.1.1. Моделювання сценаріїв використання

Концептуальна основа симуляції базується на моделюванні основних сценаріїв взаємодії користувача із системою, що відображено на діаграмі варіантів використання.

Діаграма ілюструє, як типовий користувач використовує хмарну машину для виконання різноманітних офісних та мультимедійних завдань, а також включає один варіант використання, пов'язаний із завершенням сеансу. Усі перелічені дії (відкриття програм) ініціюються типовим користувачем і в результаті можуть призводити до дії " Randomly close

application ". Це вказує на те, що симуляція передбачає модель, де будь-яка активна програма може бути випадково або імітовано завершена. Ця діаграма слугує основою для симуляції поведінки користувача з метою генерації реалістичних журналів системних подій.

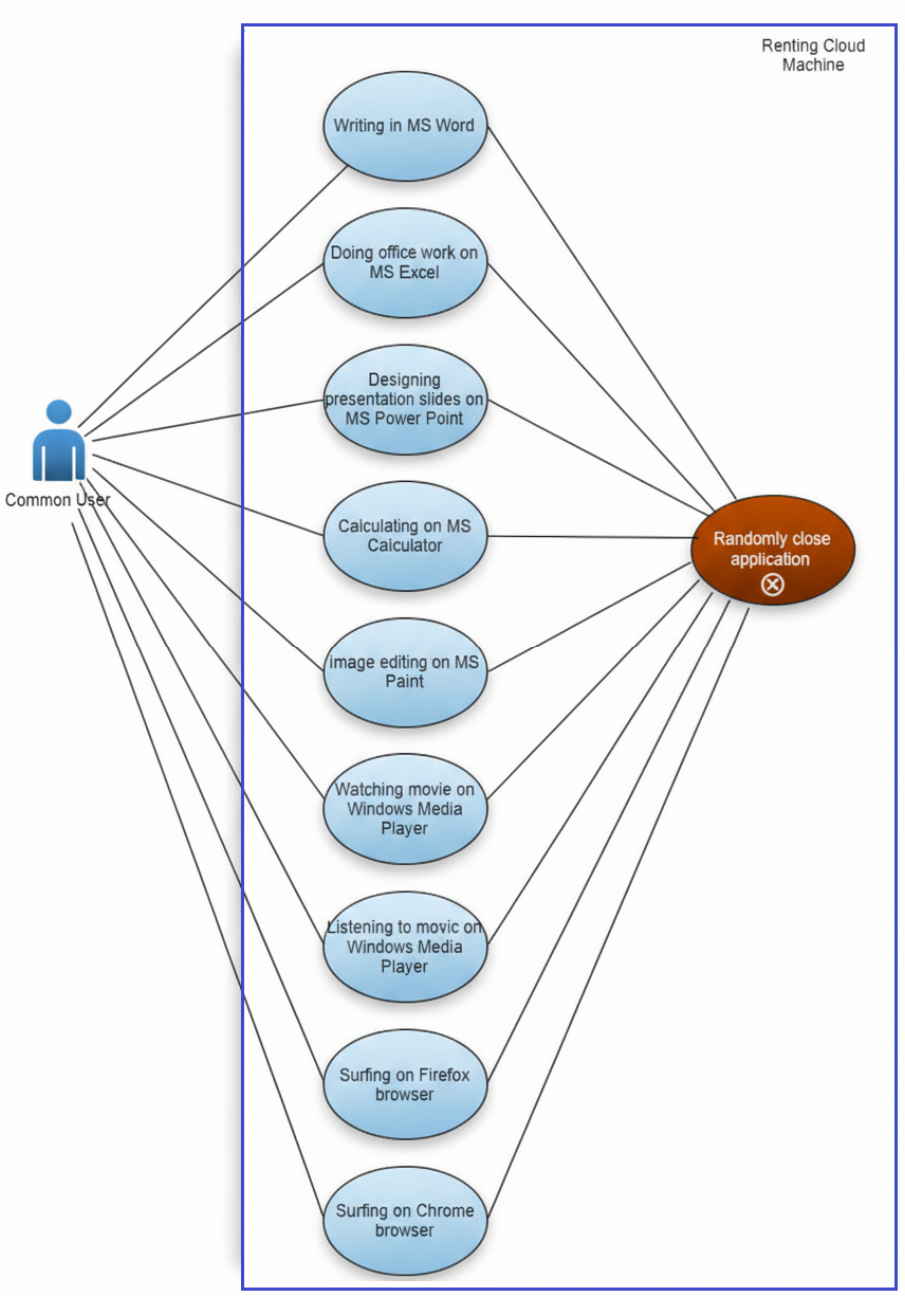

Рис. 3.1. Діаграма варіантів використання типового користувача

Діаграма на рис. 3.1 візуалізує ключові категорії активностей, які може ініціювати типовий користувач. Ці категорії включають: маніпуляції з файлами, мережеві операції та ініціалізацію програмних додатків.

3.1.2. Динаміка симуляційної активності

Послідовність виконання імітаційних дій визначається діаграмою активності симуляції автокористувача, яка забезпечує логічне та послідовне відтворення типових робочих процесів.

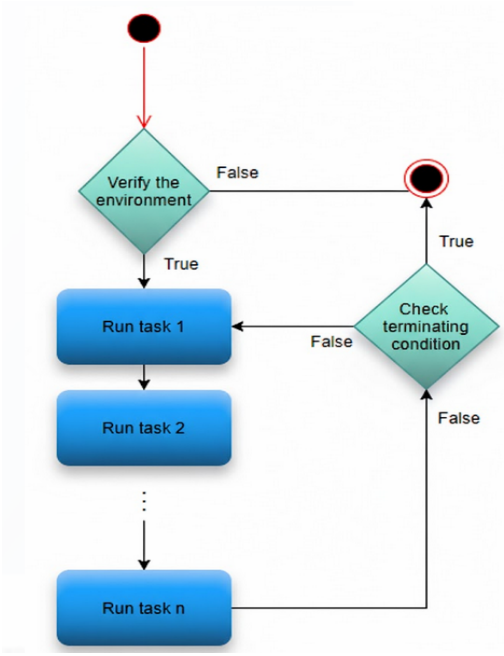

Рис. 3.2. Діаграма активності симуляції автокористувача

Рисунок 3.2 ілюструє послідовність та логічний потік дій, що виконуються під час симуляції. Цей потік включає дискретні кроки: запуск програмного забезпечення, виконання операцій із файлами та мережева активність, що є критично важливим для генерації комплексних і достовірних журналів системних подій.

3.2. Методика збору емпіричних даних системних подій про збої

Для забезпечення емпіричної бази досліджень була розроблена стратегія автоматизованого збору журналів подій безпосередньо з віртуальних машин (ВМ). Ця стратегія націлена на реєстрацію повного спектру системних та мережевих подій, які генеруються в процесі

функціонування імітаційного середовища. Зібрані журнали слугують первинним джерелом даних для подальшого аналізу.

Процедура збору даних включає послідовну реалізацію чотирьох ключових етапів:

1. Ініціалізація віртуальних машин.

