Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ткаченко Д.М.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ткаченку Давиду Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Маркетингове та економічне обґрунтування IT-проєкту.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Ткаченко Д.М. Дослідження та програмна реалізація системи гіперконвергенції ІТ-інфраструктури. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи гіперконвергенції ІТ-інфраструктури.

Метою розробки є дослідження та програмна реалізація системи гіперконвергенції ІТ-інфраструктури.

Об'єктом дослідження є процес гіперконвергенції ІТ-інфраструктури.

Предметом дослідження є методи гіперконвергенції ІТ-інфраструктури.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи гіперконвергенції ІТ-інфраструктури.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, гіперконвергенція, ІТ-інфраструктура

ABSTRACT

Tkachenko D.M. Research and software implementation of IT infrastructure hyperconvergence system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the IT infrastructure hyperconvergence system.

The purpose of the development is the research and software implementation of the IT infrastructure hyperconvergence system.

The object of research is the process of IT infrastructure hyperconvergence.

The subject of research is methods of IT infrastructure hyperconvergence.

The research methods are based on the methods of the theory of building computer networks, the methods of mathematical statistics, and the methods of software development.

The result of the work is the software implementation of the IT infrastructure hyperconvergence system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, hyperconvergence, IT infrastructure

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
ПС	–	програмна середа
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних

ВСТУП

Актуальність теми. За останній рік гіперконвергенція чи стала не самим модним трендом в IT, хоча ще два-три роки тому про неї мало хто чув. Аналітики пророкують цьому сегменту інтегрованих рішень запаморочливі темпи росту – 50-70% щорічно. Відповідні пристрої вже почали пропонувати HPE, EMC і Cisco – найбільші гравці ринку серверів, СЗД і мережного встаткування. Із чим зв'язана така популярність гіперконвергентних систем і наскільки обґрунтовані надії на успішне рішення з їхньою допомогою завдань, з якими зіштовхуються корпоративні IT-відділи?

Гіперконвергенція реалізує програмно обумовлений підхід до керування зберіганням інформації у системах зберігання даних, поєднуючи засоби зберігання, обчислювальні потужності, мережні ресурси й технології віртуалізації в одному фізичному пристрої, керування яким здійснюється як єдиним цілим**.**

Історію гіперконвергентних рішень у їхньому поточному втіленні можна простежити до 2009 року, коли були утворені два стартапа, з якими вони асоціюються в першу чергу, – Nutanix і SimpliVity. Компанія Nutanix вийшла на ринок улітку 2011 року зі своїм програмно-апаратним рішенням Complete Cluster, а через півтора року свій гіперконвергентний пристрій OmniCube представила SimpliVity. OmniCube компанії SimpliVity – один з перших гіперконвергентних пристроїв на ринку. OmniCube у форм-факторі 2U поєднує обчислювальні ресурси, гіпервізор, сервіси зберігання на базі серверної платформи x86. Крім єдиного централізованого керування, підтримуються такі функції, як убудоване резервне копіювання ВМ, дедуплікація даних не лету, стиск і оптимізація на джерелі й ін.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем гіперконвергенції IT-інфраструктури.

– Дослідження системи гіперконвергенції IT-інфраструктури.

– Програмна реалізація системи гіперконвергенції IT-інфраструктури.

Об'єктом дослідження є процес гіперконвергенції IT-інфраструктури.

Предметом дослідження є методи гіперконвергенції IT-інфраструктури.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод гіперконвергенції IT-інфраструктури.

– Розроблено вітчизняний продукт гіперконвергенції IT-інфраструктури, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі гіперконвергенції IT-інфраструктури.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Nutanix запропонувала три ключові на той момент інновації: обчислювальні потужності й ресурси зберігання були об'єднані в єдине ціле (один рівень) за допомогою розподіленої файлової системи, апаратне забезпечення спеціально адаптоване для підтримки віртуалізації з єдиним керуванням всіма ресурсами, а вся система оптимізована для використання SSD (у той час флеш-накопичувачі не одержали ще настільки широкого поширення, як сьогодні). Такий підхід дозволяв відмовитися від окремої мережі зберігання, забезпечував швидке розгортання й нарощування високопродуктивної віртуалізированной інфраструктури й сервісів на її основі (за прикладом Google і Facebook).

Ранній вихід на ринок дозволив Nutanix зайняти лідируюче положення. Так, по даним IDC, у першій половині 2022 року на її частку доводилося більше половини всіх продажів гіперконвергентних рішень (52%). Однак уже тоді було ясно, що компанії навряд чи вдасться надовго зберегти настільки високі показники. Хоча первісне оголошення Nutanix про випуск нового продукту залишилося по більшій частині непоміченим, потенційний попит на прості в керуванні, розгортаємі швидко й відносно недорогі інтегровані рішення залучив безліч нових гравців – тільки за останні кілька місяців таких виробників з'явилося більше десятка.

Потенційний попит, втім, швидко перетворився в реальний. За даними дослідження Actual Tech Media «Стан ринку гіперконвергентної інфраструктури», в 2023 році вже майже чверть із більше 500 опитаних компаній (24%) використовували відповідні рішення. Навіть із обліком того, що опитування проводилося на тематичному сайті й, отже, у ньому брали участь ті, хто зацікавлений у даному рішенні, цифра вражає. По даним IDC, на

гіперконвергентні системи доводилося 10,9% ринку конвергентних систем, а Technology Business Research прогнозує, що до 2027 року їхня частка зросте до 32%. Відповідно до оцінок IDC, уже цього року обсяг продажів гіперконвергентних систем складе 2 млрд доларів і до 2025-му досягне 4 млрд доларів, а по розрахунках Gartner, ринок гіперконвергентних інтегрованих систем буде щорічно збільшуватися на 68% – з 371,5 млн доларів в 2022 році до 5 млрд в 2025-м.

1.2 Область застосування

На хвилі зростаючого інтересу перспективний сегмент не міг не зацікавити й багаторічних лідерів IT-ринку. Так, у лютому цього року EMC представила інтегрований пристрій VxRail, а в березні свою систему HyperFlex анонсувала Cisco. Такі рішення, випускаються найчастіше в партнерстві з ким-небудь зі стартапів, є й в інших традиційних гравців, зокрема в HPE, Dell, Hitachi, Fujitsu і Lenovo. А завдяки функціональності Storage Spaces Direct убудована підтримка гіперконвергентного кластера з'явиться й в Microsoft Windows Server 2023.

Як свідчать останні анонси нових продуктів, інтерес до цього ринку величезний, але чи заслуговує він того? Буквально за пару років його засновники, Nutanix і SimpliVity, увійшли в коло «єдинорогів» – стартапів, чия ринкова вартість перевищує мільярд доларів. Так, на кінець 2022 року Nutanix оцінювалася в 2,2 млрд доларів, а SimpliVity – в 1 млрд доларів на березень 2023-го (на підставі притягнутого венчурного фінансування). Тим часом, навіть при реалізації оптимістичних прогнозів щодо розміру ринку, виникають сумніву в тому, що інвесторам удасться швидко повернути свої вкладення – занадто багато претендентів на відносно невеликий по розмірах пірог (для порівняння, щорічний оберт таких великих гравців ринку СЗД, як EMC і NetApp, становить 56 і 15 млрд доларів відповідно).

Насправді потенційний приз значно більше, ніж щорічний оберт в 4-5 млрд, про яке говорять IDC і Gartner (а є й менш оптимістичні оцінки). Найчастіше гіперконвергентні пристрої розглядаються як рішення для малих і середніх підприємств, і багато які стартапи їх так і позиціонують. Дійсно, поки основними споживачами є підприємства SME і філії великих компаній. Однак і Nutanix, і SimpliVity із самого початку націлювалися на інший ринок, уважаючи пріоритетним для себе напрямком центри обробки даних, де відбувається перехід до програмно обумовлених ЦОДам. А це вже зовсім інші обсяги. У своєму дослідженні «Ринок програмно обумовлених ЦОДів» аналітичне агентство Markets&Markets очікує, що відповідний ринок виросте з 22 млрд доларів в 2023 році до 77 млрд в 2027-м. Але наскільки гіперконвергентні пристрої підходять для цього завдання? Перш ніж намагатися відповісти на це питання, необхідно спочатку зрозуміти, а що ж таке гіперконвергенція.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи гіперконвергенції ІТ-інфраструктури, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Гіперконвергентний пристрій VxRail з єдиним керуванням

Після викупу в Cisco більшої частини її частки в спільному підприємстві VCE компанія EMC представила перший новий продукт, розроблений у партнерстві з VMware, – гіперконвергентне рішення VxRail.

VCE, спільне підприємство EMC, VMware і Cisco, торік перейшло під контроль EMC. В 2023 році EMC продала рекордну кількість конвергентних систем зберігання – у цілому на 3 млрд доларів; при цьому сума більшості угод становила від 10 до 70 млн доларів. По оцінках аналітиків, цього року на конвергентні системи зберігання буде доводитися 20% всіх продажів СЗД. Так що бажання EMC одержати повний контроль над перспективною продуктовою лінійкою цілком зрозуміло.

Про причини виходу Cisco з альянсу можна тільки догадуватися: компанія не цілком задоволена політикою VMware як потенційного конкурента на традиційному для неї ринку мережних рішень. Втім, можливо, причина криється в різному стратегічному баченні перспектив розвитку: на недавньому заході Cisco Live у Берліні компанія фактично заявила про перевагу контейнерної віртуалізації.

Конвергентні рішення з'явилися у відповідь на потребу ринку у швидкому розгортанні обчислювальної інфраструктури при всі складності, що підвищується, інтеграції складових її компонентів (серверів, систем зберігання, мережного встаткування). Однак такі продукти, як VBlock, можуть собі дозволити тільки великі компанії, а оперативне нарощування потужностей

необхідно й середнім по розмірі підприємствам. Відповіддю ринку стала пропозиція компактних конвергентних рішень від таких стартапів, як Nutanix і SimpliVity. Обсяг відповідного сегмента оцінюється в 800 млн доларів.

Своє гіперконвергентне рішення EMC розробила разом з VMware. VxRail являє собою компактний інтегрований пристрій (appliance) висотою 2U с чотирма серверними вузлами (до 112 ядер). Залежно від моделі воно може мати гібридне сховище або містити тільки флеш-накопичувачі (до 76 Тбайт флеш-пам'яті). Один пристрій здатний підтримувати до 200 віртуальних машин. Якщо знадобиться подальше масштабування, до 16 VxRail можуть бути об'єднані в кластер, розрахований на підтримку до 3000 ВМ. Як затверджується, ціна на таке рішення на 25% нижче, ніж у конкурентів (вартість пристрою в мінімальній конфігурації становить 60 тис. доларів). Цільові клієнти – середній бізнес і регіональні офіси великих підприємств.

Якщо конвергентні рішення мають традиційну архітектуру з виділеними функціональними блоками, то гіперконвергентні являють собою окремий пристрій. Крім того, VBlock і аналогічні продукти призначені для виконання традиційних робочих навантажень, таких як SAP ERP, бази даних Oracle і т.п., а VxRail – для підтримки віртуальних машин і сервісів роботи з даними: тиражування, резервного копіювання й вивантаження даних у хмару. У компанії характеризують VxRail як «програмно обумовлене сховище із глибокою інтеграцією з vSphere».

Однак головна відмінність складається в єдиному керуванні всією системою. VxRail містить комплект гіперконвергентного ПЗ VMware, до складу якого входять vCenter для керування віртуалізированою інфраструктурою, гіпервізор vSphere і VirtualSAN для віртуалізації СЗД. Крім досягнення максимальної сумісності з використовуваним устаткуванням, VxRail містить керуюче ПЗ VxRail Manager, що здійснює автоматичне настроювання всіх програмних компонентів – усього близько 200 різних операцій. Завдяки цим можливостям скорочується час початкового розгортання (15 хв із моменту

включення до запуску першої ВМ) і пред'являються менш тверді вимоги до кваліфікації обслуговуючого персоналу.

EMC планує випустити велика кількість різних конфігурацій VxRail (різноманітні процесори, оперативна пам'ять, флеш-накопичувачі, жорсткі диски, мережні з'єднання), щоб замовник міг вибрати необхідну. Зараз доступні VxRail із чотирма вузлами (серверами), незабаром – із трьома (додаткові екземпляри пристрою можна здобувати з одним вузлом). Потрібні додатки можуть бути автоматично завантажені й установлені з магазина додатків.

Cisco HyperFlex

Головним драйвером появи гіперконвергентних рішень стало прагнення замовників заощадити, і в першу чергу за рахунок відмови від дорогих традиційних (виділених) систем зберігання даних і мереж SAN. Такі рішення являють собою єдину систему, що складається з універсальних вузлів, що реалізують як обчислювальні функції, так і функції зберігання даних, а крім того, що забезпечують мережну взаємодію. Рішення Cisco HyperFlex побудоване на основі серверів Cisco UCS, до яких додана платформа HX Data. Остання поєднує твердотільні й дискові накопичувачі в єдине розподілене багаторівневе об'єктне сховище даних.

На даний момент Cisco пропонує три можливі конфігурації гіперконвергентної системи. Перший варіант кластера складається з вузлів висотою 1U (HX220c), другий – з вузлів 2U (HX240c), кожний з яких побудований на базі сервера UCS і містить твердотільні й дискові накопичувачі; в одному кластері може бути до восьми вузлів, хоча незабаром ця характеристика значно збільшиться: компанія вже тестує гіперконвергентні кластери з 64 вузлів. Третій варіант – гібридний, коли в кластері присутні як гіперконвергентні вузли (HX240c), так і звичайні обчислювальні сервери. Використання цього підходу сприяє підвищенню гнучкості і є одним з відмінностей HyperFlex від усіх раніше представлених іншими компаніями рішень, які фахівці Cisco відносять до першого покоління гіперконвергентних систем.

Cisco HX Data Platform реалізує такі стандартні для корпоративних систем функції керування даними корпоративного класу, як моментальні знімки (Snapshots), динамічне виділення ємності (Thin Provisioning) і миттєве клонування (Cloning). Кожний блок даних зберігається у двох або трьох екземплярах на різних вузлах кластера. Якщо з окремим вузлом (сервером) трапляться неполадки, то віртуальна машина автоматично переїде на інший і продовжить користуватися даними з доступних вузлів. При цьому відбувається відновлення необхідного числа резервних копій. Після поновлення роботи сервера його ресурси будуть повернуті в єдиний пул.

Системи HyperFlex поставляються із уже передвстановленим гіпервізором (поки тільки VMware, але підтримка інших гіпервізорів уже запланована) і іншими програмними компонентами. Розгортання віртуальних машин здійснюється за принципом Plug-n-Play і, по даним Cisco, займає не більше 1 ч. Однак звичайно на це йде менш 15 хв, а протягом заявленої години можна спокійно зробити все необхідне, включаючи установку самих пристроїв і їхнє підключення.

Cisco не позиціонує гіперконвергентні рішення як заміна конвергентним, зокрема Vblock і FlexPod. На його думку, ринки конвергентних і гіперконвергентних рішень стануть розвиватися паралельно, при цьому темпи росту нового сегмента будуть вище. Можливо й спільне використання цих систем. У цілому простої розгортання й адміністрування вкупі з убудованою відказостійкістю й централізованим керуванням роблять гіперконвергентні рішення кращим вибором для застосування у великих віддалених офісах і філіях компаній, а також для реалізації серверної віртуалізації, інфраструктури віртуальних робочих столів (VDI), середовищ для тестування й розробок.

