МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ,
ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ ТА РАДІОЕЛЕКТРОННИХ СИСТЕМ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Роман ОДАРЧЕНКО
“_____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Тема: «Енергоефективна обробка навантажень в інформаційній мережі

Виконавець: _____ Дмитро МАЙБОРОДА
(підпис)

Керівник: _____ Марина МАЛОЄД
(підпис)

Нормоконтролер: _____ Денис БАХТІЯРОВ
(підпис)

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра телекомунікаційних та радіоелектронних систем
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Роман ОДАРЧЕНКО
“_____”_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Майбороди Дмитра Вадимовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи: «Енергоефективна обробка навантажень в інформаційній мережі»

затверджена наказом ректора від «29» березня 2023 р. № 421/ст

2. Термін виконання роботи: з 22.05.2023 р. по 25.06.2023 р.

3. Вихідні дані до роботи: інформаційна мережа підприємства.

4. Зміст пояснювальної записки: аналіз проблеми забезпечення енергоефективної обробки навантажень в інформаційній мережі; існуючі підходи підвищення енергоефективності обробки навантаження в інформаційній мережі; дослідження та розробка рекомендацій щодо існуючих рішень щодо забезпечення енергоефективності обробки навантажень.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Класифікація підходів щодо підвищення енергоефективності обслуговування навантаження на рівні апаратного забезпечення, розгортання основних сервісів OpenStack, Блок-схема алгоритму Round Robin

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів кваліфікаційної роботи	22.05.2023-24.05.2023	Виконано
2	Вступ	25.05.2023	Виконано
3	Аналіз проблеми забезпечення енергоефективної обробки навантажень в інформаційній мережі	26.05.2023-29.05.2023	Виконано
4	Існуючі підходи підвищення енергоефективності обробки навантаження в інформаційній мережі	30.05.2023-07.06.2023	Виконано
5	Дослідження та розробка рекомендацій щодо існуючих рішень щодо забезпечення енергоефективності обробки навантажень	08.06.2023-14.06.2023	Виконано
6	Усунення недоліків та захист кваліфікаційної роботи	15.06.2023-25.06.2023	Виконано

7. Дата видачі завдання: “19~ травня 2023 р.

Керівник кваліфікаційної роботи _____ Марина МАЛОЄД
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Дмитро МАЙБОРОДА
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Кваліфікаційна робота «Енергоефективна обробка навантажень в інформаційній мережі» містить 58 сторінок, 5 рисунків, 21 використаних джерел.

Об’єкт дослідження – навантаження в інформаційній мережі та його енергоефективна обробка.

Предмет дослідження – методи та засоби енергоефективної обробки навантажень в інформаційній мережі .

Мета кваліфікаційної роботи – розробити рекомендації щодо застосування енергоефективної обробки навантажень в інформаційній мережі.

Метод дослідження – У даному дослідженні було проведено огляд літератури та аналіз експлуатаційної документації і міжнародних стандартів, пов'язаних з функціонуванням сучасних інформаційних мереж. Також було проведено порівняння цих стандартів, що дозволило визначити їхні переваги та недоліки.

У результаті аналізу було встановлено, що сучасні інформаційні мережі активно використовують технології та концепції, які базуються на програмному забезпеченні для вирішення мережевих задач. Важливою проблемою є планування навантаження в ранній стадії розробки веб-проектів. Недостатня продуктивність сервера може призвести до "падіння" в неприйнятний момент, що може мати негативні наслідки.

Одним з методів вирішення проблеми високого навантаження є використання кластеризації, де кілька серверів об'єднуються в кластер і навантаження розподіляється між ними за допомогою балансування навантаження. Кластеризація також забезпечує резервування серверів. Балансування навантаження може здійснюватися як за допомогою апаратних, так і за допомогою програмних інструментів.

Також в роботі було розглянуто методи та засоби підвищення енергоефективності обробки інформації в мережі. Були проаналізовані різні види рішень та їх класифікації, а також досліджено алгоритми підвищення

енергоефективності інформаційних мереж. На основі проведених досліджень було розроблено рекомендації щодо застосування методів та засобів для підвищення енергоефективності навантажень у інформаційних мережах.

Отже, дане дослідження надає важливі відомості про функціонування сучасних інформаційних мереж, аналізує проблеми планування навантаження та ефективної обробки інформації, і розробляє рекомендації щодо використання методів та засобів для підвищення енергоефективності в мережах.