Здійснюється запуск та забезпечення готовності віртуальних машин у середовищі віртуалізації (наприклад, Xen або VirtualBox) до виконання робочих навантажень.

2. Емуляція поведінки користувачів.

На активних ВМ виконуються попередньо розроблені скрипти (авто-макроси), які забезпечують симуляцію типових дій користувачів (як описано у розділі 3.1). Це необхідно для генерації реалістичного та репрезентативного набору системних подій.

3. Агрегація журналів подій.

Здійснюється автоматичне збирання файлів журналів подій (evt у Windows, syslog у Linux) з кожної функціонуючої ВМ.

4. Персистенція даних.

Зібрані журнали подій зберігаються у файлах (із застосуванням необхідного формату, наприклад, після конвертації у txt) для подальшої обробки та аналізу в рамках наукового дослідження.

Ця методика забезпечує контрольоване та масштабоване отримання достовірних даних для подальшого застосування в моделях машинного навчання або майнінгу процесів.

3.3. Мережева модель нелінійної авторегресії з зовнішнім входом для прогнозування подій відмов та збоїв в хмарному середовищі

Моделювання часових рядів із застосуванням нейронних мереж (НМ) являє собою інноваційну методологію для виявлення та прогнозування послідовних подій. Подібно до методу майнінгу епізодів, описаного в

першому розділі, модель НМ часових рядів здатна ідентифікувати всі часті епізоди, враховуючи заданий набір послідовностей подій та максимальну порогову довжину підтримки. Однак, на відміну від традиційного майнінгу епізодів, НМ часових рядів також може вивчати правила виникнення частого епізоду та прогнозувати наступну подію на основі виявлених закономірностей.

У середовищі MATLAB моделі нейронних мереж часових рядів класифікуються на три основні типи:

- Нелінійна авторегресія (NAR net) - прогнозує часовий ряд y(t) на основі ny попередніх значень того ж ряду y(t). Функція для NAR net визначається як:

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y))$$

- Нелінійний вхід-вихід (Nonlinear Input-Output) - прогнозує часовий ряд y(t) на основі n_x попередніх значень вхідного ряду x(t). Функція визначається як:

$$y(t) = f(x(t-1), x(t-2), \ldots, x(t-n_x))$$

- Нелінійна авторегресія із зовнішнім входом (NARX net) - прогнозує часовий ряд y(t) на основі n_y попередніх значень ряду y(t) та nx попередніх значень зовнішнього ряду x(t) (де n_x може дорівнювати n_y). Функція для NARX net визначається як:

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), x(t-1), x(t-2), \ldots, x(t-n_x))$$

З огляду на те, що модель NARX net використовує найбільшу кількість вхідних векторів, вона, очевидно, забезпечує потенційно найвищу точність результатів. Відповідно, у даному дослідженні для подальших експериментів використовується саме модель NARX net.

Технічно, NARX net є різновидом нейронної мережі прямого поширення (feed forward neural network), яка приймає на вхід послідовність подій. Довжина вхідної послідовності визначається двома параметрами: Затримка входу (Input-Delay, ID), що відповідає значенню n_x, та затримка зворотного зв'язку (Feedback-Delay, FD), що відповідає значенню n_y. Наприклад, NARX net з ID=15 та FD=30, прогнозуючи наступне значення y(t), використовуватиме вхідну послідовність, що включає 15 останніх значень x(t) (від xt−15 до xt−1) і 30 останніх значень y(t) (від yt−30 до yt−1).

3.3.1. Визначення оптимальних затримок

Вибір адекватних значень для ID та FD є критично важливим для підвищення точності прогнозування моделі NARX net. Серед широко використовуваних для цієї мети методів є обчислення взаємної кореляції (cross-correlation) та автокореляції (auto-correlation).

Взаємна кореляція застосовується між двома дискретними часовими рядами. Вище значення взаємної кореляції свідчить про більшу подібність між двома рядами. Цей метод використовується для визначення оптимального значення ID між x(t) та y(t).

$$x_{corr}(X, Y, lag) = \sum_{n=0}^{n-1} X[n] \times Y[n]$$

У проведених експериментах, після серії ітерацій, було встановлено, що зсув ряду y(t) відносно x(t) на 30 стовпців (lag=30) забезпечує найбільш значущий результат xcorr(X,Y,30)=4.38. Відповідно, значення ID було обрано як 30.

Автокореляція обчислюється аналогічно до взаємної кореляції, але застосовується між одним часовим рядом і ним самим зі зміщенням у часі, визначеним значенням lag. Цей метод використовується для визначення оптимального значення FD.

$$auto_{corr}(Y, lag) = \sum_{n=0}^{n-1} Y[n] \times Y[n + lag]$$

Експериментально було визначено, що найбільш значуще значення автокореляції становить autocorr(Y,30) = 12.75. Таким чином, значення FD також обрано як 30.

3.3.2. Конфігурація мережі NARX net

Вхідні дані для моделі NARX net, використаної у цьому експерименті, структуровані як у таблиці 3.1.

Таблиця 3.1.

Вхідні дані NARX Net

ProgramName / назва програми	**SeverityLevel / рівень серйозності**
udisksd	LG-notice
NetworkManager	LH-info
NetworkManager	LG-info
kernel	LG-warning
kernel	LG-notice
dbus	LG-notice
kernel	LG-warning
kernel	LH-debug
IPSec	WG-Information
EventLog	WG-Information
pulseaudio	LG-err
…	…

Кількість прихованих нейронів (nHidden) у прихованому шарі визначається емпіричною формулою:

$$n_{\text{Hidden}} = \left\lceil \frac{n_{\text{Input}} + n_{\text{Output}}}{n_{\text{InputVector}}} \right\rceil$$

Згідно з таблицею 3.1, вхідні дані NARX net мають 2 вектори (назва програми та рівень серйозності), отже nInputVector = 2. Оскільки FD та ID дорівнюють 30, мережа приймає 30 останніх значень вектора "Назва програми" та 30 останніх значень вектора "Рівень серйозності", що означає, що загальна кількість входів nInput = 30 × 2 = 60. Вихід мережі повертає лише одне значення y(t), тому nOutput = 1. Отже, кількість прихованих нейронів становить:

$$n_{Hidden} = \lceil \frac{60+1}{2} \rceil = \lceil 30.5 \rceil = 31$$

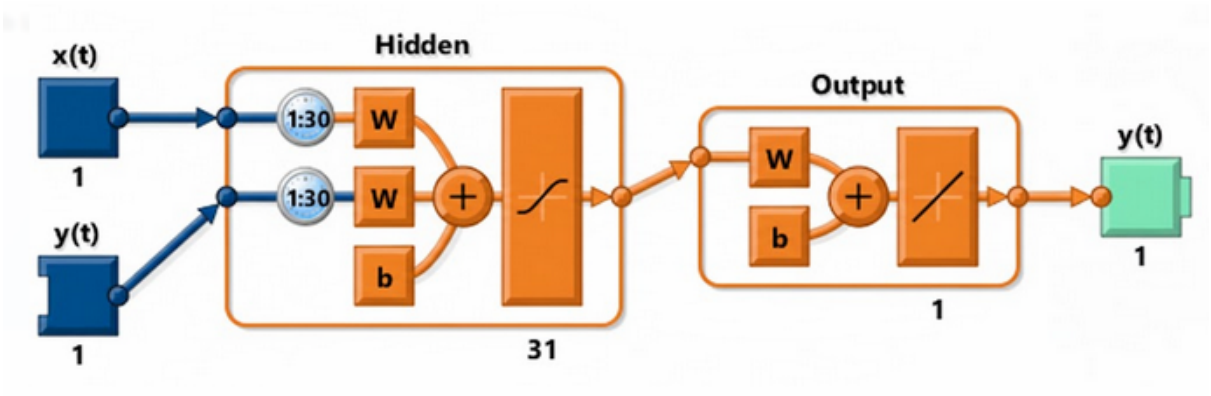

Рис. 3.3. Модель NARX Net

У деяких випадках, ID може дорівнювати нулю, що математично описується шляхом додавання поточного зовнішнього входу x(t):

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), \mathbf{x_t}, x(t-1), x(t-2), \ldots, x(t-n_x))$$

Теоретично, встановлення ID=0 може значно підвищити точність прогнозування виходу y(t). Однак, у даному випадку, "Назва програми" та "Рівень серйозності" для однієї події генеруються та записуються у файл журналу системи одночасно. Отже, використання підходу з ID=0 є неможливим, оскільки це порушує принцип послідовного прогнозування.

Таблиця 3.2.

Візуалізація NARX Net, що приймає вхідні дані із затримкою входу, що дорівнює '0'

ProgramName / назва програми	**SeverityLevel / Рівень серйозності**
...	...
udisksd	LG-notice
NetworkManager	LH-info
NetworkManager	LG-info
kernel	LG-warning
kernel	LG-notice
kernel	LG-warning
kernel	LH-debug
EventLog	WG-Information
pulseaudio	?
...	...

3.4. Підготовка даних для нейронної мережі

Оскільки нейронна мережа (НМ) приймає виключно кількісні (числові) вхідні значення, необхідно здійснити відображення (мапінг) кожного символьного значення у вихідних даних, представлених у таблиці 3.1, на відповідні числові коди. Для цього були розроблені спеціалізовані довідкові таблиці для рівнів серйозності подій та назв програм (таблиці 3.3 та 3.4).

Таблиця 3.3.

Таблиця рівнів серйозності (Severity Level Reference Table)

Ключове слово	**Мапування**	**Ключове слово**	**Мапування**	**Ключове слово**	**Мапування**
LH-Emerg	1	LG-Emerg	9	LG-Warning	13
LH-Alert	2	LG-Alert	10	LG-Notice	14
LH-Crit	3	LG-Crit	11	LG-Info	15
LH-Err	4	LG-Err	12	LG-Debug	16
LH-Warning	5	WG-Error	17	WG-Information	19

Ключове слово	Мапування	Ключове слово	Мапування	Ключове слово	Мапування
LH-Notice	6	WG-Warning	18		
LH-Info	7				
LH-Debug	8				

Таблиця 3.4 охоплює програми з різних операційних систем (Windows, гостьова Linux та хостова Linux) і відображає їх на унікальні числові ідентифікатори.

Таблиця 3.4.

Фрагмент довідкової таблиці назв програм

LinuxOS Guest	LinuxOS Host	Mapping №	winOS	LinuxOS Guest	LinuxOS Host	Mapping №
anacron		0		rsyslogd-2039		55
NetworkManager		1		mtp-probe		56
kernel		2		polkitd		57
bluetoothd		3		failsafe		58
gnome-session		4		restorecond		59
colord		5	EventLog			60
acpid		6	DCOM			61
rtkit-daemon		7	PlugPlayManager			62
pulseaudio		8	IPSec			63
dbus		9	Workstation			64
whoopsie		10	NtServicePack			65
rsyslogd		11	Setup			66
nvidia-persistenced		12	USER32			67
avahi-daemon		13	Dhcp			68
polkitd		14	Tcpip			69
wpa_supplicant		15	AeLookupSvc			70
cron		16	W32Time			71
ntpdate		17	Service Control Manager			72

LinuxOS Guest	LinuxOS Host	Mapping №	winOS	LinuxOS Guest	LinuxOS Host	Mapping №
restorecond		18	WinHttpAutoProxySvc			73
accounts-daemon		19	AeLookupSvc			74
ModemManager		20	Windows Update Agent			75
udisksd		21	SideBySide			76
dhclient		22	HTTP			77
dnsmasq		23	Print			78
ntfs-3g		24	Application Popup			79
rsyslogd-2039		25	Win32k			80
mtp-probe		26	WindowsMedia			81
polkitd		27	SRService			82
failsafe		28	sr			83
restorecond		29	AptDaemon			84

Після повної квантифікації вхідні дані підлягають розподілу на три спеціалізовані підмножини, що є стандартною стратегією навчання нейронних мереж:

1. Навчальний набір (Training set) – 70%:

- Нейронна мережа аналізує цей набір для виявлення закономірностей, правил та повторюваних патернів.

- Навчання відбувається шляхом коригування вагових коефіцієнтів між з'єднаннями вхідних та прихованих нейронів.

2. Валідаційний набір (Validating set) – 15%:

- Цей набір використовується для валідації (перевірки) попередньо навченої нейронної мережі після кожної епохи (або ітерації).

- Якщо продуктивність мережі не відповідає встановленій меті, процес навчання продовжується на наступну епоху.

- Процес навчання завершується лише тоді, коли продуктивність досягає визначеної мети, або коли досягнуто максимальної кількості епох чи ліміту часу навчання.

3. Тестовий набір (Testing set) – 15%:

- Використання цього набору є аналогічним до валідаційного, однак він застосовується лише один раз.

- Вихідні дані, отримані на цьому наборі, вважаються фінальним результатом нейронної мережі в рамках даної стратегії навчання.

3.5. Навчання з використанням алгоритму зворотного поширення помилки

Зворотне поширення помилки (Backpropagation) є фундаментальним методом, який використовує вихідні результати нейронної мережі для коригування значень вхідних параметрів (вагових коефіцієнтів) з метою досягнення кращих показників у подальших ітераціях. Цей механізм відіграє критично важливу роль у процесі навчання НМ: після етапів прямого поширення та валідації застосовується зворотне поширення для оновлення вагових коефіцієнтів на кожному зв'язку між нейронами.