Що стосується нового покоління комутаторів Nexus (Nexus 9300EX і Nexus 9200), те вони побудовані на базі спеціалізованих мікросхем Cisco Cloud Scale ASIC, а не комерційних чипів (як комутатори попереднього покоління). Власна елементна база – важлива конкурентна перевага, що забезпечує

комутаторам Cisco більше високу продуктивність (до 30 портів 100G на чип), масштабованість, можливість наскрізного аналізу трафіку на швидкості каналів та ін. Що особливо важливо для замовників, нові високошвидкісні комутатори пропонуються за ціною попередніх систем. Багато в чому це стало можливим завдяки підвищенню швидкості передачі даних по одній лінії – наприклад, десятиканальна реалізація 100Gb (10х10Гбіт/с) поступилася місцем більше економічній чотирьохканальной (4х25 Гбіт/с). Підтримуються порти 1, 10, 25, 40 і 100 Гбіт/с, причому, як запевняють в Cisco, 25-гігабітні порти пропонуються за ціною 10-гігабітних, а 100G – за ціною 40G.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив’язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп’ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи гіперконвергенції IT-інфраструктури.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Постачальники гіперконвергентних систем позиціонують свої продукти як відповідь на самі насущні виклики, з якими зіштовхуються корпоративні IT-відділи: швидке розгортання встаткування для підтримки нових послуг, скорочення капітальних і операційних витрат, недостача кваліфікованих IT-кадрів, спрощення керування IT-інфраструктурою, підвищення захищеності й доступності даних і т.д.

Відповідно до відомої пропорції, 80% усього часу витрачається на підтримку IT-інфраструктури в належному стані й тільки 20% – на її стратегічний розвиток для рішення завдань бізнесу, оскільки виконання поточних операцій поглинає більшу частину робочого часу співробітників IT-відділів. Відповідно до недавнього дослідження IDC, застосування гіперконвергентних систем дозволило корпоративним IT-відділам більше часу приділяти інноваціям і новим проектам – 29% замість колишніх 16%. Крім того, у результаті вдалося збільшити частку IT-бюджету, що направляється на нові проекти, – з 43 до 56%. І це тільки одне з потенційних переваг реалізації гіперконвергентної інфраструктури. Аналітики з 451 Research затверджують, що постачальники HCI найбільше часто виділяють ще чотири.

Скорочення числа керованих систем

Один гіперконвергентний вузол поєднує обчислювальні й СЗД-ресурси, що, відповідно, веде до скорочення числа окремих пристроїв і, як наслідок, зменшенню кількості об'єктів, які треба здобувати, установлювати й обслуговувати. Крім того, більше простому розгортанню й обслуговуванню гіперконвергентних пристроїв сприяє те, що вони базуються на стандартних серверних компонентах (це не вимагає знання пропрієтарного встаткування

постачальників традиційних СЗД). Нарешті, наявність у багатьох рішеннях інтегрованих інструментів керування спрощує завдання адміністрування.

Постачальники HCI навіть заявляють про появу нового класу IT-фахівців – універсалів (generalists). Завдяки автоматизації багатьох функцій керування, їм не доводиться вникати в специфічні деталі архітектури серверів і СЗД, як їх вузькоспеціалізованим побратимам, які в еру до-HCI фокусувалися на обслуговуванні тільки одного виду встаткування. Більше просте керування апаратним забезпеченням особливо привабливо для невеликих компаній і віддалених філій, де немає умов для утримання штату фахівців.

Спрощення масштабування

Як відзначають аналітики 451 Research, масштабування систем зберігання завжди було важкою справою: після заповнення наявних полиць для дисків або здобувалася ще одна система, який доводилося управляти окремо, або здійснювався перехід на більшу систему. На відміну від них, гіперконвергентні системи розраховані не на вертикальне (scale-up), а на горизонтальне (scale-out) масштабування. Коли виникає необхідність у додатковій обчислювальній потужності, IT-відділу досить придбати ще один вузол і додати його до наявного кластера. У порівнянні із традиційними й конвергентними рішеннями типовий квант нарощування значно менше.

Фокус на ВМ

Гіперконвергентні продукти споконвічно створювалися розраховуючи на підтримку віртуальних середовищ. Так, багато постачальників розробляли своє програмне забезпечення з обліком того, що керування зберіганням повинне здійснюватися з погляду віртуальних машин, а не СЗД. Традиційні системи зберігання оптимізовані для керування LUN. Однак, коли на хості виконуються декілька ВМ, всі вони найчастіше зберігаються в одному LUN. Ця особливість утрудняла застосування індивідуальної політики зберігання для ВМ, до того ж воно відбувалося на рівні системи зберігання.

На противагу такому підходу ВМ-центричні гіперконвергентні платформи дозволяють застосовувати політикові до окремих ВМ, призначаючи для кожної свої правила резервного копіювання, тиражування й т.д. В остаточному підсумку для IT-адміністраторів пріоритетом є керування ВМ, а не інфраструктурними складовими. А інструменти керування HCI дозволяють контролювати все гіперконвергентне середовище з однієї адміністративної консолі. Тим самим підвищується можливість маневрування (agility), тому що ВМ можна швидко переміщати між вузлами у відповідь на вимоги, що змінилися.

Підвищення продуктивності

Як і багато новітніх систем зберігання, практично всі гіперконвергентні пристрої містять флеш-накопичувачі. Вони відрізняються від флеш-масивів тим, що перебувають усередині фізичного сервера, забезпечуючи меншу – у порівнянні із зовнішніми системами зберігання – затримку. Крім того, підвищуючи продуктивність, флеш-масиви не вирішують проблеми ізольованого зберігання, не припускають єдиного керування інфраструктурою й не забезпечують плавного масштабування ЦОДа.

Використання локального кеша на базі флеш-накопичувачів підвищує продуктивність при роботі з даними, однак який може бути реальна продуктивність гіперконвергентного кластера при його масштабуванні до декількох десятків вузлів? Якщо віртуальна машина функціонує на конкретному вузлі, то дані розподіляються між вузлами. Нові записи сегментуються, звичайно по числу вузлів. Кожний сегмент передається через комутатор всім вузлам у кластері. І це тільки одна операція. А наскільки більшим виявиться мережний трафік, коли буде потрібно виконати десятки тисяч операцій уведення-виводу в секунду (але ж крім цього відбувається звичайна передача даних між серверами й між серверами й користувачами). Все це порушує питання про якість і продуктивність мережної інфраструктури.

Найчастіше у визначенні гіперконвергентного рішення мережні компоненти навіть не згадуються як його інтегральна частина. І якщо навіть

згадування про неї є присутнім, то в більшості випадків воно носить формальний, декларативний характер, оскільки для реалізації гіперконвергентного кластера потрібно зовнішня мережна інфраструктура, наявність якої мається на увазі як щось саме собою що розуміє.

Використання конвергентної інфраструктури веде до різкого збільшення обсягів трафіку в напрямку схід – захід. Донедавна самим вузьким місцем були системи зберігання. Рішенням цієї проблеми стала поява твердотільних накопичувачів, але в результаті потенційно вузьким місцем виявилася мережа. Більшість постачальників гіперконвергентних рішень успішно справляються із трафіком схід – захід усередині модулів. Деякі з них рекомендують ту або іншу архітектуру для міжз'єднанння й нерідко включають небрендовані комутатори із передвстановленої ОС (white box) у кожний модуль. Однак це не означає, що з'єднання між вузлами масштабуються аналогічним образом. Гіперконвергентна інфраструктура вимагає зміни багатьох традиційних мережних технологій.

Міжз'єднання вузлів є потенційно слабким місцем гіперконвергентних рішень. Постачальники фокусуються на обчисленнях і зберіганні, обіцяючи лінійне масштабування шляхом додавання вузлів. Гіперконвергентне програмне забезпечення гарне справляється з інтеграцією нових вузлів, але чи підходить для успішного рішення цього завдання фізичний рівень? Міжз'єднання вузлів здійснюється на основі вже наявної мережної інфраструктури. Вузли кластера взаємозалежна, тому будь-яка несправність мережної карти, комутатора або сполучних кабелів може негативно вплинути на функціонування всього кластера. У деяких випадках це не представляє проблеми, оскільки в ЦОДі вже є швидка й надійна мережа. Однак набагато більше ймовірність того, що існуюча мережа не здатна підтримувати міжз'єднання з низькою затримкою, що потрібно для гіперконвергентних рішень.

Іноді необхідна кабельна й мережна інфраструктура попросту відсутня, і підключити черговий вузол нікуди. Тим часом часто дуже мало часу при реалізації приділяється плануванню різних компонентів мережної

інфраструктури й міжз'єднаннню вузлів між собою. Як показує весь попередній галузевий досвід, у багатьох компаніях, особливо невеликих, прагнуть використовувати як можна більше дешеві кабельні компоненти, недорогі мережні карти й бюджетні комутатори. Але чи зможе така інфраструктура забезпечити надійні високошвидкісні з'єднання між вузлами кластера?

Установка комутаторів з високошвидкісними портами (10G+) у верхній частині стійки або наприкінці ряду проблеми не вирішує: підвищення продуктивності недостатньо, крім цього мережа повинна забезпечити контроль за створенням з'єднань і виділенням пропускної здатності. Програмно обумовлені фабрики, що комутуються, надають необхідну гнучкість і засоби автоматизації для керування трафіком між модулями, однак це веде до значного ускладнення мережної інфраструктури, що входить у суперечність із однієї з головних цілей HCI – спрощенням використання IT. Таким чином, деякі відсутні в модулях HCI спеціалізовані функції керування мережею прийде поки відтворювати на рівні агрегації або ядра.

Якщо багатьом постачальникам гіперконвергентної інфраструктури ще має бути знайти ефективний спосіб створення міжз'єднаннний, то від такого мережного вендора, як Cisco, можна було очікувати, що при випуску своєї гіперконвергентної системи HyperFlex він заздалегідь подбає про пошук підходящого рішення. І дійсно, в архітектурі HyperFlex, як і в USC, на якій вона базується, за організацію всіх з'єднань відповідають два комутатори Fabric Interconnect 6248s (серію 6300 з портами 40 Гбіт/с в Cisco уважають поки надлишкової для створення HCI). Це забезпечує єдине керування як мережними з'єднаннями, так і гіперконвергентними й обчислювальними вузлами (у кластер HyperFlex можна додавати блейди). Як затверджує виробник, після розгортання кластера його можна масштабувати до максимального розміру без необхідності вносити зміни в мережу.

Міжз'єднання вузлів – не єдині труднощі, з якої можуть зштовхнутися користувачі гіперконвергентних рішень. Список можливих проблемних чи місць

не більше переліку потенційних вигід. Однак до декларуємих недоліків треба ставитися не менш критично, ніж до того що заявляється перевагами. При цьому варто пам'ятати принаймні три речі: по-перше, часто недоліки є продовженням переваг (наприклад, зміна організаційної структури у зв'язку з появою IT-фахівців-універсалів нерідко викликає невдоволення інших співробітників IT-відділів); по-друге, деякі проблеми, такі як побоювання щодо можливих ризиків (втрата всього кластера через помилку в програмному забезпеченні), носять імовірнісний характер і зовсім не обов'язково повинні реалізуватися; по-третє, відмічувані недоліки не скасовують і не затьмарюють реальних переваг (на залежність від вендора можна піти заради скорочення витрат і інших вигід гіперконвергенції).

Проблема інтерконекта може бути віднесена до більше широкої проблеми потенційних архітектурних обмежень гіперконвергентних рішень. Об'єднання не занадто більших груп жорстких дисків і твердотільних накопичувачів, що перебувають на різних серверах, породжує інші труднощі, що не виникають при використанні традиційних зовнішніх масивів, коли одна система містить безліч дисків. Зокрема, програмне забезпечення повинне забезпечувати ефективне подання розподілених фізичних ресурсів зберігання як єдиного логічного цілого. Наскільки ефективно вирішується це завдання, особливо при масштабуванні кластера понад декілька вузлів, можна судити тільки за результатами практичного застосування відповідних рішень.

Програмному забезпеченню зберігання, як у випадку VSA, доводиться конкурувати з іншими віртуальними машинами й процесами за процесорний час (за деякими оцінками, воно споживає до 20-30% процесорного часу), так що продуктивність уведення-виводу може значно варіюватися. Оскільки багато гіперконвергентних рішень з'явилися зовсім недавно і є досить складними по своєму внутрішньому пристрої, можна апріорі затверджувати, що в них напевно є ті або інші помилки й баги. Програмне забезпечення розподілене між всіма вузлами й взаємодіє з дисками й накопичувачами за власними правилами. Якщо

воно раптом перестане функціонувати, то це чревате втратою всієї інфраструктури й всіх даних. Таким чином, мова йде про потенційну крапку загальносистемної відмови.

Постачальники гіперконвергентних пристроїв затверджують, що вони використовують стандартні компоненти серверів і зберігання, і в цьому змісті пропоноване рішення є відкритим. На жаль, деякі з них не задіють внутрішній потенціал гіперконвергенції для зниження залежності від вендора, а навпаки, вживають заходів для прив'язки клієнтів до своїх рішень, наприклад за рахунок використання нестандартних компонентів. У результаті доводиться все встаткування купувати в однієї компанії й багато в чому залежати від її надійності. Частково проблема вирішується шляхом використання окремого програмного рівня, що може бути розгорнуть на будь-якому встаткуванні на вибір замовника. Поряд з тим, що при цьому губиться ряд переваг гіперконвергентних пристроїв, зокрема оптимальне припасування програмного й апаратного забезпечення, сумніву в надійності постачальника, що особливо недавно з'явився на ринку, залишаються.

Можливо, головною перешкодою на шляху гіперконвергенції є людський фактор: широке поширення нового підходу вимагає серйозних організаційних змін – від моделі закупівель до структури IT-відділу. У перспективі широке використання гіперконвергентних рішень здатно привести до повної відмови від традиційних систем зберігання (хоча це зовсім не обов'язково, тому що необхідно здійснювати підтримку безлічі «успадкованих» додатків). Тим часом у багатьох організаціях зони відповідальності адміністраторів традиційно розділені, а виходить, ті, хто обслуговує системи зберігання, можуть сприймати HCI як погрозу своєму положенню й своїй роботі, що може викликати опір з боку співробітників. Звичайно, цей фактор здатний виявитися в першу чергу у великих компаніях, оскільки в невеликі й середніх IT-фахівці найчастіше є універсалами й спрощення керування будуть тільки привітати.

Гіперконвергентні пристрої розглядаються насамперед як рішення для невеликих компаній, і багато постачальників позиціонують їх саме так. Однак у звіті «Стан ринку гіперконвергентної інфраструктури», підготовленому ActualTech Media, затверджується: чим більше організація, тим вище ймовірність того, що вона вже використовує гіперконвергентні рішення. Правда, великі компанії здобувають їх переважно для установки у своїх філіях і віддалених офісах.

Основним додатком, для підтримки якого споконвічно використовувалися гіперконвергентні пристрої, були тонкі клієнти – віртуальні робочі місця (Virtual Desktop Infrastructure, VDI). Як відзначають експерти ActualTech Media, хоча VDI залишається важливим драйвером для впровадження HCI, сьогодні вона усе більш широко застосовується й для інших робочих навантажень. Гіперконвергентна інфраструктура споконвічно розрахована на підтримку віртуальних машин, а виходить, і будь-яких навантажень у вигляді ВМ.

Так, крім підтримки VDI, в SimpliVity виділяють цілий ряд сценаріїв застосування для гіперконвергентних пристроїв, у тому числі підтримку критичних для бізнесу додатків (Tier I application). Серед інших можливих застосувань називаються створення середовища для тестування й розробки, модернізація резервного копіювання, відновлення після аварій, спрощення адміністрування віддалених площадок, плавна модернізація успадкованої інфраструктури й, нарешті, консолідація серверів і центрів обробки даних.

Багато які аналітики сумніваються в тому, що HCI у вигляді гіперконвергентних пристроїв підходить для підтримки критичних для бізнесу додатків і є адекватним рішенням для модернізації ЦОДа. Такі додатки, як ERP і CRM, прийде переписувати або оптимізувати для роботи в розподіленому середовищі. Тому для їхньої підтримки більше підходять конвергентні (а не гіперконвергентні) рішення із традиційною архітектурою. У ЦОДах можуть гостріше виявитися такі слабкі місця гіперконвергентних пристроїв, як міжз'єднання вузлів (підтримка мережі) і обмежена масштабованість. Однак з

появою гіперконвергентних архітектур EVO:RAIL від VMware і VxRack від EMC інфраструктура HCI може одержати більше широке поширення в ЦОДах.