ЗМІСТ

ВСТУП

Актуальність теми. Наразі дана тема дуже актуальна бо з початку глобальної пандемії і ведення карантинних заходів кількість користувачів інформаційної мережі збільшилось на 20%.

За останні два роки, з 2019 по 2021 рік, кількість користувачів Інтернету зросла з 4,1 мільярда до 4,9 мільярда осіб. Один з основних чинників, що призвів до цього зростання, була пандемія COVID-19. Протягом року пандемії було зафіксовано зростання кількості людей, які почали використовувати Інтернет, на 10%. Це є найбільшим річним приростом за останнє десятиліття, за даними Міжнародного союзу електрозв'язку при ООН (МСЕ).

І це тільки люди які раніше не користувалися інтернетом взагалі. Що могло б не вплинути на навантаження мережі, але при цьому багато сфер діяльності перейшли на дистанційну форму праці. Тим самим збільшивши одночасне активне використання занадто великим, що призводило до перенавантаження інтернет мережі.

Мета і завдання дослідження.

Вивчити причини навантажень в інформаційній мережі та дослідити методи обробки цих навантажень. А також розробити засоби енергоефективної обробки навантажень в інформаційній мережі

Об'єктом дослідження – навантаження на інформаційну мережу та на скільки енергоефективно воно обробляється

Предметом дослідження – методи та засоби енергоефективної оброки навантажень в інформаційній мережі

Методи досліджень.

Опрацювання літератури за даною темою, аналіз експлуатаційної документації, міжнародних стандартів та їх порівняння

Апробація отриманих результатів. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- Науково-практична конференція «Проблеми експлуатації та захисту інформаційно-комунікаційних систем», м. Київ, 2023 р.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ЗАБЕЗПЕЧЕННЯ ЕНЕРГОЕФЕКТИВНОЇ ОБРОБКИ НАВАНТАЖЕНЬ В ІНФОРМАЦІЙНІЙ МЕРЕЖІ.

1.1. Процес обробки навантажень та його особливості в інформаційної мережі.

У цифрових інформаційно-комунікаційних мережах обробка пакетів відноситься до широкого спектру алгоритмів, які застосовуються до пакета даних або інформації під час його переміщення різними мережевими елементами комунікаційної мережі. Із збільшенням продуктивності мережевих інтерфейсів виникає відповідна потреба у швидшій обробці пакетів [1].

Існує два широких класи алгоритмів обробки пакетів, які узгоджуються зі стандартизованим підрозділом мережі на площину керування та площину даних. Алгоритми застосовуються до:

- керуюча інформація, що міститься в пакеті, використовується для безпечної та ефективної передачі пакета від джерела до місця призначення;
- вміст даних (часто званий корисним навантаженням) пакета, який використовується для забезпечення деякого специфічного для вмісту перетворення або виконання дії, керованої вмістом.

У будь-якому мережевому пристрої (наприклад, маршрутизаторі, комутаторі, мережевому елементі або терміналі, такому як комп’ютер або смартфон) саме підсистема обробки пакетів керує проходженням багаторівневої мережі або стека протоколів з нижчого фізичного та мережевого рівнів аж до прикладного рівня.

Щоб мережі були успішними, необхідно мати єдиний стандарт, який визначає архітектуру мережевих систем. Основна вимога для такого стандарту полягає в тому, щоб забезпечити структуру, яка дозволить виробникам обладнання та програмного забезпечення в усьому світі розробляти мережеві технології, які працюватимуть

разом, і використовувати їхні сукупні інвестиційні можливості для просування стану мереж уперед.

У 1970-х роках дві організації, Міжнародна організація зі стандартизації [2] (ISO) і Міжнародний союз електрозв'язку (ITU-T), кожна з них працювала над проектом зі створення міжнародних мережевих стандартів. У 1983 році ці зусилля були об'єднані, і в 1984 році ISO опублікував стандарт під назвою « Базова еталонна модель для взаємозв'язку відкритих систем» [4], а ITU-T — як стандарт X.200 [5].

Модель OSI — це 7-рівнева модель [6], яка описує, як працює мережева операційна система. Багаторівнева модель має багато переваг [7], включаючи можливість змінювати один рівень, не впливаючи на інші, і як модель для розуміння того, як працює мережева ОС. Поки зберігається взаємозв'язок між рівнями, постачальники можуть покращити реалізацію окремого рівня без впливу на інші рівні. Проте варто зауважити що вона є абстрактною і не використовується на практиці.