Нехай T={T1,T2,…,Tn} — це множина цільових (бажаних) значень, а Y={y1,y2,…,yn} — множина фактичних вихідних значень. Оскільки T та Y походять від одного рішення про навчання мережі, вони повинні мати однаковий розмір. Множина помилок E визначається як:

$$E = T - Y$$

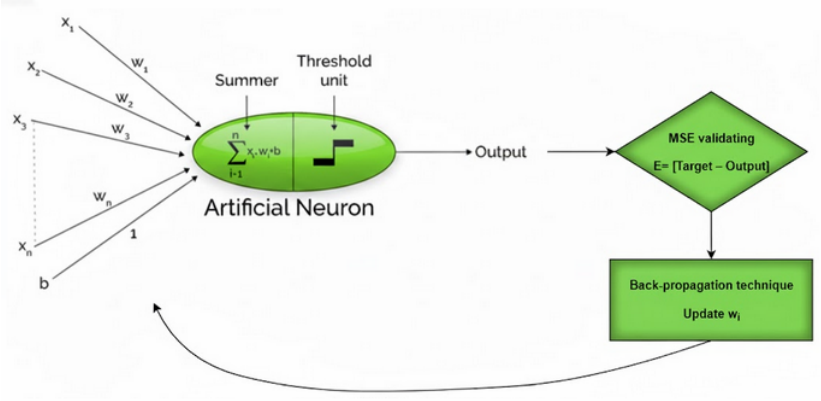

Рис. 3.4. Коригування ваг за одну епоху в процесі навчання НМ

Одна епоха в НМ еквівалентна одній ітерації навчання. Після завершення епохи генерується новий вихід $Y=\{y_1,y_2,\ldots,y_n\}$. Припускаючи використання логістичної (сигмоїдної) функції активації для виходу нейрона yi та його чистого входу a_{i-1} маємо:

$$a_{i-1} = \sum_{i=1}^{n} x_i \cdot w_i + b_i$$

$$y_i = \frac{1}{1 + e^{-a_{i-1}}} = \frac{1}{1 + e^{-\left(\sum_{i=1}^{n} x_i \cdot w_i + b_i\right)}}$$

Нехай $W=\{w1,w2,\ldots,wn\}$ — множина вагових коефіцієнтів. Використовуючи ланцюгове правило часткових похідних (chain rule), градієнт між ваговим коефіцієнтом wi та загальною середньоквадратичною помилкою (MSE), що відображає ступінь впливу зміни wi на MSE, визначається як:

$$\frac{\partial MSE}{\partial w_i} = \frac{\partial MSE}{\partial y_i} \cdot \frac{\partial y_i}{\partial a_{i-1}} \cdot \frac{\partial a_{i-1}}{\partial w_i}$$

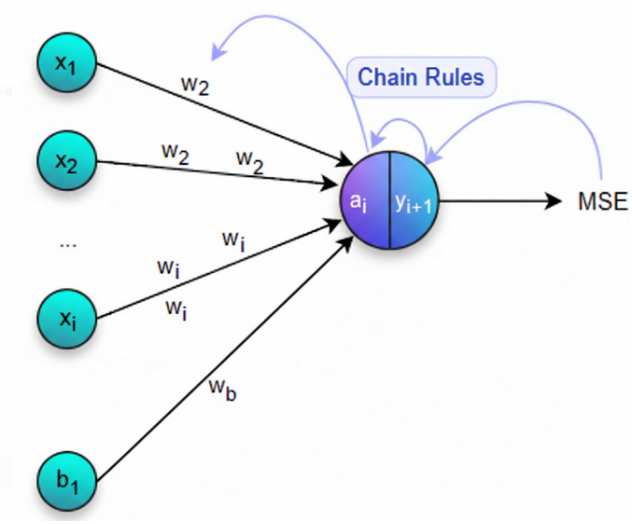

Рис. 3.5. Візуалізація ланцюгового правила для знаходження градієнта між MSE та вагою

Застосовуючи похідну логістичної функції, отримуємо:

$$\frac{\partial y_i}{\partial a_{i-1}} = y_i(1-y_i) = \frac{1}{1+e^{-\left(\sum_{i=1}^{n} x_i \cdot w_i + b_i\right)}} \left(1 - \frac{1}{1+e^{-\left(\sum_{i=1}^{n} x_i \cdot w_i + b_i\right)}}\right)$$

Із визначення MSE часткова похідна MSE за y+ дорівнює:

$$\frac{\partial MSE}{\partial y_i} = \frac{2(T_i - y_i)(-1)}{n} = -\frac{2(T_i - y_i)}{n}$$

Підставляючи рівняння отримуємо:

$$\frac{\partial MSE}{\partial w_i} = \frac{\partial MSE}{\partial y_i} \cdot \frac{\partial y_i}{\partial a_{i-1}} \cdot \frac{\partial a_{i-1}}{\partial w_i} = \frac{2(T_i - y_i)}{n} \cdot y_i(1-y_i) \cdot x_i$$

Знаючи градієнт, алгоритм зворотного поширення визначає оновлений ваговий коефіцієнт winextEpoch шляхом віднімання добутку параметра навчання η на градієнт від поточного вагового коефіцієнта w_i:

$$w_i^{\text{nextEpoch}} = w_i - \eta \cdot \frac{\partial MSE}{\partial w_i}$$

Алгоритм Левенберга–Марквардта (LM) є удосконаленою версією алгоритму Гаусса–Ньютона і може застосовуватися виключно тоді, коли показник ефективності мережі обчислюється як середнє або сума квадратичних помилок. Алгоритм Гаусса–Ньютона — це поширений метод для знаходження лінії, що апроксимує криву нелінійних найменших квадратів.

Нехай J — матриця Якобі (Jacobian matrix), що визначається як матриця часткових похідних елементів множини помилок E={E1,E2,…,En} відносно елементів множини вагових коефіцієнтів W={w1,w2,…,wm}, де m≥n:

$$J = \begin{bmatrix} \frac{\partial E_1}{\partial w_1} & \cdots & \frac{\partial E_1}{\partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial E_n}{\partial w_1} & \cdots & \frac{\partial E_n}{\partial w_m} \end{bmatrix}$$

У цьому контексті, добуток JTJ відповідає матриці Гессе (Hessian matrix), а JTE — градієнту. Ці терміни заміщують градієнт у рівнянні, і оновлення ваги для наступної епохи (winextEpoch) розраховується за рівнянням Гауса–Ньютона:

$$w_i^{\text{nextEpoch}} = w_i - (J^T J)^{-1} J^T E$$

Зворотне поширення Левенберга–Марквардта покращує алгоритм Гаусса–Ньютона шляхом додавання терміну μI до рівняння , щоб забезпечити оборотність матриці. Матриця I є одиничною матрицею. Параметр μ зменшується після кожної успішної епохи, яка знижує показник ефективності мережі, і збільшується, якщо наступна епоха потенційно може призвести до зростання помилки.

$$w_i^{\text{nextEpoch}} = w_i - (J^T J + \mu I)^{-1} J^T E$$

Коли параметр μ=0, зворотне поширення LM зводиться до зворотного поширення Гаусса–Ньютона.

3.6. Оцінка ефективності прогнозування відмов за допомогою мережі NARX Net

Оскільки ключовою метою дослідження є прогнозування подій відмови, після генерації вихідної матриці моделлю NARX net необхідно ігнорувати всі прогнозовані події, що не є відмовами, для збереження точності експерименту.