Як би те не було, інтерес до гіперконвергентних рішень стрімко росте, про що свідчать обсяги продажів. Почасти таку їхню популярність (поки за рубежем) можна пояснити тим, що далеко не всі замовники готові перенести робочі навантаження в публічну хмару. Тим часом багато компаній зацікавлені в реалізації хоча б деяких переваг хмарних технологій у власній інфраструктурі, і гіперконвергентні рішення допомагають їм це зробити

3.2 Розробка структурної схеми

Якщо говорити про гіперконвергенцію взагалі, то це споконвічна інтеграція в одне ціле двох і більше різнорідних компонентів. Ключовим словом тут є «споконвічна»: для досягнення найбільшого ефекту компоненти повинні бути інтегровані з нуля, а не просто об'єднані один з одним. Так про які ж компоненти мова йде?

У поточному IT-дискурсі часто маються на увазі гіперконвергентні системи зберігання, однак для стислості згадування про їх опускається. Таким чином, на самому елементарному рівні можна говорити про з'єднання обчислювальних потужностей і ресурсів зберігання в одному пристрої. Однак таке розхоже розуміння (фактично воно припускає звичайний сервер) жорстко критикується основними галузевими гравцями, оскільки вендори цілком обґрунтовано вважають, що воно не відбиває всієї повноти картини й до того ж фокусується на окремому пристрої, випустити з уваги відповідну інфраструктуру (Hyper Converged Infrastructure, HCI). Гіперконвергентні рішення являють собою щось більше, ніж об'єднання в загальний пул обчислювальних ресурсів і засобів зберігання.

Однак, як звичайно буває, особливо коли ринок тільки формується, кожний з постачальників розуміє модний термін так, як йому вигідно. Останнє

твердження зовсім не має на увазі негативних коннотацій – кожний прагне підкреслити найбільш сильні сторони своїх рішень. Наприклад, деякі постачальники акцентують увагу на убудованому захисті даних – підтримці функції резервного копіювання, відновлення даних, тиражування, дедуплікації й стиску, однак аналітики Gartner уважають наявність таких можливостей опціональним для гіперконвергентних рішень, які вони розглядають у якості однієї з різновидів інтегрованих систем.

У вузькому розумінні гіперконвергенція – це програмно обумовлений підхід до керування зберіганням, що поєднує засобу зберігання, обчислювальні потужності, мережні ресурси й технології віртуалізації в одному фізичному пристрої, керування яким здійснюється як єдиною системою (визначення techtarget.com). У широкому змісті гіперконвергенція – це об'єднання обчислювальних потужностей, мережних ресурсів і засобів зберігання з використанням стандартних компонентів x86 у якості загальних будівельних блоків, при цьому інтелект більше не прив'язаний до окремих фізичних пристроїв – всі функції центра обробки даних виконуються як програмне забезпечення на гіпервізорі (трактування VMware).

Якщо перше визначення фокусується на окремому пристрої й зберіганні, то друге по своїй суті еквівалентно програмно обумовленому центру обробки даних. Фактично обоє визначення задають напрямок розвитку гіперконвергентних рішень: від інтегрованих пристроїв, поєднуваних у кластер, до повномасштабного програмно обумовленому ЦОДу. Однак вони опускають ряд важливих конкретних деталей – наприклад, відмова від окремої мережі зберігання даних (Storage Area Network, SAN).

Гіперконвергенція пропонує повноцінну альтернативу домінуючої моделі використання виділених систем зберігання. Вона припускає відмову від окремого рівня зберігання (див. рис. 3.1), що розглядається як одне із ключових переваг HCI.

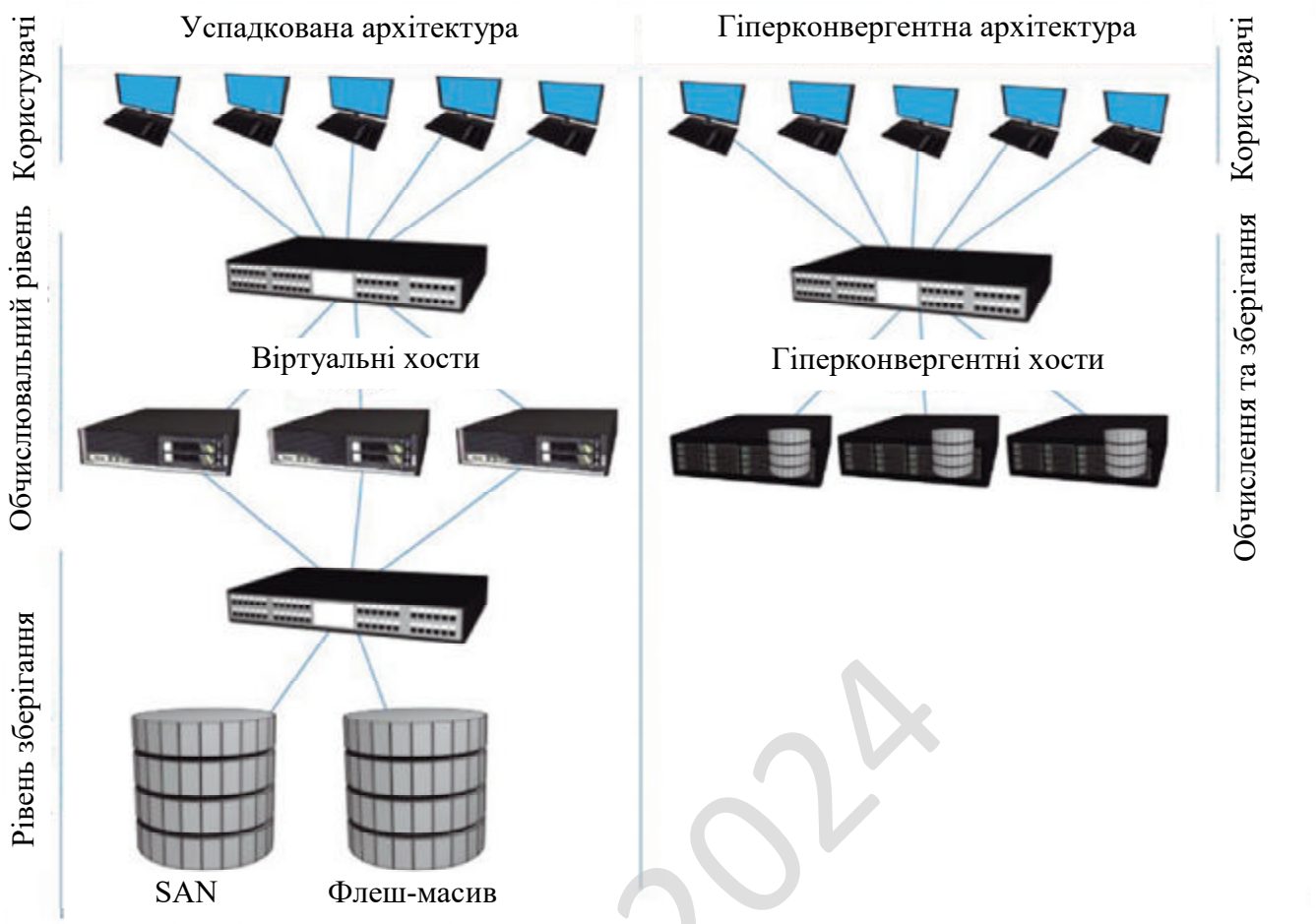

Рисунок 3.1 – Структурна схема системи

Возз'єднання обчислювальних ресурсів і ресурсів зберігання в одному пристрої може породити побоювання про повернення в 90-е роки, коли переважали системи, що підключаються прямо, зберігання (Directed Attached Storage, DAS). Однак, на відміну від колишніх архітектур, гіперконвергентна не приводить до утворення ізольованих острівців зберігання. Щоб кожний сервер міг скористатися ресурсами зберігання на інших системах, вона опирається на програмно обумовлені рішення в області зберігання (Software-Defined Storage, SDS).

Програмно обумовлений підхід припускає триетапний процес: абстрагування, об'єднання в пул і автоматизація. Завдяки програмному рівню абстрагування фізичних ресурсів, всі сервери фактично перетворюються у вузли

єдиного кластера. Абстрагування фізичних компонентів окремих серверів дозволяє об'єднати в загальний пул їхні ресурси зберігання, які потім представляються кластеру як зв'язне ціле. Автоматизація дозволяє спростити й прискорити керування, реалізуючи різного роду високорівневі функції.

Гіперконвергентні системи звичайно складаються з декількох фізичних модулів (вузлів), поєднуваних у горизонтально масштабований кластер. Кожний з них містить обчислювальне ядро, ресурси зберігання, мережні компоненти й, як правило, гіпервізор. Наявність останнього формально не обов'язково, але він є у всіх основних продуктах відомих вендорів, причому нерідко на вибір пропонуються два гіпервізора або більше.

Окремий пристрій (appliance) має від одного до чотирьох вузлів, кожний з яких являє собою самостійний сервер із процесором і пам'яттю в загальному шасі. Гіперконвергентні кластери звичайно містять від 4 до 64 вузлів, хоча деякі виробники не вказують конкретних меж масштабованості. Щоб вузли могли спільно використовувати ресурси зберігання, застосовуються програмне забезпечення для створення віртуальної мережі зберігання або кластерна файлова система.

Програмне забезпечення для реалізації гіперконвергентної інфраструктури може пропонуватися як окремо, так і передвстановленим на фізичні пристрої (appliance). Кожний підхід має свої плюси й мінуси. У випадку покупки ПЗ недолік полягає в тому, що замовникові прийде окремо здобувати встаткування й потім самостійно його встановлювати, а перевагою є можливість самостійно вибирати апаратне забезпечення (на відміну від придбання готових пристроїв із передвстановленим ПЗ). Основну частину пропонованих на ринку рішень становлять повністю готові самодостатні обчислювальні системи, хоча деякі стартапи роблять ставку на пропозицію відповідні ПЗ (правда, найчастіше замовники купують його не прямо, а вже передвстановленим на сервери, чим займаються OEM-партнери розроблювачів або інтегратори).

Найчастіше постачальники (насамперед із числа тих же стартапів) розробляють своє власне програмне забезпечення для реалізації функцій зберігання й керування або ці функції виконуються як віртуальна машина на використовуваному гіпервізорі. Так, у багатьох продуктах використовується віртуальний пристрій зберігання (Virtual Storage Appliance, VSA) для об'єднання ресурсів зберігання в загальний пул. Інший підхід складається у використанні платформи VMware EVO:RAIL, свого роду референсної архітектури, у якій функції зберігання й керування інтегровані у відповідний гіпервізор.

Дозволяючи позбутися від виділеної мережі зберігання, гіперконвергентна інфраструктура фактично реалізує віртуальну мережу зберігання (Virtual SAN, VSAN). Звичайно така SAN припускає використання VSA у вигляді ВМ. Віртуальний пристрій зберігання надає функції контролера зберігання для гіпервізора в кластері. Ресурси зберігання фізичного вузла надаються VSA. Група VSA спільно створює віртуальну мережу зберігання. Виключенням із цього підходу є реалізація VSAN компанії VMware. Сервіси зберігання містяться безпосередньо в ядро vSphere, так що рішення повністю інтегрується в vSphere. Однак необхідне використання vSphere 5.5 або більше пізніх версій.

3.3 Розробка функціональної схеми

Для організації гіперконвергенції IT-інфраструктури використовується розподілені системи передачі даних (СПД). Розглянемо загальні принципи структурної й функціональної організації розподілених СПД системи гіперконвергенції IT-інфраструктури, завдання проектування й різні методи дослідження СПД системи гіперконвергенції IT-інфраструктури, а також проблеми розрахунку моделей СПД системи гіперконвергенції IT-інфраструктури. Виконаємо огляд моделей і методів дослідження СПД системи гіперконвергенції IT-інфраструктури. При цьому моделі СПД системи гіперконвергенції IT-інфраструктури різних класів будуються на основі моделей

масового обслуговування. Розглянуто класифікації систем (СМО) і мереж (СеМО) масового обслуговування. СМО використовуються як моделі каналів зв'язку, а СеМО – як моделі СПД системи гіперконвергенції ІТ-інфраструктури. Для дослідження СПД системи гіперконвергенції ІТ-інфраструктури використовуються аналітичні (точні, наближені й евристичні) і імітаційні методи. Точні аналітичні методи використовуються для експонентних моделей, наближені методи використовуються для розрахунку неекспонентних моделей, розрахунок яких точними методами не можливий. Імітаційне моделювання реалізується в середовищі, використовуваному для дослідження складних систем з дискретним характером функціонування.

Розглянемо завдання приведення неоднорідного потоку до однорідного, розрахунку потоків пакетів у КЗ і оцінки пропускних здатностей каналів зв'язку СПД системи гіперконвергенції ІТ-інфраструктури із урахуванням реальних особливостей.

Виявлено основні відмінності традиційного завдання оптимізації пропускних здатностей каналів зв'язку з однорідним навантаженням, припущеного Клейнроком, від завдання оптимізації з неоднорідним навантаженням, розглянутого в роботі.

Неоднорідність потоків пакетів описується параметрами різних типів повідомлень, до них відносяться:

– середня довжина повідомлень;

– середня зовнішня інтенсивність повідомлень від користувачів.

Крім того, при відомості неоднорідного потоку до однорідного необхідно враховувати:

– топологію СПД системи гіперконвергенції ІТ-інфраструктури;

– метод маршрутизації;

– модель взаємодії користувачів;

– розподіл прикладних програм і наборів даних по вузлах мережі й т.д.

Розрахунок потоків пакетів у каналах СПД системи гіперконвергенції IT-інфраструктури проводиться в три етапи.

На першому етапі виконується розрахунок інтенсивностей $\lambda_{i,j}$ пакетів з вузла i у вузол j шляхом множення зовнішніх інтенсивностей $\Lambda_{i,j}$ повідомлень із вузла i у вузол j на число пакетів $k = 1,2,...$ в одному повідомленні:

$$\lambda_{i,j} = k\Lambda_{i,j} \qquad (i, j = \overline{1,n}; \quad i \neq j), \tag{3.1}$$

На другому етапі розрахунок інтенсивностей пакетів у КЗ виконується на основі обчислених на першому етапі зовнішніх інтенсивностей потоків пакетів, заданих таблиць маршрутизації й імовірності передачі по основному шляху:

$$\lambda_{(k,l)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_{i,j} X_{(k,l)}^{(i,j)} \qquad (k, l = \overline{1,n}; \quad k \neq l), \tag{3.2}$$

де $X_{(k,l)}^{(i,j)}$ – частка пакетів $(0 \le X_{(k,l)}^{(i,j)} \le 1)$, що проходять по каналу (k, l), причому повинне виконуватися умова збереження потоку в мережі:

$$\sum_{k=1}^{n} X_{(k,l)}^{(i,j)} - \sum_{k=1}^{n} X_{(l,k)}^{(i,j)} = \begin{cases} -1, & l=i; \\ 0, & l \neq i; \\ 1, & l=j. \end{cases} \tag{3.3}$$

Для визначення $X_{(k,l)}^{(i,j)}$ використовується зворотна рекурсивна процедура, визначаюча розподіл інтенсивностей потоків на парі (i – вузол-джерело, j – вузол-призначення) і підсумовуюча їх при розгляді всіх пар вузлів. Ця процедура реалізується в такий спосіб. Скануються всі вузли, через які пакети можуть пройти від вузла-джерела до вузла-призначення з урахуванням таблиць маршрутизації, і будується двійкове дерево, причому для кожного вузла розглядаються два шляхи передачі пакета: основний і альтернативний. СПД системи гіперконвергенції IT-інфраструктури відрізняється від двійкового дерева тим, що її топологія довільна. Для того, щоб усунути ситуацію, коли пакети можуть циркулювати в мережі нескінченно, використовується маскування вузлів, які вже сканувалися.

При виклику рекурсивної процедури частка пакетів у кожному з каналів зв'язку розраховується за правилом:

1. Якщо наступний розглянутий вузол є основним, і він не маскований, то частка пакетів у цьому каналі збільшується на величину, рівну добутку ймовірності передачі пакетів по основному шляху на частку пакетів попереднього КЗ.

2. Якщо наступний розглянутий вузол є альтернативним і він не маскований, то частка пакетів у цьому каналі збільшується на величину, рівну добутку ймовірності передачі по альтернативному шляху на частку пакетів попереднього каналу зв'язку.