Обладнання на основі IP можна розділити на три основні елементи: площина даних, площина управління та площина управління [21].

− площина даних — це підсистема вузла мережі, яка отримує та надсилає пакети з інтерфейсу, обробляє їх відповідно до вимог відповідного протоколу та доставляє, віддає або пересилає їх відповідно;

− площа керування зберігає інформацію, яка може бути використана для зміни даних, що використана площиною даних. Підтримка цієї інформації вимагає роботи зі складними протоколами сигналізації. Реалізація цих протоколів у площині даних призведе до низької продуктивності пересилання. Загальний спосіб керування цими протоколами неможливий тому, щоб дозволити площині даних виявляти вхідні пакети сигналізації та локально пересилати їх на площину керування. Протоколи сигналізації площини управління можуть оновлювати інформацію про площину даних і вводити вихідні пакети сигналізації в площину даних. Ця архітектура працює, після чого сигнальний трафік стає дуже невеликим у частині глобального трафіку;

− площа керування забезпечує адміністративний інтерфейс у загальній системі. Він містить процеси, які підтримують оперативне адміністрування,

управління або дії конфігурації/ініціалізації, такі як: засоби для підтримки збору та агрегування статистики, підтримка впровадження протоколів управління, інтерфейс командного рядка, графічний інтерфейс конфігурації користувача через веб-сторінки або традиційне керування SNMP (Простий протокол керування мережею). Також, можуть бути включені більш складні рішення на основі XML.

Використаний метод обробки даних визначатиме час відповіді на запит і надійність результату. Тому потрібно ретельно вибирати техніку обробки даних. Наприклад, у ситуації, де доступність має вирішальне значення, наприклад на порталі фондової біржі, обробка транзакцій має бути кращим методом.

Важливо відзначити різницю між обробкою даних і системою обробки даних. Обробка даних стосується правил, за якими дані перетворюються на корисну інформацію. Система обробки даних − це програма, оптимізована для певного типу обробки даних. Наприклад, система розподілу часу розроблена для оптимального виконання процесів розподілу часу. Також можна використовувати його для пакетної обробки. Однак це не дуже добре масштабується для роботи.

У цьому сенсі, вибираючи правильний тип обробки даних для ваших потреб, слід вибрати правильну систему.

Пакетна обробка − це кількість фрагментів даних, що зберігаються протягом певного періоду часу, аналізуються разом або пакетами. Пакетна обробка необхідна, коли власникам бізнесу та дослідникам даних потрібен великий обсяг даних для аналізу для отримання детальної інформації. Наприклад, показники продажів постійно піддаються пакетній обробці, що дозволяє компаніям використовувати такі функції візуалізації даних, як діаграми, графіки та звіти, для отримання цінності даних. Оскільки задіяно великий обсяг даних, системі знадобиться час для їх обробки. Пакетна обробка даних економити обчислювальні ресурси.

Віддаючи перевагу пакетній обробці над обробкою в реальному часі, коли точність важлива для швидкості. Крім того, можна виміряти ефективність пакетної обробки з огляду на пропускну здатність. Пропускна здатність — це кількість даних, що обробляються за одиницю часу.

Обробка транзакцій − це тип обробки даних, який обробляє «транзакції» − події або транзакції, які мають бути записані та збережені. Загалом, це передбачає запис таких дій, як продажі та покупки, у базу даних.

Обробка транзакцій розгортається в критично важливих ситуаціях. Це ситуації, які, якщо їх порушити, негативно вплинуть на бізнес-операції − наприклад, обробку біржових операцій, як згадувалося раніше. При обробці транзакцій доступність є найважливішим фактором. На доступність впливають такі фактори, як:

− апаратне забезпечення: система обробки транзакцій повинна мати апаратне резервування, яке допускає часткові збої. Це тому, що надлишкові компоненти автоматизовані, щоб взяти на себе керування та підтримувати роботу комп’ютерної системи;

− програмне забезпечення: програмне забезпечення системи обробки транзакцій повинно швидко відновлюватися після збою. Як правило, системи обробки транзакцій використовують для цього абстракцію транзакцій. Простіше кажучи, у разі збою незафіксовані транзакції припиняються. Це дозволяє системі, як процесор, швидко перезавантажуватися.