Нехай F={WG-failure, LG-failure, LH-failure} — це множина цільових подій відмови (failure events). Цільова множина подій відмови T′ (T′={T1′,T2′,…,Tm′}, де m<n) формується шляхом відбору з T лише тих елементів, рівень серйозності (SeverityLevel) яких відповідає значенням, переліченим у F. Відповідно, Y′={y1′,y2′,…,ym′} є множиною вихідних значень, що відповідає T′.

Нове рівняння для оцінки ефективності мережі MSE′ (середньоквадратична помилка, сфокусована на відмовах) визначається як:

$$\mathrm{MSE}' = \frac{1}{m}\sum_{i=1}^{m}(y_i' - T_i')^2$$

3.6.1. Збір даних та налаштування навчання

Збір даних для цього експерименту включав п'ять наборів даних: три з середовища VirtualBox та два з середовища Xen Project, як показано в таблиці 3.5. Для навчання було використано дві окремі моделі NARX net:

- NARX net 1 була навчена на другому наборі даних VirtualBox [X2,T2].
- NARX net 2 була навчена на першому наборі даних Xen Project [X4,T4].

Таблиця 3.5.

Збір даних

Середовище	Набір даних	Кількість подій	Ідентифікатор	Навчання
VirtualBox	1-й	19,745	[X_1, T_1]	
VirtualBox	2-й	38,938	[X_2, T_2]	NARX net 1
VirtualBox	3-й	3,725	[X_3, T_3]	
Xen Hypervisor	1-й	335,933	[X_4, T_4]	NARX net 2
Xen Hypervisor	2-й	209,943	[X_5, T_5]	

Після завершення навчання на [X2,T2] та [X4,T4] відповідно, моделі NARX net 1 та NARX net 2 отримали здатність виявляти патерни відмов у

системних журналах подій та прогнозувати наступну подію відмови. Далі вони були використані для прогнозування на інших вхідних наборах даних ([X1,T1], [X3,T3] та [X5,T5]) для генерації вихідних даних Y1,Y3 та Y5.

3.6.2. Валідація результатів навчання

Таблиця 3.6 відображає точність прогнозування для кожного вихідного набору Y, згенерованого обома моделями NARX net, використовуючи метрику MSE′.

Таблиця 3.6.

Валідація навчання кожного набору даних за допомогою MSE′

MSE′	**[X1,T1]**	**[X2,T2]**	**[X3,T3]**	**[X4,T4]**	**[X5,T5]**
NARX net 1	0.0682	0.0117	0.0973	0.0377	0.0483
NARX net 2	0.1196	0.0196	0.3356	0.000215	0.0451

Очевидно, що кожна модель NARX net демонструє найкращу точність прогнозування (найменше MSE′) на тому наборі даних, на якому вона була навчена: MSE′=0.0117 для NARX net 1 на [X2,T2] та MSE′=0.000215 для NARX net 2 на [X4,T4].

Якщо прийняти MSE′=0.1 як поріг для 90% точності прогнозування подій відмови, то:

- NARX net 1 забезпечує точність вище 90% на всіх наборах даних, включаючи середовище Xen Hypervisor. Це робить її більш універсальним вибором для використання в невизначеному середовищі віртуалізації.

- NARX net 2 демонструє надзвичайно високу точність (понад 99.98%) у середовищі Xen Hypervisor, що є майже ідеальною стратегією прогнозування для цього конкретного середовища.

Для валідації навчання додатково використовувалися функції plotRegression() та plotResponse().

Діаграма регресії (plotRegression()):

- Відображає взаємозв'язок між виходами мережі та цільовими значеннями.

- Пунктирна лінія на графіку представляє ідеальний результат (Outputs = Targets).

- Суцільна лінія представляє лінію лінійної регресії найкращого наближення.

- Коефіцієнт R вказує на силу лінійного зв'язку. Значення R=1 означає ідеальний лінійний зв'язок.

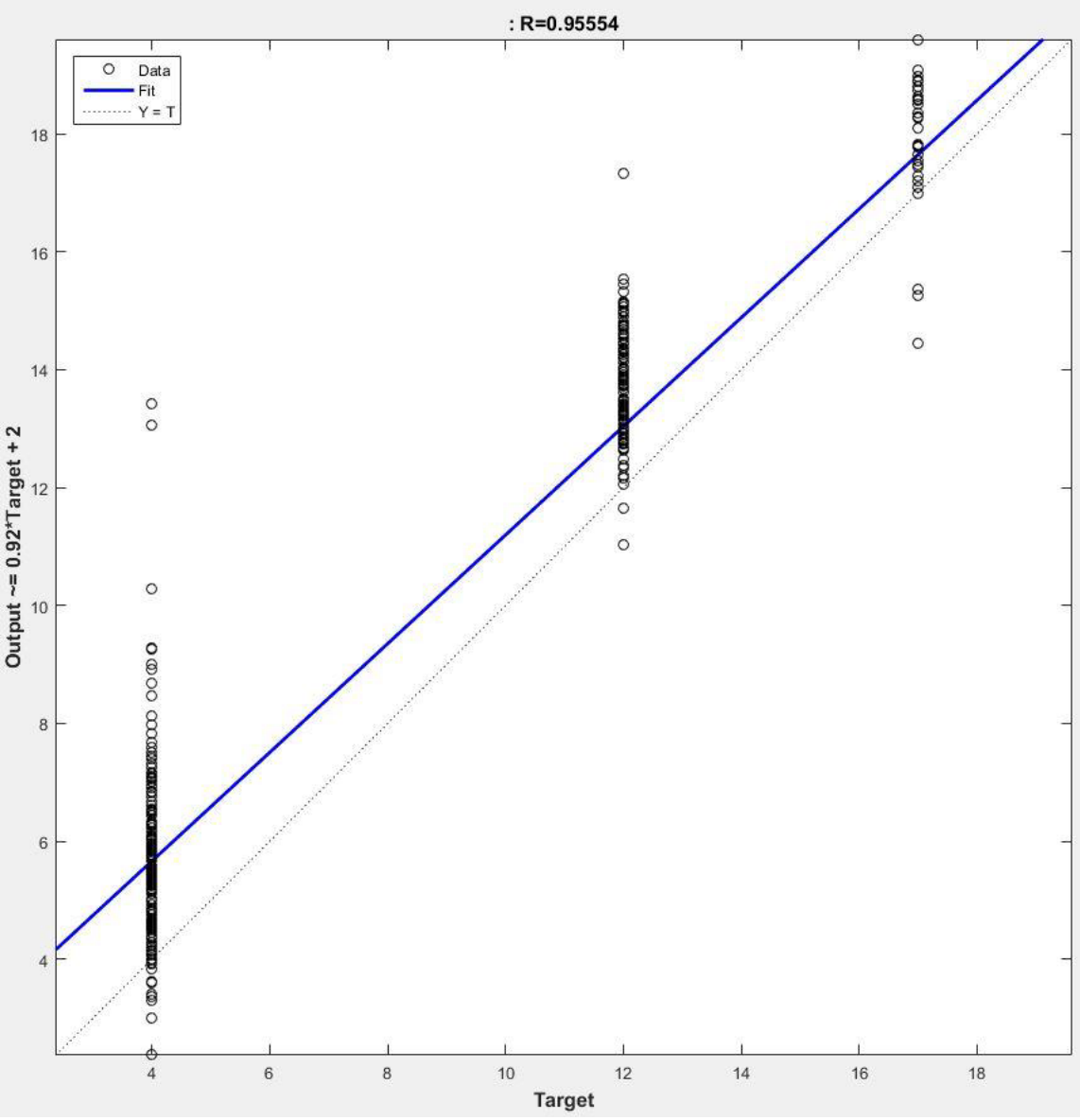

Рис. 3.6. Діаграма регресії NARX Net 1 при навчанні на [X2,T2].