Рекурсивна процедура викликається тільки тоді, коли наступний розглянутий вузол у процесі сканування не маскований і не збігається з вузла-призначення. Зворотний механізм у рекурсивному алгоритмі реалізується за допомогою масиву маскування – після виклику рекурсивної процедури якоїсь пари (вузол-джерело, вузол-призначення) потрібно відзначити вузол-джерело як немаскований.

Якщо основний шлях розглянутого вузла збігається з вузлом-призначення й альтернативний шлях цього вузла є маскованим вузлом, то до частки пакетів основного каналу зв'язку додається частка пакетів альтернативного, рівна добутку ймовірності передачі пакета по альтернативному шляху на частку пакетів попереднього каналу зв'язку, тому що пакети не можуть повторно попадати в той самий вузол СПД системи гіперконвергенції ІТ-інфраструктури.

На третьому етапі виконується розрахунок коефіцієнтів передач для кожного каналу зв'язку:

$$\alpha_{(k,l)} = \lambda_{(k,l)} / \lambda_{вн.}, \tag{3.4}$$

де $\lambda_{вн.} = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{i,j}$ – сумарна інтенсивність надходження пакетів у СПД системи гіперконвергенції ІТ-інфраструктури.

Після розрахунку інтенсивностей пакетів у КЗ для кожного типу повідомлень із використанням вищенаведеного алгоритму неоднорідний потік

пакетів зводиться до однорідного шляхом підсумовування інтенсивностей потоків пакетів у КЗ.

Даний алгоритм розрахунку інтенсивностей пакетів вбудований у програмний комплекс і являє собою ключовий етап для рішення завдання оцінки пропускних здатностей на основі методу Лагранжа.

Представлено завдання оптимізації пропускних здатностей каналів зв'язку на основі методу Лагранжа при обмеженні на середній час доставки пакетів у СПД системи гіперконвергенції ІТ-інфраструктури і на вартість СПД системи гіперконвергенції ІТ-інфраструктури. Наведено алгоритм вибору дискретних значень пропускних здатностей на основі отриманих безперервних значень.

Досліджуємо вплив процесів передачі даних на характеристики функціонування СПД системи гіперконвергенції ІТ-інфраструктури.

Виконано численні імітаційні експерименти для оцінки погрішностей аналітичних методів розрахунку моделі каналу зв'язку, що базуються на наближених формулах розрахунку часових характеристик каналу зв'язку.

Експерименти дозволяють зробити вивід про те, що наближена формула розрахунку часових характеристик системи G/M/1, використовуваної як модель КЗ, дає невелику погрішність для потоку, коефіцієнт варіації (КВ) якого менше 1 (у більшості випадків вона менше 20% при завантаженні більше 0.3). Коефіцієнт варіації – це відношення середньоквадратичного відхилення до середньої арифметичної, який показує ступінь відхилення отриманих значень. Для потоку, КВ якого більше 1, при збільшенні завантаження зменшення цієї погрішності близько до лінійного. При великому завантаженні каналу зв'язку погрішність розрахунку характеристик передачі пакетів перебуває в межах 20%, прийнятних для інженерних розрахунків. Численні імітаційні експерименти дозволили виконати детальне дослідження характеристик системи й зробити наступні виводи:

1. Чим менше завантаження досліджуваної системи, тим більшу погрішність дає наближена формула.

2. При зміні завантаження системи за рахунок варіювання середнього часу обслуговування (при фіксованому середньому часі надходження) і за рахунок варіювання середнього часу надходження (при фіксованому середньому часі обслуговування) експерименти дають однаковий результат, тобто результати розрахунку погрішностей характеристик не залежать від того, за рахунок якого параметра змінюється завантаження.

Формула розрахунку коефіцієнтів варіацій вихідного потоку каналу зв'язку дає в більшості випадків верхню границю для коефіцієнта варіації вихідного потоку заявок із системи G/G/1, причому відносна погрішність формули перебуває в прийнятних межах (менше 20%) при малому завантаженні системи (від 0 до 0.4) для коефіцієнтів варіації вхідного потоку заявок більше 1, а також при великому завантаженні системи (від 0.8 до 1) для випадків КВ менше 1.

Розглянуто різні підходи по оцінці впливу обмеженої довжини пакетів на характеристики каналів окремо й у мережі в цілому. Показано, що для нормування усіченого експонентного розподілу обидва методи (імітаційний і комбінація аналітичного і імітаційного) дають однакові результати. Для експонентного розподілу обидва способи обмеження пакетів ліворуч і праворуч (усіканням і зсувом) дають практично однакові часові характеристики передачі даних. Однак, для гіперекспонентного розподілу спосіб «зі зсувом» дає значення часу очікування значно більші, ніж для випадку «з усіканням».

Для мережі Ethernet відносне відхилення результатів розподілу зі зсувом у порівнянні із традиційними становить десятки відсотків, і це відхилення збільшується зі збільшенням завантаження каналу й може перевищувати 100% для гіперекспоненційного розподілу. Традиційні розподіли можуть використовуватися при моделюванні мережі TCP/IP через велике відношення між максимальною l_{max} (мінімальною – l_{min}) і середньою $l_{ср}$ довжиною пакетів.

Розходження в результатах для СПД системи гіперконвергенції IT-інфраструктури, у загальному випадку, може становити десятки відсотків, що

робить необхідним облік обмеженої довжини пакетів у процесі системотехнічного проектування СПД системи гіперконвергенції ІТ-інфраструктури.

Проведено аналіз впливу характеру трафіку в СПД системи гіперконвергенції ІТ-інфраструктури і тривалості передачі пакетів у каналах зв'язку на характеристики функціонування СПД системи гіперконвергенції ІТ-інфраструктури. Коли всі КВ розподілу потоків і часу передачі пакетів у каналах збільшуються до одиниці, всі часові характеристики неекспонентної мережі прагнуть до часових характеристик експонентної мережі, тобто можна говорити, що експонентна мережа розглядається як верхня межа в цьому випадку. Коли значення КВ різних розподілів збігаються, характеристики мережі майже збігаються. У мережах з гіперекспонентним розподілом часу обслуговування (передачі) пакетів у каналах СПД системи гіперконвергенції ІТ-інфраструктури забезпечується менше значення часу доставки пакетів у порівнянні з Гамма-розподілом, причому при більших значеннях КВ це розходження може становити десятки відсотків. При більших значеннях КВ інтервалів часу між вступниками в мережу пакетами й часу їхньої передачі по каналах зв'язку необхідно враховувати більше високі моменти розподілів, зокрема третій момент, що обумовлює значне розходження часу доставки пакетів у СПД системи гіперконвергенції ІТ-інфраструктури.

Розглянуто вплив третього моменту гіперекспонентного (із КВ більше 1) і гіпоекспоненційного (із КВ менше 1) розподілу часу обслуговування (передачі пакетів у каналі) на характеристики системи. Для гіперекспонентного закону з одними й тим же другим моментом можна одержати різні значення третього моменту, і чим більше значення другого моменту, тим більше значення третього моменту і їхнє розходження. Залежності часу очікування систем $H_2/M/1$, $H_2/D/1$, $H_2/Г_2/1$, (де $Г_2$ – гамма-розподіл з коефіцієнтом варіації 2) від величин третього моменту гіперекспонентного потоку для завантаження 0.5 і 0.9, показані. Чим більше значення третього моменту, тим менше час очікування в системі. Однак,

коли завантаження дорівнює 0.5, час очікування різних систем різко падає долілиць при незначній зміні третього моменту на початку (коли значення третього моменту <100) і майже не змінюється, коли ці значення більше 100. Для випадку, коли завантаження дорівнює 0.9, час очікування систем більше «лінійно» залежить від третього моменту. Для всіх випадків, час очікування сильно залежить від третього моменту й дає велике розходження (більше 100% при завантаженні 0.5).

Дослідження впливу третього моменту поступаючих даних в систему $E_x/G/1$ потоку на час очікування дає результати, показані впливом третього моменту гіпоекспоненційного вступника потоку з різними КВ на відносне розходження між максимальним і мінімальним значеннями часу очікування декількох систем при $k = 50$.

Функціональна схема розробленої системи зображена на рисунку 3.2. Представимо опис розробленого програмного комплексу для дослідження СПД системи гіперконвергенції IT-інфраструктури із неоднорідним потоком повідомлень на основі аналітичного й імітаційного моделювання.

Завдання проектування мережі складається у визначенні пропускних здатностей КЗ із урахуванням неоднорідності трафіку, різноманіття топологій, алгоритмів маршрутизації, варіантів розміщення прикладних програм і наборів даних по вузлах мережі, способів взаємодії користувачів мережі.

Аналітична модель СПД системи гіперконвергенції IT-інфраструктури будується у вигляді розімкнутої СеМО для будь-якої топології мережі, що задається аналітично або графічно. На основі аналітичної моделі вирішується завдання визначення пропускних здатностей КЗ у розподілених СПД системи гіперконвергенції IT-інфраструктури при обмеженнях на час доставки пакетів або на вартість мережі.

Програма дозволяє розрахувати характеристики СПД системи гіперконвергенції IT-інфраструктури, такі як:

– пропускні здатності КЗ;

– час затримки пакетів при передачі по кожному каналу й у мережі в цілому;

– завантаження кожного КЗ;

– інтенсивності потоків пакетів у каналах зв'язку;

– імовірності передачі пакетів від користувачів у мережу, між каналами й з мережі до користувачів.

Програма дозволяє варіювати отримані характеристики, такі як:

– пропускні здатності;

– час передачі по КЗ;

– завантаження КЗ;

– час доставки пакетів у СПД системи гіперконвергенції ІТ-інфраструктури;

– вартість СПД системи гіперконвергенції ІТ-інфраструктури для одержання необхідних результатів.

Крім того, вона дозволяє вибрати значення пропускних здатностей з дискретного ряду значень, знайдених із традиційної оптимізації безперервних значень пропускних здатностей.

Імітаційна модель СПД системи гіперконвергенції ІТ-інфраструктури генерується автоматично на основі розрахунку характеристик аналітичної моделі у вигляді розімкнутої експонентної СеМО. Отримані характеристики аналітичної моделі використовуються як параметри імітаційної моделі СПД системи гіперконвергенції ІТ-інфраструктури, що представляє собою розімкнуту СеМО, вузли якої відповідають каналам зв'язку, а заявки – пакетам у СПД системи гіперконвергенції ІТ-інфраструктури. Число джерел заявок в імітаційній моделі дорівнює числу вузлів у СПД системи гіперконвергенції ІТ-інфраструктури, при цьому інтенсивності надходження заявок у моделі визначаються як зовнішні інтенсивності пакетів від користувачів до вузлів СПД системи гіперконвергенції ІТ-інфраструктури відповідно.

Рисунок 3.2 – Функціональна схема системи

Аналітична модель, реалізована в програмі, дозволяє розрахувати наступні параметри для імітаційної моделі.

1. Імовірність передачі пакетів від користувача *j* до каналу *k*.

2. Імовірність передачі пакетів від каналу *k* до каналу *h*.

3. Імовірність передачі пакетів від каналу *k* до користувача *j*.

4. Середній інтервал часу між вступниками пакетами від користувача *j* у СПД системи гіперконвергенції ІТ-інфраструктури.

5. Середній час передач пакетів у каналах зв'язку *k*.

Крім **розрахованих параметрів** для імітаційної моделі СПД системи гіперконвергенції ІТ-інфраструктури додатково необхідно **задати наступні параметри**:

а) закони розподілів інтервалів часу між вступниками пакетами від користувачів у СПД системи гіперконвергенції ІТ-інфраструктури із коефіцієнтами варіації цих розподілів;

б) закони розподілів часу передачі пакетів у КЗ із коефіцієнтами варіації цих розподілів.

Відзначимо, що імітаційна модель СПД системи гіперконвергенції ІТ-інфраструктури призначена для детального аналізу характеристик функціонування мережі, спроектованої в процесі аналітичного моделювання. При цьому, у випадку відмінності реального характеру процесів надходження пакетів у мережу або передачі пакетів по каналах зв'язку від експонентного, в імітаційній моделі передбачена можливість варіювання законів розподілу часу передачі (довжин обслуговування) пакетів у кожному із КЗ, а також законів розподілу інтервалів часу між вступними в мережу пакетами. Як такі закони в роботі використовувалися наступні розподіли: експонентний, детермінований; гіпоекспоненційний різного порядку й, відповідно, з різними коефіцієнтами варіації; рівномірний; експонентний з ненульовими зсувами; гіперекспонентний; Гамма-розподіл.

Аналогічно **розроблений засіб дослідження часових характеристик каналу зв'язку** шляхом генерування імітаційної моделі каналу зв'язку у вигляді СМО типу G/G/1 з можливістю зміни завантаження каналу й варіювання законів розподілу інтервалів часу між пакетами й часу передачі пакетів по каналі. Даний засіб дозволяє одержати значення часових характеристик каналу зв'язку й оцінити погрішність аналітичних методів розрахунку характеристик каналу зв'язку.

Сформульовано **методику дослідження розподіленої СПД системи гіперконвергенції ІТ-інфраструктури** із неоднорідним трафіком, що містить наступні етапи.

1. Підготовка вихідних даних для програмного комплексу:

– кількість вузлів мережі і їхнє взаємне розташування;

– навантаження СПД системи гіперконвергенції ІТ-інфраструктури, створювана користувачами при роботі із прикладними програмами, наборами даних і в процесі обміну повідомлень;

– обмеження на середній час доставки пакетів або на вартість СПД системи гіперконвергенції ІТ-інфраструктури;

– максимальна довжина пакета;

– вибір типу каналів і завдання вартісних коефіцієнтів КЗ.

2. Розрахунок оптимальних пропускних здатностей типових топологій СПД системи гіперконвергенції ІТ-інфраструктури при заданих наступних параметрах:

– типова топологія: зірка, кільце, дерево, повнозв’язна;

– модель взаємодії користувачів мережі: RDA (Remote Data Access), DBS (DataBase Server) і AS (Application Server);

– розподіл прикладних програм і наборів даних по вузлах СПД системи гіперконвергенції ІТ-інфраструктури;

– метод маршрутизації.

Після завдання цих параметрів розраховуються оптимальні значення пропускних здатностей КЗ, час передачі пакетів і завантаження каналів.

3. Розрахунок пропускних здатностей КЗ при розподілених топологіях.

4. Вибір найкращої топології СПД системи гіперконвергенції ІТ-інфраструктури, для якої обрана залежно від постановки завдання як критерій ефективності характеристика СПД системи гіперконвергенції ІТ-інфраструктури (середній час доставки пакетів або вартість СПД системи гіперконвергенції ІТ-інфраструктури) приймає найменше значення.

5. Варіювання параметрів і характеристик КЗ і СПД системи гіперконвергенції ІТ-інфраструктури і вибір дискретних значень пропускних здатностей.

6. Оцінка погрішностей аналітичних методів розрахунку характеристик КЗ.

7. Оцінка впливу характеру потоків пакетів і тривалості передачі даних на характеристики СПД системи гіперконвергенції ІТ-інфраструктури:

– вплив третього моменту розподілу вхідного потоку пакетів на характеристики КЗ;

– вплив довжини пакетів на характеристики каналів зв'язку й СПД системи гіперконвергенції ІТ-інфраструктури;

– вплив законів розподілів трафіку в мережах і часі передачі пакетів у КЗ на характеристики функціонування СПД системи гіперконвергенції ІТ-інфраструктури.

8. Дослідження СПД системи гіперконвергенції ІТ-інфраструктури при різних розміщеннях прикладних програм і наборів даних.