Обробка в реальному часі – це процес обчислення даних, щойно вони створені або отримані. Це форма розподіленої обробки, яка дозволяє фіксувати й аналізувати потоки вхідних даних, дозволяючи швидко реагувати на дані аналізу.

Обробка в реальному часі схожа на обробку транзакцій тим, що використовуючи її в ситуаціях, коли, відповідно, очікується результат у реальному часі. Однак вони відрізняються тим, як вони справляються з втратою даних. Обробка в реальному часі обчислює вхідні дані якомога швидше. Якщо він стикається з помилкою у вхідних даних, він ігнорує цю помилку та переходить до наступної частини введених даних. Програми GPS-стеження є найпоширенішим прикладом обробки даних у реальному часі.

Порівнюючи це з обробкою транзакцій. У разі помилки, наприклад збою системи, обробка транзакцій припиняє поточну обробку та повторно ініціалізує. Можна віддати перевагу обробці в реальному часі над обробкою транзакцій у випадках, коли достатньо орієнтовних відповідей.

У світі аналізу даних потокова обробка є звичайним застосуванням обробки даних у реальному часі. Вперше популяризований Apache Storm, потокова обробка аналізує дані, коли вони надходять. Google BigQuery та Snowflake є прикладами хмарних платформ даних, які використовують обробку в реальному часі. Виходячи з цього потрібен певний підхід для балансування або планування навантажень, проаналізую їх у кваліфікаційній роботі далі.

Програмно конфігуруванні мережі — це підхід до мережевої архітектури, який дозволяє мережевим інженерам і адміністраторам централізовано контролювати або програмувати мережу та її трафік за допомогою програмних додатків. SDN допомагає компаніям залишатися адаптованими, дозволяючи мережевим інженерам ефективно керувати мережевими службами для пристроїв якщо є необхідність.

SDN подібний, але відрізняється від SD-WAN. SD-WAN народжується з SDN і справді застосовує ті ж основні концепції, що й SDN, для швидкого й ефективного спрямування мережевого трафіку. Однак SDN має менший географічний діапазон, ніж SD-WAN, який може охоплювати більший плацдарм. Ще одна ключова відмінність між ними полягає в тому, що користувачі (адміністратори мережі) програмують SDN, тоді як постачальник забезпечує та оптимізує послуги, які відповідно надає SD-WAN.

Це може здатися парадоксальним, але SDN допомагають компаніям створити інтегровану мережеву екосистему, але водночас відокремлюють площину керування від площини даних у мережі. Це дозволяє програмному забезпеченню працювати незалежно від пристрою та операційної системи, навіть на межі мережі. Програмне забезпечення все ще доступне для мережевих комутаторів і маршрутизаторів, які інакше стояли б за закритою фірмовою мікропрограмою.

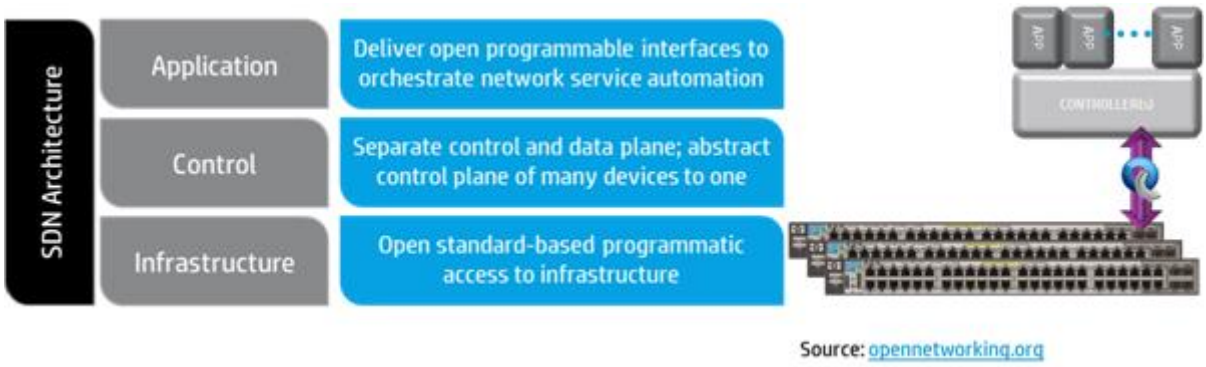

Рис. 1.1. Архітектура контролера SDN

Згідно з рисунком 1.1, помітно, що SDN (Software-Defined Networking) контролер не лише дозволяє класичне управління мережею за допомогою прямих команд системного адміністратора до контролера, але також підтримує запуск додатків управління мережею безпосередньо на ньому.