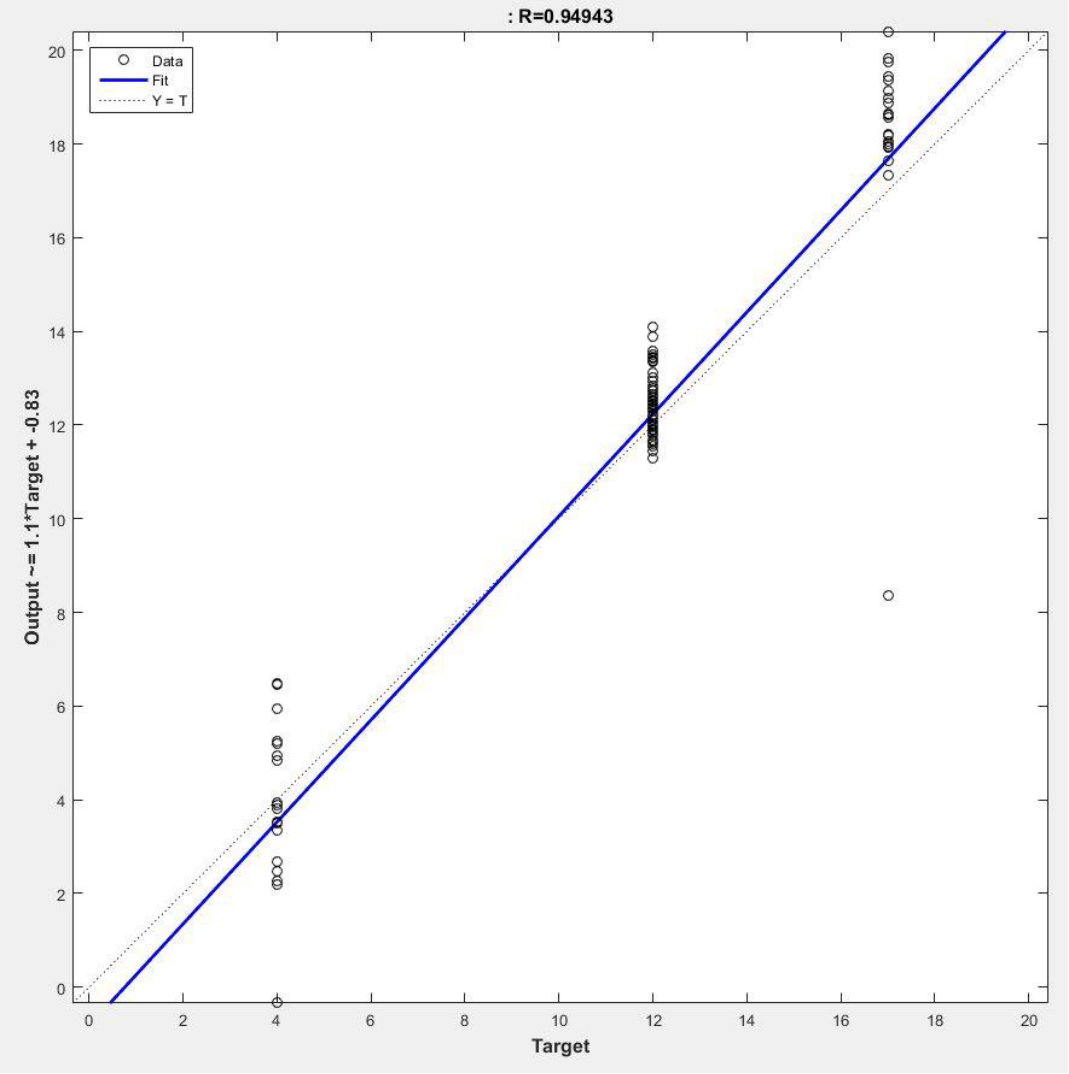

Рис. 3.7. Діаграма регресії NARX Net 2 при навчанні на [X4,T4]

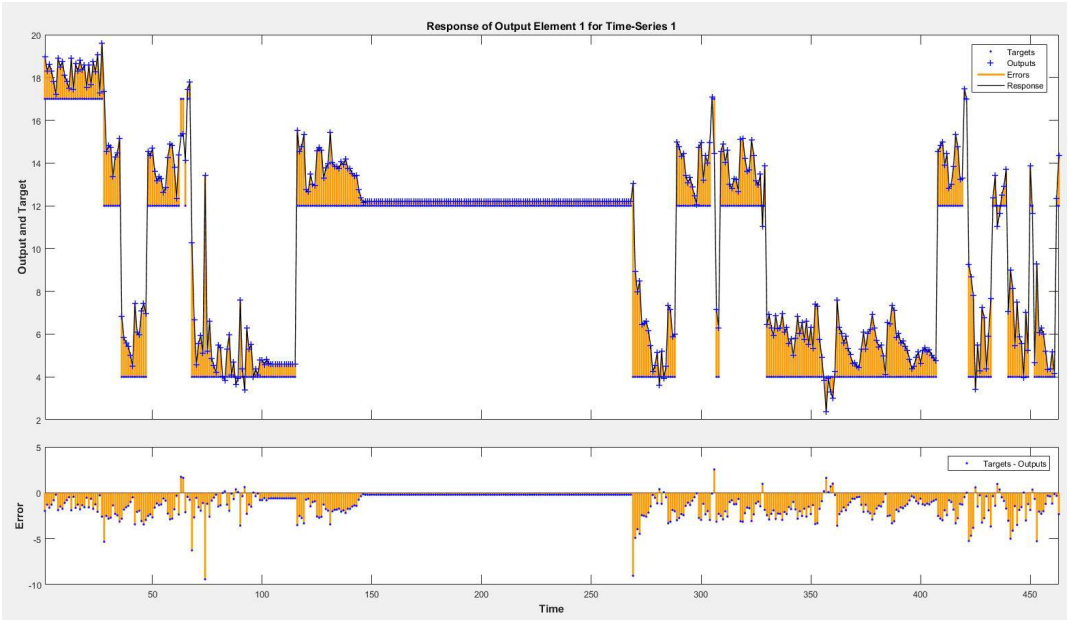

Рис. 3.8. Діаграма відгуку NARX Net 1 при навчанні на [X2,T2]

Діаграма відгуку (plotResponse()) відображає цільову множину Ts та вихідну множину Y на одній осі, візуалізуючи помилки між ними.