9. Детальний аналіз спроектованої СПД системи гіперконвергенції ІТ-інфраструктури, що припускає варіювання параметрів СПД системи гіперконвергенції ІТ-інфраструктури, у тому числі: довжини запитів і відповідей прикладних програм і наборів даних, імовірність передачі по основному шляху, способи маршрутизації й т.д.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації, також те, що при розробці програми слід надати особливу увагу модулю гіперконвергенції IT-інфраструктури.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні блоки можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

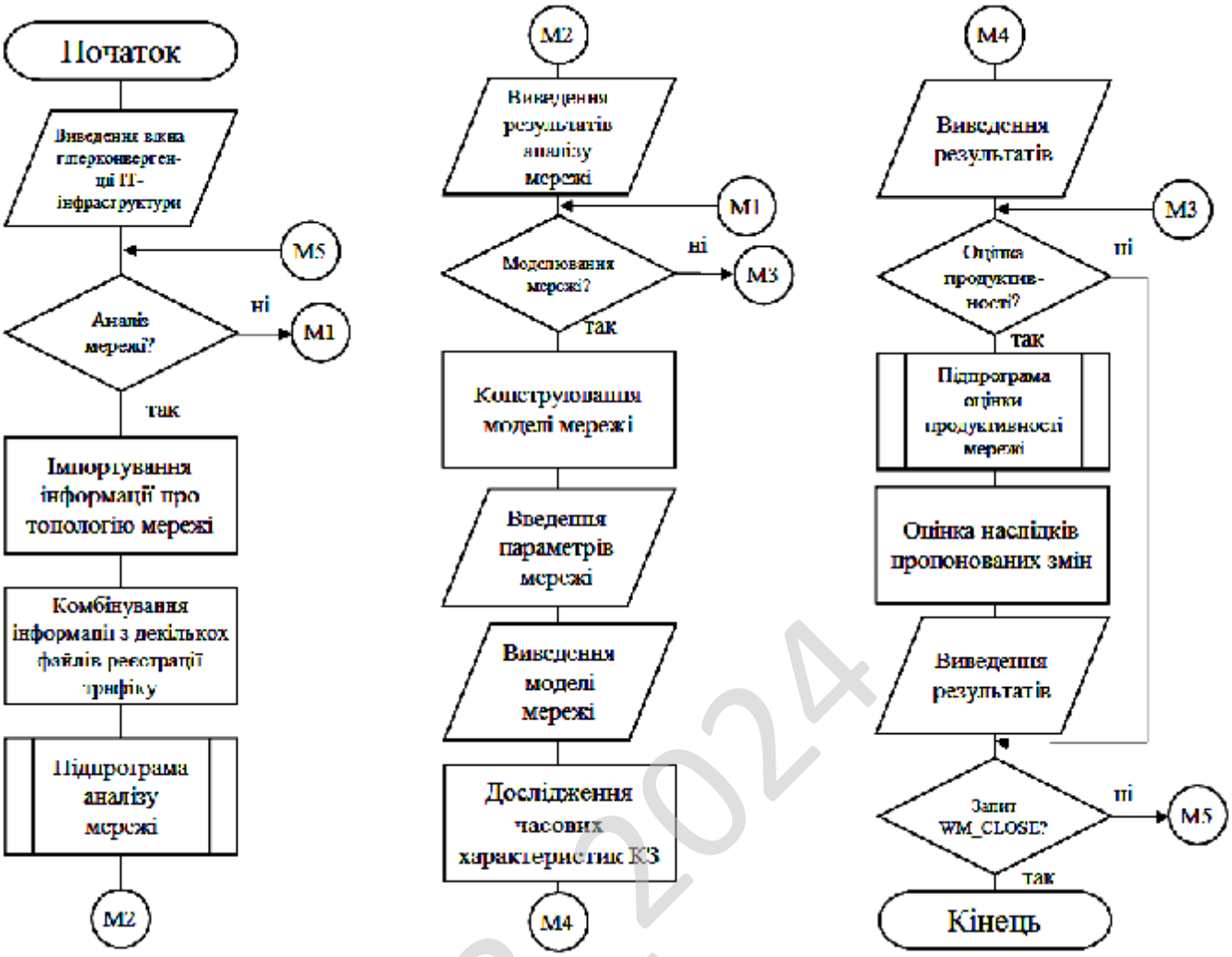

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

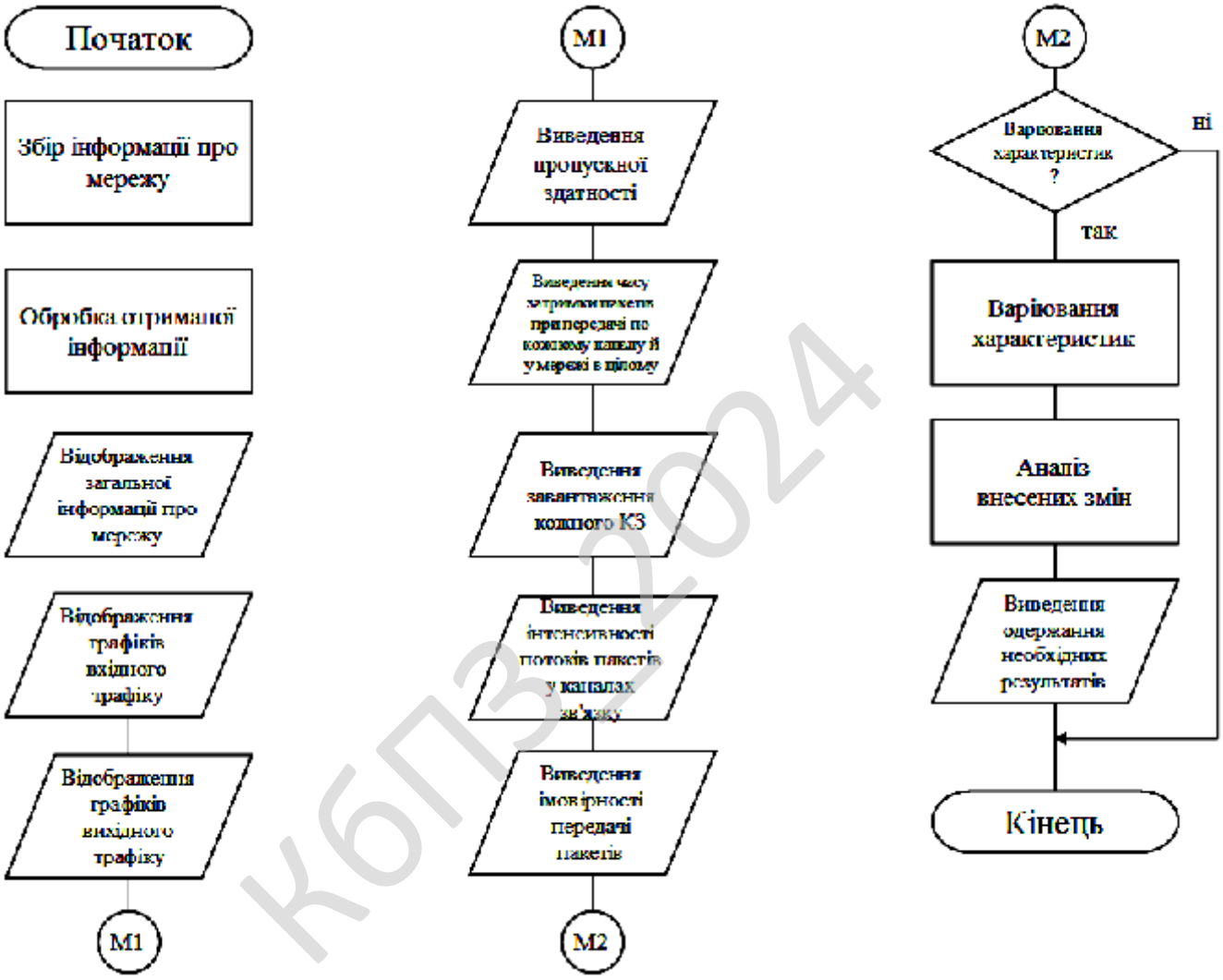

Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для

визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться

горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. **Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:**

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу. Переваги:

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

Управління вимогами це процес запису, аналізу, трасування, пріоритезації і узгодження вимог та контролю змін і доведення до їх зацікавлених сторін. Це безперервний процес протягом всього життя проекту. Вимога – якість, якій мають відповідати результати проекту (продукту або послуги).

Мета управління вимогами полягає в тому, щоб переконатися, що організація відповідає потребам і очікуванням своїх клієнтів, внутрішніх або зовнішніх зацікавлених сторін. Управління вимогами починається з аналізу і виявлення цілей і обмежень організації. Управління вимогами додатково включає в себе підтримку планування вимог, інтеграції вимог і організації роботи з ними (атрибути для вимог).

Управління вимогами передбачає спілкування між членами проектної групи і зацікавленими сторонами, і адаптацію до змін у вимогах протягом всього проекту. Щоб запобігти перетину поля одного класу вимог з іншим, постійні зв'язки між членами команди розробників є критичними. Наприклад, при розробці програмного забезпечення для внутрішнього використання у бізнесу можуть бути

настільки сильні потреби, що він може проігнорувати вимоги користувачів, або вважати, що створені сценарії використання покриють також і користувальницькі вимоги.

Відслідковування вимоги фактично означає документування всього життєвого циклу вимоги. Часто необхідно дізнатися першоджерело кожної вимоги. Для цього всі зміни вимог повинні бути задокументовані, щоб досягти стану повного відстеження. Відстежувати треба бути навіть використання реалізованих вимог.

Вимоги мають різні джерела, такі як ділова людина, що замовляє продукт, менеджер зі збуту і фактичний користувач. У всіх цих людей є різні вимоги до продукту. Використовуючи відслідковування вимог, реалізована в системі функція може бути простежена назад до людини або групі, яка замовляла її під час збору вимог. Ця особливість може, наприклад, використовуватися в процесі розробки для пріоритезації вимог, визначаючи, наскільки цінною є дана вимога для певного користувача.

Відслідковування може також використовуватися після розгортання продукту. Наприклад, коли вивчення використання системи показує, що якась функція не використовується, можна визначити навіщо вона була потрібна спочатку.

Завдання управління вимогами

На кожному етапі процесу розробки існують ключові методи і задачі пов'язані з управлінням вимогами. Для ілюстрації, розглянемо наприклад стандартний процес розробки з п'ятьма фазами: дослідженням, аналізом здійсненності, дизайном, розробкою та тестуванням і випуском.

Дослідження. Під час фази дослідження збираються перші три класи вимог від користувачів, бізнесу і команди розробників. У кожній області задають однакові питання: які цілі, які обмеження, які використовуються процеси та інструменти і так далі. Тільки коли ці вимоги добре зрозумілі, можна приступати до розробки функціональних вимог.

Тут необхідне застереження: незалежно від того, як сильно група намагається це зробити, вимоги не можуть бути повністю визначені на початку проекту. Деякі вимоги змінюються, або тому що вони просто не були знайдені спочатку, або тому що внутрішні чи зовнішні сили торкаються проекту в середині циклу. Таким чином, учасники групи повинні спочатку погодитися, що головна умова успіху – гнучкість у мисленні та діях.

Результатом стадії дослідження є документ – специфікація вимог, схвалений усіма членами проекту. Пізніше, в процесі розробки, цей документ буде важливий для запобігання розповзання меж проекту або непотрібних змін. Оскільки система розвивається, кожна нова функція відкриває світ нових можливостей, таким чином специфікація вимог прив'язує команду до оригінального бачення системи і дозволяє контрольоване обговорення змін.

У той час як багато організацій все ще використовують звичайні документи для керування вимогами, інші управляють своїми базовими вимогами, використовуючи програмні інструменти.

Ці інструменти керують вимогами використовуючи базу даних, і зазвичай мають функції автоматизації відстеження (наприклад, дозволяючи створювати зв'язки між батьківськими і дочірніми вимогами, або між тестами і вимогами), управління версіями, і управління змінами. Зазвичай такі інструментальні засоби містять функцію експорту, яка дозволяє створювати звичайний документ, експортуючи дані вимог.

Аналіз здійсненності

На стадії аналізу здійсненності визначається вартість вимог. Для користувальницьких вимог поточна вартість роботи порівнюється з майбутньою вартістю встановленої системи. Задаються питання такі як: «Скільки нам зараз варті помилки введення даних?» Або, «Яка вартість втрати даних через помилки оператора пов'язаної з використовуваним інтерфейсом?». Фактично, потреба в новому інструменті часто розпізнається, коли подібні питання потрапляють до уваги людей, що займаються в організації фінансами.

Ділова вартість включає відповіді на такі питання як: «У якого відділу є бюджет на це?» «Який рівень повернення коштів від нового продукту на ринку?» «Який рівень скорочення внутрішніх витрат на навчання і підтримку, якщо ми зробимо нову, більш просту в використанні систему?»

Технічна вартість пов'язана з вартістю розробки програмного забезпечення та апаратною вартістю. «Чи є у нас потрібні люди, щоб створити інструмент?» «Чи потребуємо ми нове устаткування для підтримки нової системи?»

Подібні питання дуже важливі. Група повинна з'ясувати, чи буде новий автоматизований інструмент мати достатню ефективність аби перенести частину тягаря користувачів на систему і зекономити час людей.

Ці питання також вказують на основну суть управління вимогами. Людина і інструмент формують систему, і це розуміння особливо важливе, якщо інструмент – комп'ютер або новий додаток на комп'ютері. Людський розум вкрай ефективний у паралельній обробці та інтерпретації тенденцій з недостатніми даними. Комп'ютерний процесор ефективний у послідовній обробці і точному математичному обчисленні. Основна мета управління вимогами для програмного проекту полягала б у тому, щоб гарантувати, що автоматизована робота призначена «правильному» процесору.

Наприклад, «не змушуйте людину пам'ятати, де вона знаходиться в системі. Примусьте інтерфейс завжди повідомляти про місцезнаходження людини в системі ». Або «не змушуйте людини вводити ті ж самі дані в два екрани. Примусьте систему зберігати дані і заповнювати їх де необхідно автоматично». Результатом стадії аналізу здійсненності є бюджет і графік проекту.

Дизайн. **Припускаючи, що вартість точно визначена і переваги, які будуть отримані, є досить великими, проект може перейти до стадії проектування.**

На стадії дизайну основна діяльність управління вимогами полягає в тому, щоб перевіряти чи відповідають результати дизайну документу вимог, щоб упевнитися, що робота залишається в межах проекту.

І знову, гнучкість є ключем до успіху. Ось класичний приклад змін проекту, які відмінно працювали. Проектувальники Форда на початку 1980-х очікували, що ціни на бензин піднімуться до 3,18 дол за галон до кінця десятиліття. На середині процесу дизайну автомобіля Ford Taurus, ціни встановилися приблизно на рівні 1,50 дол за галон. Колектив дизайнерів вирішив, що вони могли б створити більший, більш зручний, і більш потужний автомобіль, якщо б ціни на бензин залишилися низькими. Таким чином, вони перепроектувати автомобіль. Коли новий автомобіль вийшов, він встановив загальнонаціональні рекорди продажів.

У більшості випадків, однак, відступ від оригінальних вимог до такої міри не працює. Таким чином документ вимог стає ключовим інструментом, який допомагає команді приймати рішення про зміни дизайну.

Розробка та тестування. На стадії розробки і тестування, основна діяльність управління вимогами – це гарантувати, що робота і ціна залишаються в межах графіка і бюджету, і що створюваний інструмент дійсно відповідає вимогам. Основним інструментом, використовуваним на цій стадії, є створення прототипу і ітераційне тестування. Для програмного додатка користувацький інтерфейс може бути створений на папері і перевірений з потенційними користувачами, в той час як створюється основа програми. Результати цих тестів записуються в керівництві по дизайну користувацького інтерфейсу і передаються колективу дизайнерів. Це економить їх час і робить їх завдання набагато простіше.

При розробці ПЗ було використано підходи ри́зик-ме́неджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

– систему управління;

– систему ідентифікації і вимірювання;

– систему супроводження (моніторингу та контролю).

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації. У 1738 швейцарський математик Даніель Бернуллі доповнив теорію вірогідності методом корисності або привабливості того або іншого результату подій. Ідея Бернуллі полягала в тому, що в процесі ухвалення рішення люди приділяють більше уваги розміру наслідків різних результатів, ніж їх вірогідність. В кінці XIX століття англійський дослідник Ф. Гальтон запропонував вважати регресію або повернення до середнього значення універсальною статистичною закономірністю. Суть регресії трактувалася ним як повернення явищ до норми з часом. Згодом було доведено, що правило регресії діє в найрізноманітніших ситуаціях, починаючи з азартних ігор та розрахунку вірогідності виникнення нещасних випадків, і закінчуючи прогнозуванням коливань економічних циклів. У 1952 аспірант Університету Чикаго Гарі Марковіц в статті «Диверсифікація вкладень» («Portfolio Selection») математично обґрунтував стратегію диверсифікації інвестиційного портфеля, зокрема, він показав, як шляхом продуманого розподілу вкладень мінімізувати відхилення прибутковості від очікуваного показника. У 1990 Г. Марковіцу присуджена Нобелівська премія за розробку теорії і практики оптимізації портфеля фондових активів.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – фотографіями, скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– Вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму SHA-3 (Keccak) – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Keccak став переможцем конкурсу криптографічних алгоритмів, проведеним Національним інститутом стандартів і технологій США. 5 серпня 2015 року алгоритм

затверджено та опубліковано в якості стандарту FIPS 202[1]. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Проте в апаратних реалізаціях Keccak виявився набагато швидшим, ніж всі інші фіналісти.