Є чотири різні типи SDN.

– API: контролює потік даних між рівнями управління та інфраструктури через інтерфейси програмування;

– відкритий: використовує відкриті протоколи для маршрутизації зв'язку між віртуальними та апаратними пристроями;

– накладення: створює карту віртуального рівня поверх апаратної екосистеми, щоб забезпечити сегментований доступ і пропускну здатність між пристроями та центрами обробки даних;

– гібридний: об'єднує традиційні та програмно-визначені мережі, призначаючи найкращий протокол залежно від типу трафіку даних.

SDN складається з трьох рівнів (частин): програми, керування та інфраструктури, а саме:

– рівень програми на якому містяться програми мережі. Програми надсилають запити на рівень керування мережею. SDN використовує API для взаємодії з програмами та керування ними через контролер, що виконується програмою;

– контрольний рівень або рівень керування (площина керування) містить контролер і запускає всі служби на всіх пристроях у мережі з одного центрального

місця. Тому цей рівень часто влучно називають мозком мережі. Рівень керування діє як посередник між програмами, що працюють на пристроях, і основною інфраструктурою комутатора. Він направляє запити на рівень інфраструктури, встановлює маршрути та призначає часові та частотні інтервали. Контролер буває трьох різних типів відповідно до архітектури мережі SDN: централізований, розподілений і гібридний. Правильний контролер для будь-якого конкретного обчислювального середовища залежить від розміру системи, а також від бажаного рівня безпеки, стійкості та масштабованості. Розподілений контролер є хорошим компромісом між безпекою та складністю системи;

− рівень інфраструктури, який також називають площиною даних або прямою площиною, є основою мережі. Він містить фізичні пристрої, відомі як пристрої площини даних, а саме маршрутизатори та комутатори. Цей рівень в основному регулює та сегментує трафік усередині та між мережами. Комутатори змінюють трафік у мережі, спрямовуючи трафік туди, куди він найбільше потрібний у будь-який момент, і визначаючи спосіб його доставки. Вони схожі на м’язи тіла, які діють на основі повідомлень, які надсилаються з мозку (керування). Рівень інфраструктури повинен мати можливість підтримувати різні типи пристроїв і операційних систем, щоб додатки працювали належним чином;

Технологія сегментної маршрутизації є однією з кількох, які виникли завдяки SDN. Сегментна маршрутизація додає до мережі рівень безпеки, який захищає від циклічних атак і DoS-атак, а також у поєднанні з програмно визначеним периметром розрізняє надійні та зловмисні пристрої, які намагаються підключитися до мережі.

Secure Access Service Edge (SASE) − це комплексний набір функцій безпеки, включаючи WAN і Zero Trust , які підвищують безпеку SDN і зміцнюють безпеку мережі.

SDN, особливо в поєднанні з віртуалізацією мережевих функцій (NFV) , забезпечує необхідну підтримку для запуску, живлення та оптимізації інфраструктури 5G.

Як мережевий підхід, який програмує мережі за допомогою програмного забезпечення, а не апаратного забезпечення, SDN став еталоном стійкої, швидко

адаптованої мережевої стратегії підприємств. SDN розвинувся, щоб включити ряд типів SDN, що дозволяє компаніям будь-якого розміру скористатися перевагами SDN, такими як гнучкість, простота та підтримка новітніх технологій, таких як 5G.

1.2. Проблеми енергоефективності та вимоги щодо обслуговування різних типів навантаження

У розробці продуктів у сфері телекомунікацій враховуються визначені правила та стандарти, що стосуються стабільності, дотримання протоколів та забезпечення якості, що відображається використанням поняття операторського рівня для позначення обладнання, яке демонструє цей високий фактор надійності та продуктивності [3]. Незважаючи на те, що ця модель добре працювала в минулому, вона неминуче призвела до довгих циклів продукту, повільного темпу розробки та залежності від спеціального апаратного забезпечення, наприклад, індивідуальних інтегральних схем для конкретного застосування.(ASIC). Ця модель розробки призвела до значних затримок під час розгортання нових послуг, створила складні проблеми взаємодії та значне збільшення CAPEX/OPEX під час масштабування мережевих систем та інфраструктури та розширення можливостей мережевих послуг для задоволення зростаючих вимог до навантаження на мережу та продуктивності. Крім того, зростання значної конкуренції в пропозиціях комунікаційних послуг з боку гнучких організацій, що працюють у великих масштабах у загальнодоступному Інтернеті (таких як Google Talk , Skype , Netflix), спонукало постачальників послуг шукати інноваційні способи порушити статус-кво та збільшити потоки доходів .