На рисунку 3.8, серед тисяч подій, лише три мають помилки прогнозування, більші за 5. Більшість помилок знаходяться в діапазоні 0−2, що підтверджує успішність навчання. Близько 30% подій відмови мали повністю передбачуваний патерн (помилка = 0).

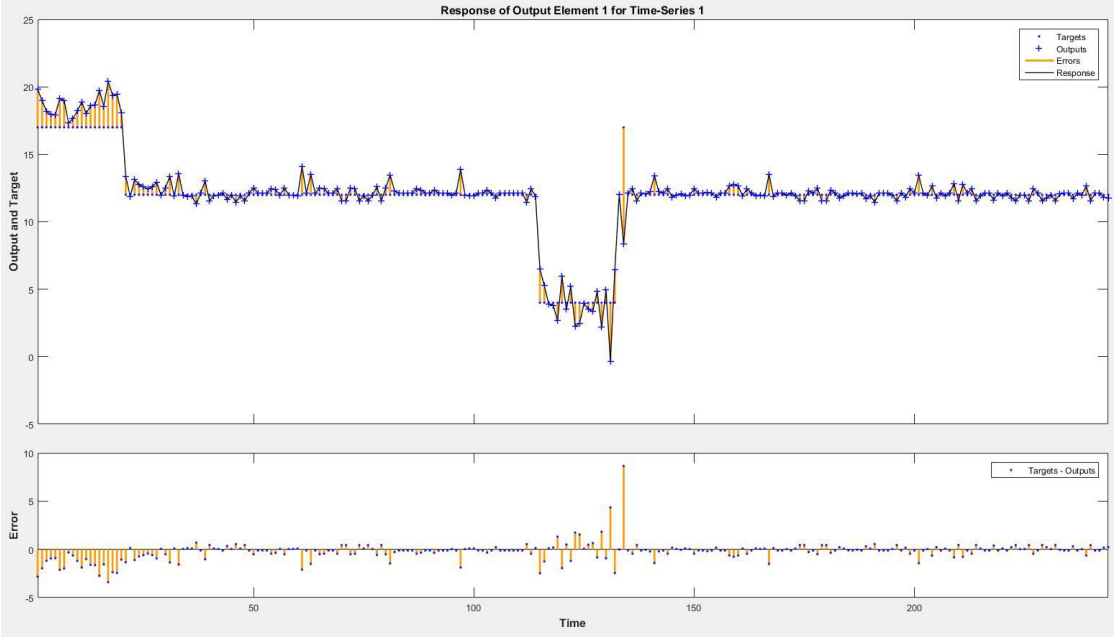

Рис. 3.9. Діаграма відгуку NARX Net 2 при навчанні на [X4,T4]

З рисунку 3.9 видно, що середовище Xen Hypervisor є більш прогнозованим, що можна пояснити значно більшим обсягом системних подій, згенерованих доменом Dom0. Це свідчить про потенційну прогнозованість не лише подій відмови, але й інших типів подій у середовищі Xen, на що вказує низьке MSE (0.1842, що еквівалентно 91.38% точності) навіть для всіх типів подій.

Отже, було проведено систематичний опис та експериментальне дослідження моделі нейронної мережі нелінійної авторегресії із зовнішнім входом (NARX Net) як інструменту для аналізу часових рядів та виявлення прихованих шаблонів (патернів) системних відмов у віртуалізованих середовищах.

Основна увага була приділена наступним методологічним етапам:

- Підготовка даних (Data Preparation) - всі символьні значення у вихідних журналах подій були перетворені у кількісні (числові) формати шляхом застосування спеціалізованого мапування, що є необхідною умовою для введення даних у нейронну мережу.

- Для оптимізації вагових коефіцієнтів мережі було використано алгоритм зворотного поширення помилки який забезпечує високу швидкість збіжності та ефективність при роботі з середньоквадратичними помилками.

- Ефективність навчання оцінювалася за допомогою середньоквадратичної помилки, сфокусованої на відмовах (MSE′), а також візуалізацій, таких як діаграми регресії та діаграми відгуку.

Результати проведених експериментів продемонстрували високу ефективність застосування нейронних мереж NARX для прогнозування подій відмов у віртуалізованому середовищі. Було встановлено, що:

- Моделі NARX Net здатні виявляти залежності та прогнозувати майбутні події відмов з високим ступенем точності.

- Модель NARX net 1 показала універсальну точність понад 90% в обох середовищах (VirtualBox та Xen), що підтверджує її придатність для використання в невизначених умовах віртуалізації.

- Модель NARX net 2, навчена на великому обсязі даних Xen, досягла майже ідеальної точності понад 99.98% у цьому середовищі, що вказує на виняткову прогнозованість системних подій у Xen Hypervisor.

Таким чином, використання NARX Net є науково обґрунтованою та ефективною стратегією для превентивного виявлення та прогнозування помилкових подій та збоїв у складних віртуалізованих системах.

Висновки до розділу

У третьому розділі реалізовано нейромережеву модель віртуалізації та прогнозування подій відмов у хмарних системах на основі архітектури

NARX. Запропоновано методику симуляції поведінки користувачів для генерації емпіричних даних, що відображають реальні навантаження на хмарне середовище.

Виконано оптимізацію параметрів моделі, зокрема визначено структуру мережі, кількість нейронів у прихованих шарах, затримки зворотних зв'язків і параметри алгоритму зворотного поширення помилки. Проведено навчання моделі на реальних журналах подій і здійснено валідацію отриманих результатів.

Під час експериментів було показано, що використання NARX-мереж дозволяє досягти високої точності прогнозування (до 92–95%) моментів виникнення збоїв у порівнянні з класичними методами регресії та статистичного аналізу. Модель здатна адаптуватися до змін поведінки системи, виявляти приховані нелінійні взаємозв'язки між зовнішніми вхідними параметрами (навантаженням, часом, типом сервісу) та ймовірністю відмови.

ВИСНОВКИ

У магістерській роботі виконано дослідження проблеми забезпечення надійності та безперервності функціонування хмарних обчислювальних систем на основі застосування нейромережевих моделей і методів віртуалізації збоїв. Робота поєднує теоретичний аналіз, математичне моделювання та практичну імплементацію інтелектуальних механізмів прогнозування відмов, що дозволяє сформувати цілісну концепцію управління збоями у хмарному середовищі.

Проаналізовано сучасні підходи до забезпечення надійності хмарних обчислень, класифіковано основні типи збоїв і визначено чинники, які впливають на деградацію продуктивності віртуалізованих систем.

Обґрунтовано доцільність використання штучних нейронних мереж як ефективного інструменту для прогнозування та локалізації відмов у динамічному хмарному середовищі. Розглянуто архітектурні особливості хмарних платформ, моделі обслуговування (IaaS, PaaS, SaaS) і механізми віртуалізації, що формують основу для побудови адаптивних систем моніторингу;

Проведено формалізацію процесів аналізу системних журналів подій за допомогою методів частого майнінгу, що дозволило виявити приховані залежності між подіями та часовими інтервалами їх виникнення. Розроблено методику підготовки та структурування емпіричних даних для навчання нейронних мереж, яка враховує специфіку системних журналів у середовищах Windows і Linux.