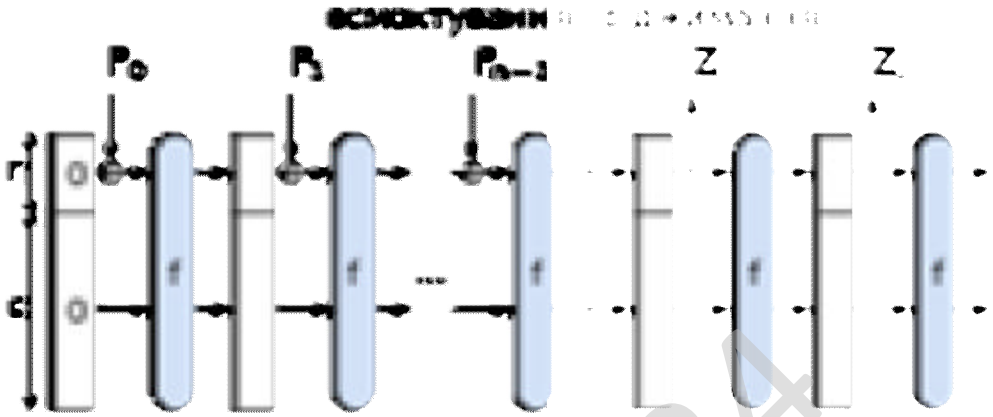

Рисунок 4.3 – Конструкція функції губки, використана в геш-функції

Конструкція функції губки, використана в геш-функції. P_i – вхідні блоки, Z_j – вихід алгоритму. Невикористаний для виведення набір бітів c («capacity») повинен мати значний розмір для досягнення стійкості до атак.

Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak раніше).

Геш-функції сімейства SHA-3 побудовані на основі конструкції криптографічної губки, в якій дані спочатку «вбираються» в губку, при якому початкове повідомлення M піддається багатораундовим перестановкам f, потім результат Z «віджимається» з губки. На етапі «вбирання» блоки повідомлення додаються за модулем 2 з підмножиною стану, який потім перетвориться з допомогою функції перестановки f. На етапі «віджимання» вихідні блоки зчитуються з одного і того ж підмножинного стану, зміненого функцією перестановок f. Розмір частини стану, який записується і зчитується, називається

«швидкістю» (англ. rate) і позначається r, а розмір частки, яка незаймана введенням / виведенням, називається «ємністю» (англ. capacity) і позначається c.

Алгоритм отримання значення хеш-функції можна розділити на кілька етапів:

– Вихідне повідомлення M додається до рядка P довжини, кратній r, за допомогою функції доповнення (pad-функції).

– Рядок P ділиться на n блоків довжини r: P_0, P_1,..., P_{n-1}

– «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжини b біт і підсумовується по модулю 2 з рядком стану S, де S – рядок довжини b біт (b = r + c). Перед використанням цієї функції всі елементи S дорівнюють нулю. Для кожного наступного блоку стан – рядок, отриманий застосуванням функції перестановок f до результату попереднього кроку.

– «Віджимання»: поки довжина Z менша d (d – кількість біт в результаті геш-функції), до Z додається r перших біт стану S, після кожного додавання до S, застосовується функція перестановок f. Потім S обрізається до довжини d біт

– Рядок Z довжини d біт повертається в якості результату

Завдяки тому, що стан містить c додаткових біт, алгоритм стійкий до атаки подовженням повідомлення, до якої прийняті алгоритми SHA-1 і SHA-2.

У SHA-3 стан S – це масив 5 × 5 слів довжиною w = 64 біта, всього 5 × 5 × 64 = 1600 біт. Також в Keccak можуть використовуватися довжини w, рівні меншим ступенями 2 (від w = 1 до w = 32).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ гіперконвергенції ІТ-інфраструктури яке зображено на рисунку 5.1. Розрахунок проходе через консольний додаток з передачею результатів у інтерфейс. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Аналізатор мережі; Параметри; Довідка.

– Вікна обрання групи даних.

– Розділу налаштування ПЗ.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

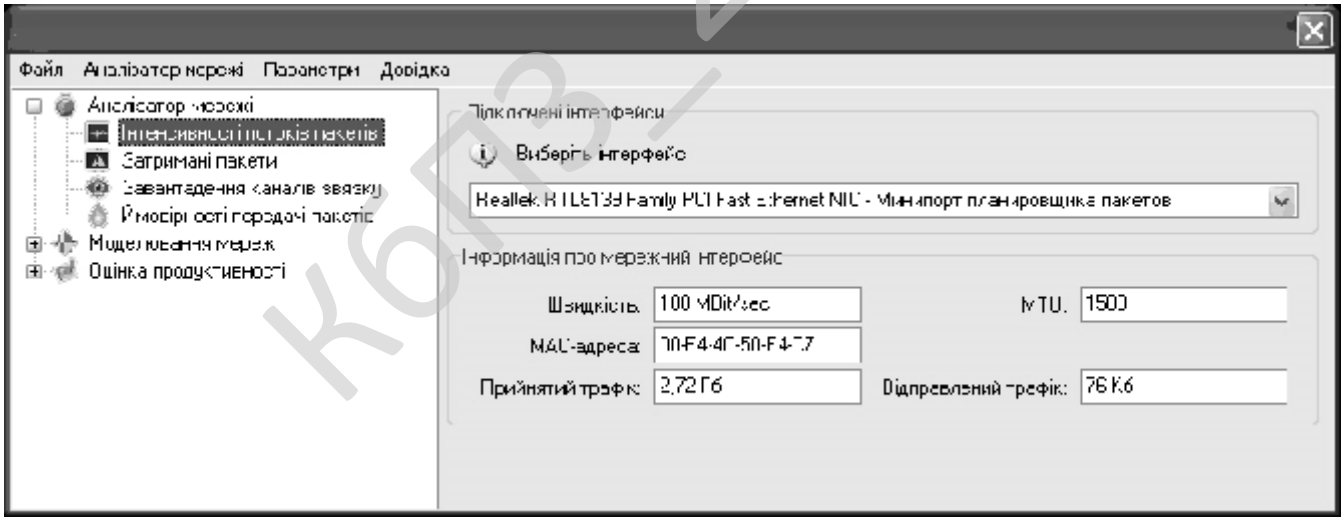

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

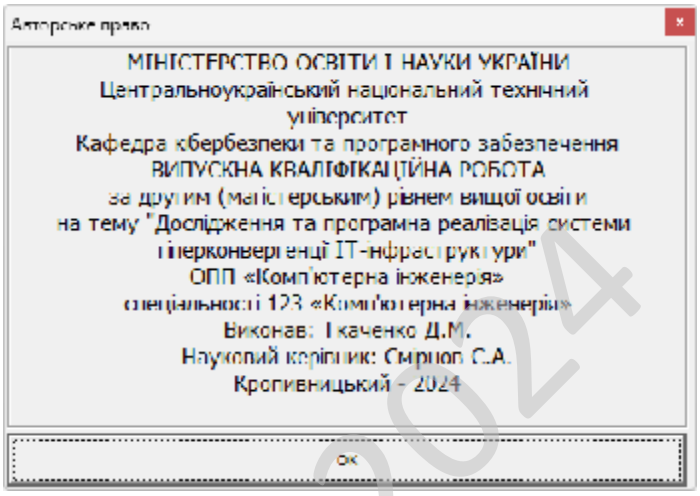

Рисунок 5.2 – Авторське право

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи гіперконвергенції ІТ-інфраструктури.

Метою розробки є дослідження та програмна реалізація системи гіперконвергенції ІТ-інфраструктури.

Об'єктом дослідження є процес гіперконвергенції ІТ-інфраструктури.

Предметом дослідження є методи гіперконвергенції ІТ-інфраструктури.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод гіперконвергенції ІТ-інфраструктури.

– Розроблено вітчизняний продукт гіперконвергенції ІТ-інфраструктури, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи гіперконвергенції ІТ-інфраструктури можуть бути цікавими цільовій групі, яку представлено на рисунку 7.1.

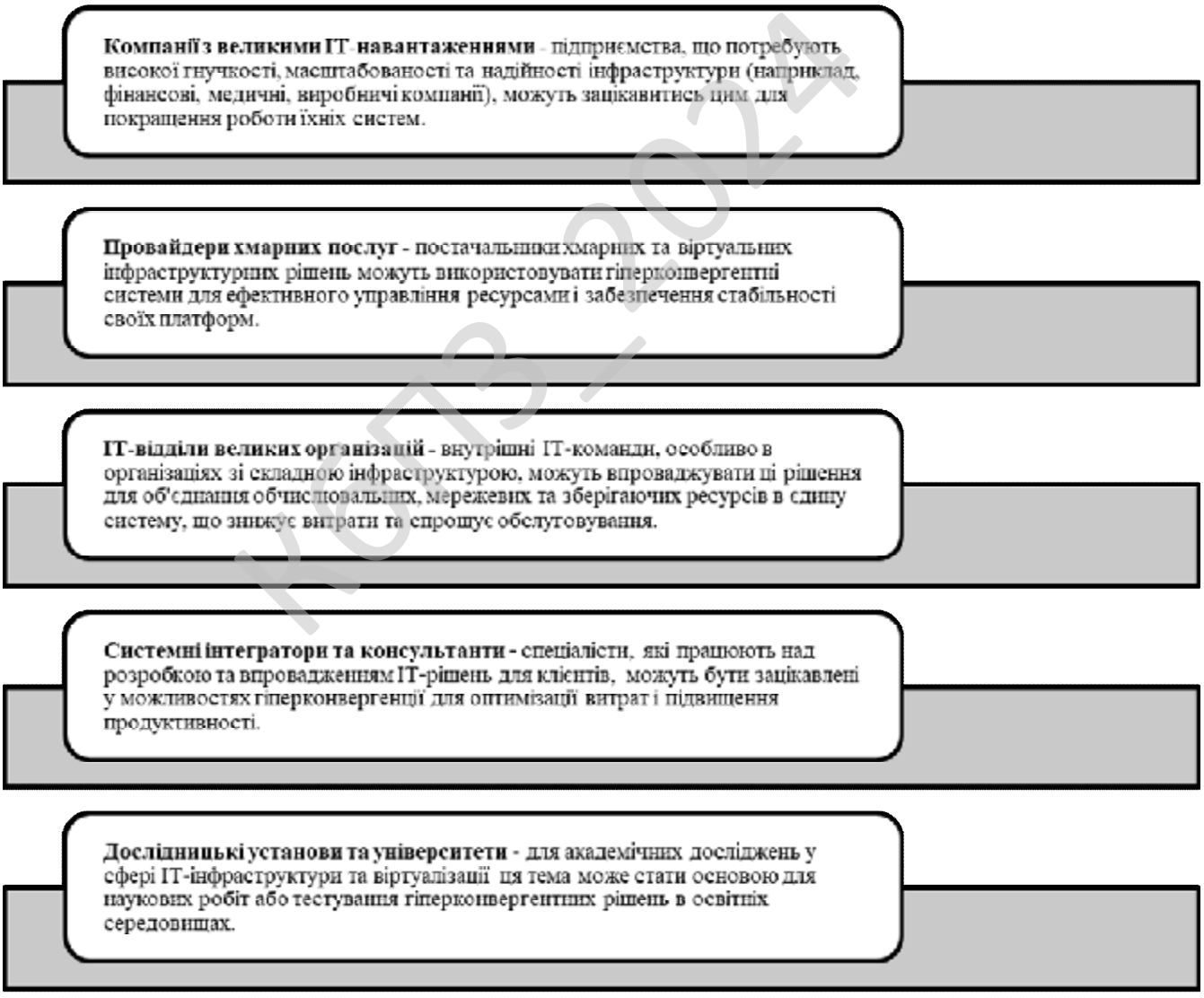

Рисунок 7.1 – Цільова аудиторія

Результати дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури можуть також бути корисними для будь-якої компанії, що працює з великими обсягами даних або має потребу в адаптації гнучких та ефективних IT-ресурсів.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

При оцінці привабливості програмної реалізації системи гіперконвергенції IT-інфраструктури можна застосувати метод експертних оцінок, який передбачає залучення фахівців для аналізу різних аспектів проєкту. Один із поширених підходів – метод бальної оцінки, коли експерти оцінюють проєкт за визначеними критеріями на основі попередньо встановленої шкали.

Залучаються експерти з різних сфер, пов'язаних із гіперконвергентною інфраструктурою: системні архітектори, IT-менеджери, фахівці з безпеки, представники користувачів (бізнес-аналітики, менеджери).

Ними визначаються критерії. Кожен критерій оцінюється на шкалі від 1 до 5, де 1 – мінімальна привабливість, а 5 – максимальна.

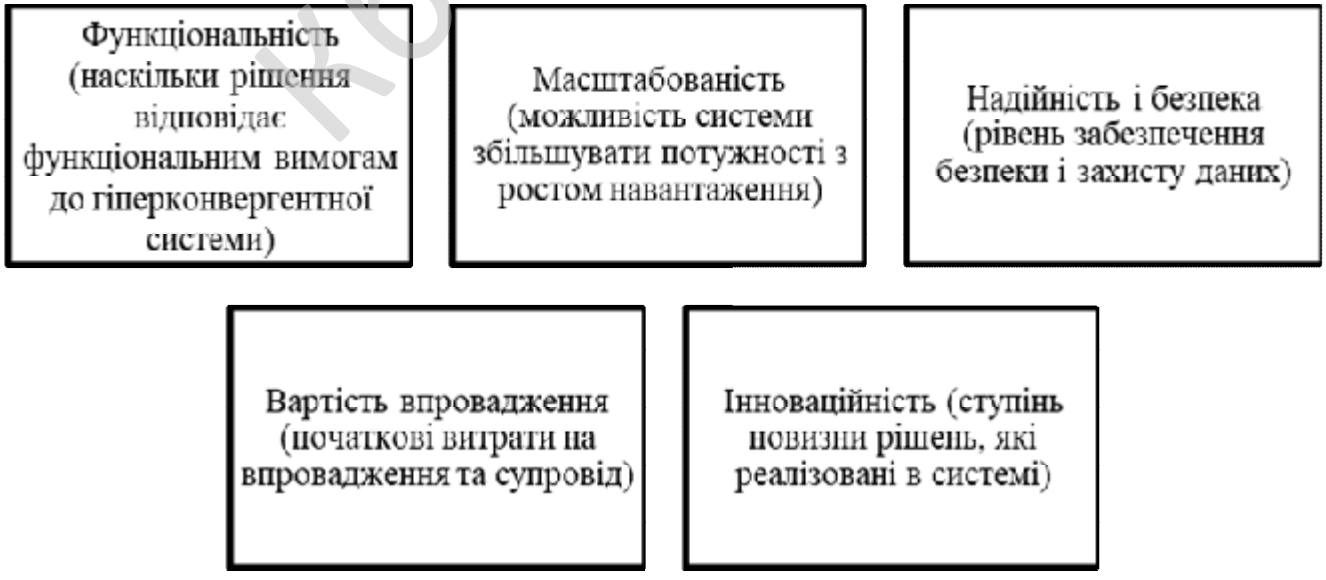

Рисунок 7.2 – Критерії експертної оцінки

Експерти оцінюють проєкт за кожним із критеріїв, після чого визначаються середні оцінки для кожного критерію.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерій	Експерт 1	Експерт 2	Експерт 3	Середній бал
Функціональність	5	4	5	4.67
Масштабованість	4	4	5	4.33
Надійність і безпека	5	5	4	4.67
Вартість впровадження	3	3	4	3.33
Інноваційність	4	5	5	4.67

Обчислюється середній інтегральний бал за всіма критеріями: Інтегральна оцінка=4.67+4.33+4.67+3.33+4.675=4.33

З отриманих результатів можна зробити висновок про привабливість проєкту:

– високі бали за критеріями функціональності, надійності, безпеки та інноваційності свідчать про значну привабливість системи для потенційних впроваджень;

– низький бал за критерієм вартості може вказувати на необхідність оптимізації витрат або пошуку альтернативних підходів фінансування.

Таке оцінювання допомагає ухвалити обґрунтоване рішення щодо доцільності впровадження системи гіперконвергенції для конкретних цілей або компаній.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи гіперконвергенції ІТ-інфраструктури краще використовувати комбінований підхід із декількох методів, оскільки він дозволяє отримати точніший результат. Пропонуємо звертати увагу на метод аналогій та оцінки за компонентами.

Метод експертної оцінки (аналогій) використовується для орієнтовної оцінки на основі вартості подібних проєктів, уже реалізованих у компанії або на ринку. Досвідчені експерти надають оцінку, аналізуючи попередні проєкти аналогічної складності та масштабу. Цей метод дає можливість швидко отримати базове уявлення про вартість, однак потребує залучення фахівців із досвідом у гіперконвергенції.

Метод оцінки за компонентами (деталізований аналіз) включає: програмні компоненти (ліцензії, розробка та інтеграція), обчислювальні ресурси (сервери, сховища), інфраструктура віртуалізації, витрати на налаштування, тестування і супровід.

Цей метод забезпечує деталізований підхід до оцінки вартості, що дозволяє точно розподілити бюджет за основними категоріями витрат. Такий підхід є корисним, якщо необхідно уникнути непередбачуваних витрат.

Такий підхід допоможе уникнути прихованих витрат і забезпечить повне уявлення про бюджет проєкту.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність від впровадження системи гіперконвергенції IT-інфраструктури може проявлятися в декількох ключових аспектах. Розглянемо розрахунок із розрахунками економії та підвищення продуктивності для компанії, яка планує впровадження такої системи. Вхідні дані та розрахунки внесено до таблиці 7.2.

Таким чином, за перший рік впровадження гіперконвергентної системи компанія отримує вигоду в розмірі $182,000, з яких $102,000 складає річна економія на обслуговуванні, енергоспоживанні та підвищеній продуктивності.

Таблиця 7.2 – Основні показники впровадження проєкту

1. **Зниження витрат на обладнання** Гіперконвергенція об'єднує обчислювальні потужності, сховища та мережеву інфраструктуру в єдине рішення. Це зменшує необхідність у традиційному обладнанні (окремі сервери, мережеві сховища, маршрутизатори тощо). **Розрахунок**: Початкові витрати на традиційне обладнання: $200,000 Початкові витрати на гіперконвергентну систему: $120,000 **Економія**: $80,000
2. **Зниження витрат на обслуговування та підтримку** Заміна багатьох окремих систем одним гіперконвергентним рішенням спрощує підтримку й обслуговування, зменшуючи витрати на IT-персонал і час простою. **Розрахунок**: Річні витрати на обслуговування традиційної інфраструктури: $50,000 Річні витрати на обслуговування гіперконвергентної інфраструктури: $30,000 **Річна економія**: $20,000
3. **Зменшення енергоспоживання** Консолідація інфраструктури дозволяє значно скоротити витрати на електроенергію, оскільки знижується кількість обладнання, що потребує енергопостачання та охолодження. **Розрахунок**: Річні витрати на електроенергію для традиційної системи: $15,000 Річні витрати на електроенергію для гіперконвергентної системи: $8,000 **Річна економія**: $7,000

Продовження таблиці 7.2

4. **Підвищення продуктивності та швидкості розгортання** Гіперконвергенція дозволяє швидше розгортати ресурси та масштабувати їх за потребою, що покращує продуктивність роботи та зменшує час простою. Це може призвести до підвищення ефективності роботи бізнесу. **Розрахунок**: Очікуваний приріст продуктивності: 15% Річний дохід компанії до впровадження системи: $500,000 **Додатковий дохід** завдяки підвищенню продуктивності: $500,000 * 0.15 = $75,000
5. **Загальна економічна ефективність за рік** Обчислимо загальну річну економію та вигоду від впровадження: Економія на обладнанні (одноразово): $80,000 Річна економія на обслуговуванні: $20,000 Річна економія на енергоспоживанні: $7,000 Додатковий дохід завдяки підвищенню продуктивності: $75,000 **Загальна економічна вигода за перший рік**: 80,000+20,000+7,000+75,000=182,000

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування проєкту програмної реалізації системи гіперконвергенції ІТ-інфраструктури може бути успішним завдяки чіткому алгоритму, що включає як технічні, так і маркетингові кроки (рисунок 7.3).

Цей алгоритм дозволить системно підходити до просування та збільшити шанси на успішне впровадження та комерційний успіх проєкту.

Рисунок 7.3 – Алгоритму просування проєкту

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи гіперконвергенції IT-інфраструктури можна застосувати низку стратегій. Основними цілями тут є ефективне охоплення цільової аудиторії, скорочення витрат на просування та максимізація доходу від продажу. Нижче наведені пропозиції, що можуть бути корисними для досягнення цих цілей – рисунок 7.4.

Впровадження цих стратегій допоможе оптимізувати канали збуту та шляхи реалізації проєкту програмної реалізації системи гіперконвергенції IT-інфраструктури, що, в свою чергу, може підвищити рівень продажів і задоволення клієнтів.

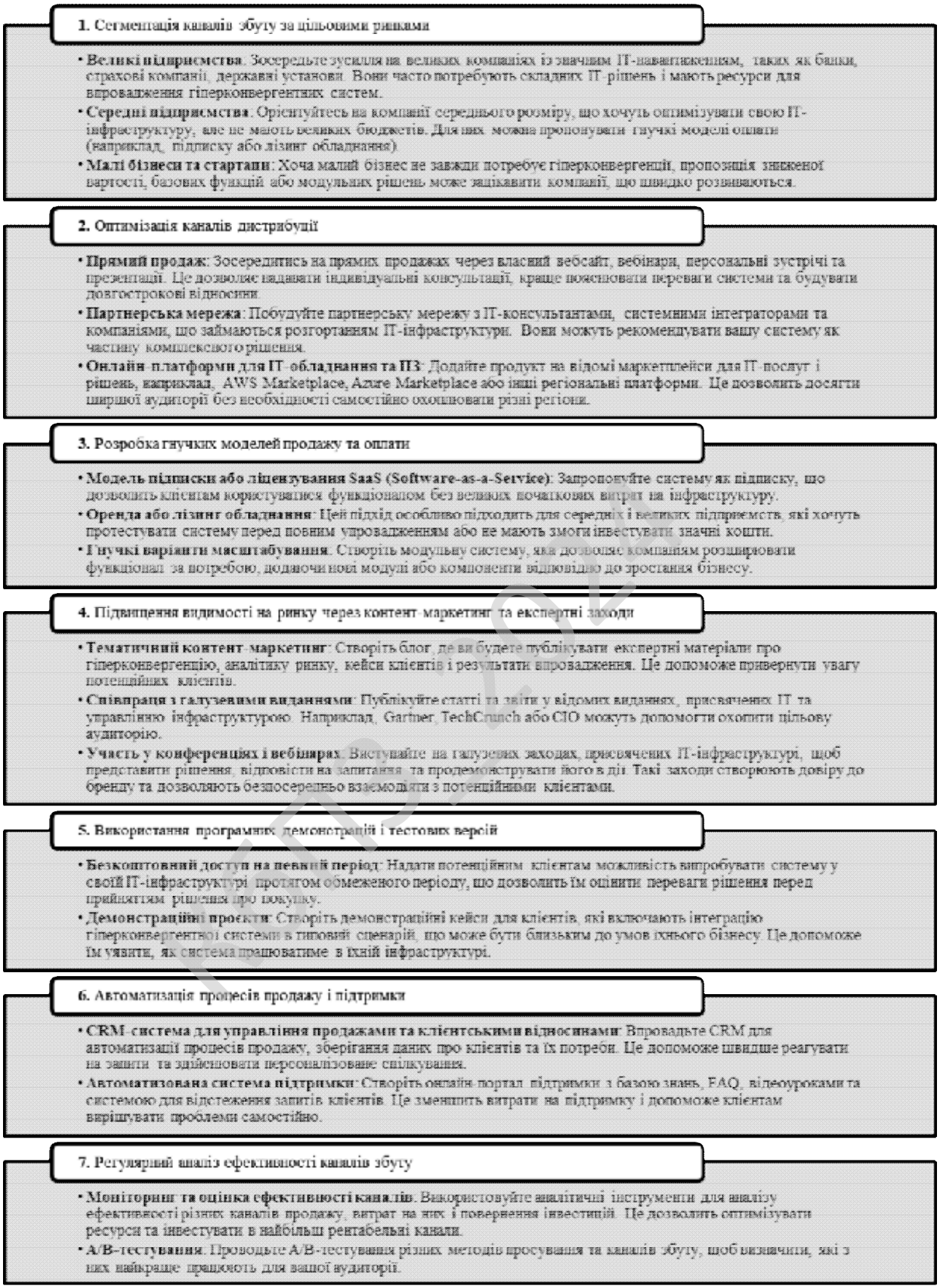

Рисунок 7.4 – Оптимізація каналів збуту та шляхів реалізації

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху проєкту програмної реалізації системи гіперконвергенції IT-інфраструктури є комплекс стратегічних, технічних і організаційних елементів, які забезпечують ефективне впровадження, функціональність і високу цінність рішення для клієнтів.

Основні фактори успіху представлені на рисунку 7.5.

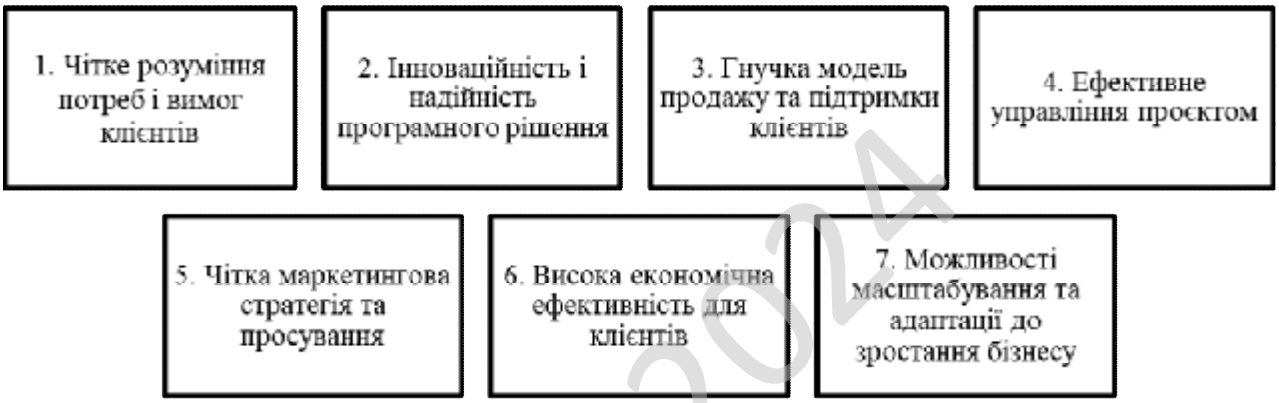

Рисунок 7.5 – Ключові фактори успіху проєкту

Ці ключові фактори забезпечують як технічний, так і комерційний успіх проєкту. Вони сприяють високому рівню задоволення клієнтів, підвищують конкурентоспроможність та сприяють довгостроковій прибутковості рішення.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – система збереження життя і здоров'я працівників у процесі трудової діяльності, що включає правові, соціально-економічні, організаційні, технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи

з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2].

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової діяльності. Їх праця стала більш інтенсивною, напруженою і вимагає значних витрат розумової, емоційної і фізичної енергії. Це призвело до необхідності у знаходженні комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку.

Охорона здоров'я робітників, забезпечення безпеки умов праці, ліквідація та профілактика професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

8.2 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах. Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами. Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань. Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців ІТ-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців ІТ-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців ІТ-індустрії.

Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Пропозиції щодо підвищення працездатності ІТ-фахівців, розіб'ємо на декілька категорій:

Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням ІТ-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють ІТ-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження ІТ-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці ІТ-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність ІТ-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві ІТ-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який

повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9]. Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В. Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення будемо застосовувати вертикальні електроди з сталевого прокату круглого перерізу діаметром 35 мм., довжиною L=2 м., та горизонтальний електрод – метелева полоса з перетином 35·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником) (рис. 8.1).

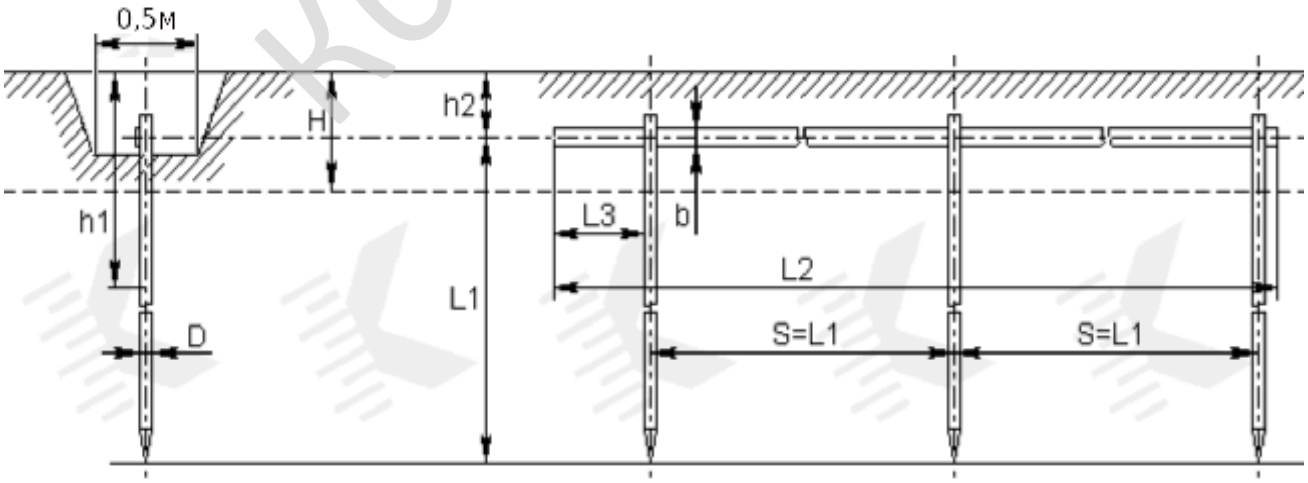

Рисунок 8.1 – Схема штучного заземлення

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення:

– тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір ρ_2 = 40 Ом·м);

– умовна товщина верхнього шару грунта: Н=0,6 м;

– відстань між вертикальними заземлювачами (електродами) А=3 м;

– глибина закладення горизонтального контура заземлення t=0,75 м;

– опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом.

Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Виконаємо розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,75+2/2=1,75 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому грунті [10];

ρ_2= 40 Ом·м – табличне значення питомого опору нижнього шару грунта (глина) [10].

Діаметр вертикального електрода (заданий) Dв= 35 мм = 0,035 м.

Відношення A/L=3/2=1,5.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [10]:

$$R_O=0,366\cdot(\rho/L)\cdot[\lg(2L/Dв)+(1/2)\cdot\lg((4T+L)/(4T-L))]=$$

$$=0,366\cdot(54,5/2)\cdot[(\lg(2\cdot2/0,035)+(1/2)\cdot\lg((4\cdot1,75+2)/(4\cdot1,75-2))]=21,7 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів Кев=0,53 при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [10].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N=R_О /(Кев\ R_{ЗН})= 21,7 / (0,53\cdot4)= 10,2 \approx 10 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot3\cdot10=32,3\approx 32 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [10]:

$$R_П =0,366\cdot(\rho\cdot K_П/L_П)\cdot\lg(2(L_П \cdot L_П)/(B\cdot t))=$$

$$=0,366\cdot(40\cdot5/40)\cdot\lg((2\cdot40^2)/(0,035\cdot0,75))=11,14 \text{ Ом.}$$

де $K_П$=5 – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичної зони для з'єднуючої полоси [10]:

B = 35 мм. = 0,035 м.- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [10]:

$$R =(R_О\cdot R_П)/(R_О\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(21,7\cdot 11,14)/(21,7\cdot 0,55+ 10\cdot 11,14\cdot 0,53)=3,4 \text{ Ом.}$$

де $\eta_П$ = 0,55 – табличне значення коефіцієнта екранування з'єднуючої полоси [10].