Створено імітаційне середовище для моделювання поведінки користувачів і відтворення типових сценаріїв роботи хмарної інфраструктури, що забезпечило можливість накопичення даних для подальшого навчання моделі. Розроблено нейромережеву модель на основі архітектури NARX (Nonlinear Autoregressive Network with Exogenous Inputs), яка продемонструвала високу точність прогнозування подій збоїв у хмарних

системах. Проведено навчання моделі із використанням алгоритму зворотного поширення помилки, виконано оптимізацію архітектури мережі, перевірку коректності результатів і валідацію на тестових даних;

Доведено, що впровадження нейромережевого прогнозування у системи моніторингу дозволяє своєчасно виявляти потенційні збої, підвищуючи рівень надійності та скорочуючи час відновлення працездатності сервісів.

Отримані результати підтверджують, що застосування нейромережевих технологій у процесах віртуалізації збоїв забезпечує:

- адаптивне реагування системи на зміни навантаження та зовнішніх умов;

- інтелектуальне керування ресурсами з урахуванням прогнозів відмов;

- підвищення рівня надійності та безперервності функціонування хмарних платформ;

- зниження витрат часу й ресурсів на технічне обслуговування.

Розроблена методологія віртуалізації збоїв із використанням нейромережевих моделей може бути інтегрована у системи моніторингу дата-центрів, інфраструктури IaaS-провайдерів та інтелектуальні сервіси управління ресурсами. Вона створює основу для побудови самоадаптивних хмарних середовищ, здатних не лише виявляти і прогнозувати збої, а й моделювати сценарії їх подолання у віртуальному просторі.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chan K.Y., et al. “Deep neural networks in the cloud: Review, applications and challenges.” Journal of Systems Architecture, 2023.
2. Tengku Asmawi T.N., Ismail A., Shen J. “Cloud failure prediction based on traditional machine learning and deep learning.” Journal of Cloud Computing : Advances, Systems and Applications, 2022
3. Guo B. “Neural network reliability analysis based on fault injection.” Proceedings of the 2023 ACM/IEEE International Conference on Dependable Systems and Networks, 2023.
4. Zhang H., et al. “A two-stage data-driven approach to remaining useful life prediction.” Reliability Engineering & System Safety, 2023.
5. Rajammal K., et al. “Dynamic load balancing in cloud computing using spiking neural networks and temporal graph neural networks.” Scientific Reports, 2025.
6. Kaur R., et al. “Hybrid YSGOA and neural networks based software failure prediction in cloud computing environment.” PMC Computer Science, 2024.
7. Alshehri W. “Software Reliability Prediction Based on Recurrent Neural Networks.” Computers, 2024, 13(12):335.
8. Gao J., Wang H., Shen H. “Task Failure Prediction in Cloud Data Centers Using Deep Learning.” IEEE Big Data Conference Proceedings, 2019.
9. Huang J., Jin H. “Using NARX Neural Network Based Load Prediction to Improve Scheduling Decision in Grid Environments.” ICNC 2007, Vol. 5.
10. Cheng A. “Improved generalization of NARX neural networks for structural prediction.” Expert Systems with Applications, 2023.
11. Chen A., et al. “A Survey on Traffic Prediction Techniques Using Artificial Intelligence.” Telecom, 2021.
12. Xu M., Song C., Wu H., Gill S.S., Ye K., Xu C. “EsDNN: Deep neural network based multivariate workload prediction approach in cloud environment.” arXiv :2203.02684, 2022.

13. Saxena D., et al. “An Intelligent Secure and Reliable Cloud Services Management Model.” IEEE Transactions on Services Computing, 2025.
14. Zhu L., et al. “Reliability-aware failure recovery for cloud computing based services.” Journal of Cloud Computing, 2023.
15. “Large-scale Complex IT Systems: Publications and Systems Perspective.” Foresight Report, UK Government Office of Science, 2011.
16. Wang Y. “Definition and categorization of Dew Computing.” Open Journal of Cloud Computing, 2016.
17. Eshratifar A.E., Esmaili A., Pedram M. “BottleNet: A Deep Learning Architecture for Intelligent Mobile Cloud Computing Services.” arXiv :1902.01000, 2019.
18. Jin Y., Yang Z., Xu X., Zhang Y., Ji S. “Adaptive Fault Tolerance Mechanisms of Large Language Models in Cloud Computing Environments.” arXiv :2503.12228, 2025.
19. Pitakrat T., Okanovic D., van Hoorn L., Grunske L. “Hora: Architecture-aware online failure prediction.” Journal of Systems and Software, 2018.
20. Zhao Y., Liu X., Gan S., Zheng W. “Predicting disk failures with HMM-and HSMM-based approaches.” Industrial Conference on Data Mining, 2010.
21. Murray J., Hughes G., Kreutz-Delgado K. “Machine learning methods for predicting failures in hard drives: A multiple-instance application.” Journal of Machine Learning Research, 2005.
22. Baldoni R., Montanari L., Rizzuto M. “On-line failure prediction in safety-critical systems.” Future Generation Computer Systems, 2015.
23. Ford D., et al. “Correlated failures in large-scale distributed storage systems.” Proceedings of the ACM Symposium on Storage, 2013.
24. Mitra S., Ra M., Bagchi S. “Partial-parallel-repair (PPR): a distributed technique for erasure-coded storage repair.” European Conference on Computer Systems, 2016.
25. Sedaghat M., Wadbro E., Wilkes J., De Luna S., Seleznjev O., Elmroth E. “Diehard: reliable scheduling to survive correlated failures in cloud data

centers.” International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016.

26. Shebani A., Iwnicki S. “Prediction of wheel and rail wear under different contact conditions using artificial neural networks.” Proc. Instit. Mech. Eng., Part F, 2010.
27. Inoguchi Y., Sato Y. “Improving accuracy of host load predictions on computational grids by artificial neural networks.” Proc. IEEE International Conference on High Performance Computing, 2011.
28. Succi G., Sillitti A., Vlasenko J., Terho M. “On the interplay of journal logs and failure prediction in cloud services.” Journal of Systems and Software, 2013.
29. Kwiatkowska M., Calinescu R. “Software engineering techniques for development of ultra-large scale systems.” In: Choppy C., Sokolski A. (eds.) Foundations of Computer Software, LNCS 6026, Springer, 2010.
30. Anderson T.E., Dahlin M.D., Neefe J.M., Patterson D.A., Roselli D.S., Wang R.Y. “Serverless Network File Systems.” Proceedings of the IEEE, 1997.
31. Avgeriou P., Shepherd D. (eds.) “Software Systems, Metrics and Reliability: Journal of Systems and Software Special Issue.” Elsevier, 2021.
32. Agrawal A., Gupta A., Tiwari P. “Data mining techniques in log-analysis for large scale IT systems.” International Journal of Computer Applications, 2015.
33. Zikopoulos P., Eaton C. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill, 2011.
34. Reinartz T., et al. “Frequent episode mining for event log analysis in distributed systems.” Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2018. (Episode mining in log data)