Умова $R \le R_{ЗН}$ виконується ($3,4 \le 4$).

Оскільки при 10 вертикальних електродах R суттево быльше $R_{ЗН}$, зменшимо кількість вертикальних електродів N до 9 і виконаємо перерахунок. У результаті остаточно отримали: R = 3,9 Ом. при кількості вертикальних електродів N = 9.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд питань пожежної безпеки, небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів о питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп’ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи гіперконвергенції IT-інфраструктури.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів гіперконвергенції IT-інфраструктури.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем гіперконвергенції IT-інфраструктури.

– Досліджена система гіперконвергенції IT-інфраструктури.

– На основі отриманих результатів досліджень створена програмна реалізація системи гіперконвергенції IT-інфраструктури.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання гіперконвергенції IT-інфраструктури.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHA-3.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ткаченко Д.М. Дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

4. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

5. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

6. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

7. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

8. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

9. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

10. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

11. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of

Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

12. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

13. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

14. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

15. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

16. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

17. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

18. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

19. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

20. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

21. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

22. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

23. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

24. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

25. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

26. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

27. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

28. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

29. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

30. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

31. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

32. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду

абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

33. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

35. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

36. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

38. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

39. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

41. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

42. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

47. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

50. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

51. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

52. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

53. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи гіперконвергенції IT-інфраструктури.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи гіперконвергенції IT-інфраструктури;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті пропозиції щодо підвищення працездатності ІТ-фахівців.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 94 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 17.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Смірнов С.А.

Дослідження та програмна реалізація системи гіперконвергенції IT-інфраструктури

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2024 року

Файл main.py

```
from node import Node
from logger import setup_logger, log_event, log_error
from monitoring import monitor_cpu, monitor_memory, monitor_network
from node_config import save_node_config, load_node_config
from orchestration import balance_load, scale_up
from backup import create_backup
from security import hash_data, verify_hash

class HyperconvergedInfrastructure:
    def __init__(self):
        self.nodes = []

    def add_node(self, name: str, ip_address: str, cpu: int, memory: int):
        #Додавання нового вузла в інфраструктуру
        node = Node(name, ip_address, cpu, memory)
        self.nodes.append(node)
        log_event(f"Вузол {name} додано в інфраструктуру.")

    def remove_node(self, name: str):
        #Видалення вузла з інфраструктури
        self.nodes = [node for node in self.nodes if node.name != name]
        log_event(f"Вузол {name} видалено з інфраструктури.")

    def find_node(self, name: str) -> Node:
        #Пошук вузла за ім'ям
        for node in self.nodes:
            if node.name == name:
                return node
        return None

    def start_service_on_node(self, node_name: str, service_name: str):
        #Запуск сервісу на певному вузлі
        node = self.find_node(node_name)
        if node:
            node.start_service(service_name)
            log_event(f"Сервіс {service_name} запущено на вузлі {node_name}.")

    def stop_service_on_node(self, node_name: str, service_name: str):
        #Зупинка сервісу на певному вузлі
        node = self.find_node(node_name)
        if node:
            node.stop_service(service_name)
            log_event(f"Сервіс {service_name} зупинено на вузлі {node_name}.")

    def monitor_all_nodes(self):
        #Моніторинг всіх вузлів
        for node in self.nodes:
            node.monitor()
```

```
    def get_infrastructure_status(self):
        #Отримання статусу всієї інфраструктури
        for node in self.nodes:
            print(node.to_dict())

#Основна програма
setup_logger()
hci = HyperconvergedInfrastructure()

#Додавання вузлів
hci.add_node('Node1', '192.168.1.2', 8, 16)
hci.add_node('Node2', '192.168.1.3', 16, 32)

#Запуск сервісу
hci.start_service_on_node('Node1', 'WebServer')

#Моніторинг вузлів
hci.monitor_all_nodes()

#Збереження налаштувань вузлів
save_node_config(hci.nodes)

#Масштабування та балансування
balance_load(hci.nodes)
scale_up(hci.find_node('Node1'))
#Резервне копіювання
create_backup('/data', '/backup/data_backup')
```

Файл security.py

```
import hashlib

def hash_data(data: str) -> str:
    #Генерація хешу для даних
    return hashlib.sha256(data.encode()).hexdigest()

def verify_hash(data: str, expected_hash: str) -> bool:
    #Перевірка хешу даних
    return hash_data(data) == expected_hash
```

Файл backup.py

```
import shutil
import os

def create_backup(source_dir: str, backup_dir: str):
    #Створення резервної копії папки
    if os.path.exists(source_dir):
        shutil.copytree(source_dir, backup_dir)
    else:
        print(f"Джерело {source_dir} не знайдено.")
```

Файл orchestration.py

```
def balance_load(nodes):
    #Балансування навантаження між вузлами
    total_cpu = sum(node.cpu_usage for node in nodes)
    avg_cpu = total_cpu / len(nodes)

    for node in nodes:
        if node.cpu_usage > avg_cpu:
            print(f"Перенесення навантаження з {node.name}")
            #Логіка переміщення навантаження

def scale_up(node):
    #Масштабування ресурсу на вузлі
    print(f"Масштабування ресурсів для {node.name}")
    #Логіка додавання ресурсів
```

Файл node_config.py

```
import json
from node import Node

def save_node_config(nodes, filename='nodes_config.json'):
    #Збереження налаштувань вузлів у JSON файл
    node_data = [node.to_dict() for node in nodes]
    with open(filename, 'w') as f:
        json.dump(node_data, f, indent=4)

def load_node_config(filename='nodes_config.json'):
    #Завантаження налаштувань вузлів з JSON файлу
    with open(filename, 'r') as f:
        node_data = json.load(f)
    return node_data
```

Файл monitoring.py

```
import psutil

def monitor_cpu():
    #Моніторинг використання CPU
    return psutil.cpu_percent(interval=1)

def monitor_memory():
    #Моніторинг використання пам'яті
    return psutil.virtual_memory().percent

def monitor_network():
    #Моніторинг мережевого трафіку
    net_io = psutil.net_io_counters()
    return {'sent': net_io.bytes_sent, 'received': net_io.bytes_recv}
```

Файл logger.py

```
import logging

def setup_logger():
    #Налаштування логування
    logging.basicConfig(
        filename='hci_system.log',
        level=logging.INFO,
        format='%(asctime)s - %(levelname)s - %(message)s'
    )

def log_event(message: str):
    #Запис повідомлення в лог
    logging.info(message)

def log_error(message: str):
    #Запис помилки в лог
    logging.error(message)
```

Файл node.py

```
import psutil
from typing import Dict

class Node:
    def __init__(self, name: str, ip_address: str, cpu: int, memory: int):
        self.name = name
        self.ip_address = ip_address
        self.cpu = cpu
        self.memory = memory
        self.disk = self.get_disk_usage()
        self.services = []
        self.status = 'Inactive'

    def get_disk_usage(self) -> Dict:
        #Отримання інформації про дисковий простір
        disk_usage = psutil.disk_usage('/')
        return {
            'total': disk_usage.total,
            'used': disk_usage.used,
            'free': disk_usage.free,
            'percent': disk_usage.percent
        }

    def start_service(self, service_name: str):
        #Запуск сервісу на вузлі
        self.services.append(service_name)
        self.status = 'Active'

    def stop_service(self, service_name: str):
        #Зупинка сервісу на вузлі
        if service_name in self.services:
            self.services.remove(service_name)
        if not self.services:
            self.status = 'Inactive'

    def monitor(self):
        #Моніторинг стану вузла
        self.cpu_usage = psutil.cpu_percent(interval=1)
        self.memory_usage = psutil.virtual_memory().percent
        self.disk = self.get_disk_usage()

    def to_dict(self) -> Dict:
        #Конвертація об'єкта у словник
        return {
            'name': self.name,
            'ip_address': self.ip_address,
            'cpu': self.cpu,
            'memory': self.memory,
            'disk': self.disk,
            'services': self.services,
```

```
        'status': self.status
    }
```

Файл diagnostics.py

```
import subprocess

def ping_node(ip_address: str) -> bool:
    #Перевірка доступності вузла через ping
    try:
        output = subprocess.check_output(['ping', '-c', '1', ip_address])
        return True
    except subprocess.CalledProcessError:
        return False

def check_service_running(service_name: str) -> bool:
    #Перевірка, чи працює певний сервіс
    try:
        output = subprocess.check_output(['systemctl', 'is-active',
service_name])
        if 'active' in output.decode('utf-8'):
            return True
        else:
            return False
    except subprocess.CalledProcessError:
        return False

# Імпорт та використання:
from diagnostics import ping_node, check_service_running

if ping_node('192.168.1.2'):
    print("Вузол доступний.")
else:
    print("Вузол не відповідає.")

if check_service_running('WebServer'):
    print("Сервіс WebServer працює.")
else:
    print("Сервіс WebServer не працює.")
```

Файл network_management.py

```
import psutil

def get_network_stats() -> dict:
    #Отримання статистики по мережі
    net_io = psutil.net_io_counters()
    return {
        'bytes_sent': net_io.bytes_sent,
        'bytes_recv': net_io.bytes_recv,
        'packets_sent': net_io.packets_sent,
        'packets_recv': net_io.packets_recv
    }

def configure_firewall_rule(rule: str):
    #Налаштування правила фаєрвола
    print(f"Застосування правила фаєрвола: {rule}")
    #Реальна логіка застосування правила
```

Файл notifications.py

```
import smtplib
from email.mime.text import MIMEText

def send_email_notification(subject: str, body: str, to_address: str):
    #Надсилання сповіщення через електронну пошту
    msg = MIMEText(body)
    msg['Subject'] = subject
    msg['From'] = 'system@hci.com'
    msg['To'] = to_address

    #Логіка відправки
    with smtplib.SMTP('smtp.example.com') as server:
        server.login('user', 'password')
        server.sendmail('system@hci.com', [to_address], msg.as_string())

def send_sms_notification(phone_number: str, message: str):
#Надсилання SMS сповіщення (мок-реалізація)
    print(f"Відправлено SMS на {phone_number}: {message}")
#Логіка інтеграції з API для відправки SMS

# Імпорт та використання:
from notifications import send_email_notification, send_sms_notification

send_email_notification(
    subject="Помилка системи",
    body="Вузол Node1 не відповідає.",
    to_address="admin@example.com"
)

send_sms_notification(
    phone_number="+1234567890",
    message="Вузол Node1 не відповідає."
)
```

Файл reporting.py

```
def generate_report(infrastructure_status: list, filename: str = 'report.txt'):
    #Генерація текстового звіту
    with open(filename, 'w') as file:
        file.write("Звіт про стан інфраструктури:\n")
        for node in infrastructure_status:
            file.write(f"Вузол: {node['name']}\n")
            file.write(f"IP-адреса: {node['ip_address']}\n")
            file.write(f"CPU: {node['cpu']}%\n")
            file.write(f"Пам'ять: {node['memory']}%\n")
            file.write(f"Сервіси: {', '.join(node['services'])}\n")
            file.write(f"Статус: {node['status']}\n")
            file.write("-" * 20 + "\n")

def analyze_performance(nodes):
    #Аналіз продуктивності вузлів
    high_cpu_nodes = [node for node in nodes if node.cpu_usage > 80]
    if high_cpu_nodes:
        print("Вузли з високим завантаженням CPU:")
        for node in high_cpu_nodes:
            print(f"{node.name}: {node.cpu_usage}% CPU")
    else:
        print("Всі вузли працюють в нормальному режимі.")
# Імпорт та використання:
from reporting import generate_report, analyze_performance

#Отримати статус системи
infrastructure_status = hci.get_infrastructure_status()

#Генерація звіту
generate_report(infrastructure_status)

#Аналіз продуктивності вузлів
analyze_performance(hci.nodes)
```

Файл database.py

```
import sqlite3

def init_db():
    #Ініціалізація бази даних
    conn = sqlite3.connect('hci_infrastructure.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS nodes (
            name TEXT PRIMARY KEY,
            ip_address TEXT,
            cpu INTEGER,
            memory INTEGER,
            disk_usage TEXT,
            services TEXT,
            status TEXT
        )
    ''')
    conn.commit()
    conn.close()

def save_node_to_db(node):
    #Збереження інформації про вузол у базу даних
    conn = sqlite3.connect('hci_infrastructure.db')
    cursor = conn.cursor()
    cursor.execute('''
        INSERT OR REPLACE INTO nodes (name, ip_address, cpu, memory, disk_usage,
services, status)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    ''', (node.name, node.ip_address, node.cpu, node.memory, str(node.disk),
','.join(node.services), node.status))
    conn.commit()
    conn.close()

def get_all_nodes():
    #Отримання списку всіх вузлів з бази даних
    conn = sqlite3.connect('hci_infrastructure.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM nodes')
    nodes = cursor.fetchall()
    conn.close()
    return nodes
#Імпорт та використання:
from database import init_db, save_node_to_db, get_all_nodes

#Ініціалізація бази даних
init_db()

#Збереження вузлів у базу даних
for node in hci.nodes:
    save_node_to_db(node)
```

```
#Отримання всіх вузлів з бази
nodes_from_db = get_all_nodes()
print(nodes_from_db)
```

Файл failover.py

```
def failover(nodes, failed_node_name: str):
    #Автоматичний перехід на резервний вузол у випадку відмови
    for node in nodes:
        if node.name == failed_node_name:
            print(f"Виявлено збій на вузлі {node.name}")
            #Логіка відключення вузла
            node.status = 'Failed'
            print(f"Виконання failover для вузла {node.name}")

            #Пошук активного вузла для переміщення сервісів
            for backup_node in nodes:
                if backup_node.status == 'Active' and backup_node.name !=
node.name:
                    print(f"Переміщення сервісів на вузол {backup_node.name}")
                    for service in node.services:
                        backup_node.start_service(service)
                    break
            break
# Імпорт та використання:
from failover import failover

#Виконання failover для вузла Node1
failover(hci.nodes, 'Node1')
```

Файл advanced_monitoring.py

```
import psutil
import matplotlib.pyplot as plt

def monitor_disk_usage():
    #Моніторинг використання диску
    disk_usage = psutil.disk_usage('/')
    return {
        'total': disk_usage.total,
        'used': disk_usage.used,
        'free': disk_usage.free,
        'percent': disk_usage.percent
    }

def generate_cpu_usage_chart():
    #Генерація графіку використання CPU
    cpu_usages = [psutil.cpu_percent(interval=1) for _ in range(10)]
    plt.plot(cpu_usages)
    plt.title('Використання CPU')
    plt.xlabel('Час (сек)')
    plt.ylabel('CPU (%)')
    plt.savefig('cpu_usage_chart.png')
    print("Графік використання CPU збережено у файл cpu_usage_chart.png")
```

Файл a system_update.py

```
import subprocess

def update_system(node_name: str):
    #Оновлення програмного забезпечення на вузлі
    print(f"Оновлення програмного забезпечення на вузлі {node_name}")
    try:
        result = subprocess.run(['sudo', 'apt-get', 'update'], check=True,
capture_output=True)
        print(f"Оновлення успішно завершено на {node_name}")
    except subprocess.CalledProcessError as e:
        print(f"Помилка під час оновлення на {node_name}: {e}")

def upgrade_system(node_name: str):
    #Оновлення ядра і критичних компонентів
    print(f"Оновлення ядра на вузлі {node_name}")
    try:
        result = subprocess.run(['sudo', 'apt-get', 'upgrade', '-y'],
check=True, capture_output=True)
        print(f"Оновлення ядра успішно завершено на {node_name}")
    except subprocess.CalledProcessError as e:
print(f"Помилка під час оновлення ядра на {node_name}: {e}")
```