Віртуалізація мережевих функцій (NFV) [1] − це концепція мережевої архітектури, яка використовує технології IT-віртуалізації для віртуалізації цілих класів функцій мережевих вузлів у певні блоки, які можуть з’єднуватися або з’єднуватися разом для створення та надання послуг зв’язку.

NFV (Network Function Virtualization) базується на традиційних методах віртуалізації серверів, які використовуються в корпоративному IT. Віртуалізовані мережеві функції, або VNF, реалізовані за допомогою однієї або кількох віртуальних

машин або контейнерів, на яких працює різне програмне забезпечення та процеси. Вони використовують комерційно доступні (COTS) сервери, комутатори, пристрої зберігання даних або навіть інфраструктуру хмарних обчислень, замість спеціальної апаратної інфраструктури для кожної мережевої функції. Це уникає потреби у блокуванні від постачальників.

Наприклад, контролер границі віртуального сеансу може бути розгорнутий для захисту мережі без необхідності встановлення фізичних блоків захисту мережі, що звичайно пов'язано з витратами і складністю. Інші приклади використання NFV включають віртуальні балансувальники навантаження, брандмауери, системи виявлення вторгнень та прискорювачі WAN [2].

Відокремлення програмного забезпечення мережевих функцій від налаштованої апаратної платформи реалізує гнучку мережеву архітектуру, яка забезпечує гнучке керування мережею, швидке розгортання нових послуг зі значним скороченням капітальних і експлуатаційних витрат, що є на сьогоднішній день досить актуальним.

NFV (Network Function Virtualization) має структуру, що складається з трьох основних компонентів [9]:

Віртуалізовані мережеві функції (VNF): Це програмні реалізації мережевих функцій, які можуть бути розгорнуті в інфраструктурі віртуалізації мережевих функцій (NFVI). Вони представляють собою віртуальні машини або контейнери, на яких працює програмне забезпечення, що здійснює необхідні мережеві функції [10].

Інфраструктура віртуалізації мережевих функцій (NFVI): Це сукупність апаратних і програмних компонентів, які створюють середовище для розгортання NFV. Інфраструктура NFV може включати різні місця розташування, а мережа, що забезпечує зв'язок між цими місцями, також вважається частиною інфраструктури NFV.

Архітектурна структура управління віртуалізацією мережевих функцій і оперування (Architectural Framework NFV-MANO): Це сукупність функціональних блоків, сховищ даних, які використовуються цими блоками, а також контрольних

точок та інтерфейсів, через які ці функціональні блоки обмінюються інформацією з метою керування та оперування NFVI і VNFs.

Ця архітектура визначає спосіб керування віртуалізованими мережевими функціями та оперативного управління віртуалізованою інфраструктурою.

Платформа NFV (Network Function Virtualization) є основою для компонентів NFVI (Network Function Virtualization Infrastructure) і NFV-MANO (Network Function Virtualization Management and Orchestration).

NFVI складається з віртуальних та фізичних ресурсів обробки і зберігання, а також програмного забезпечення віртуалізації. Вона надає інфраструктуру для розгортання і виконання віртуалізованих мережевих функцій (VNF). NFVI забезпечує ресурси, які потрібні для функціонування VNF, включаючи обчислювальні ресурси, пам'ять, сховище та мережеві ресурси.

NFV-MANO включає менеджерів VNF і NFVI, а також програмне забезпечення віртуалізації, яке працює на апаратному контролері. Його завдання полягає у керуванні і оркестрації віртуалізованими мережевими функціями та інфраструктурою. NFV-MANO забезпечує функції керування, моніторингу, відновлення після збоїв і безпеки для компонентів платформи NFV. Він дозволяє операторам мереж забезпечувати ефективну управління і нагляд за віртуалізованими мережевими функціями та їх ресурсами.

Усі ці компоненти платформи NFV взаємодіють для забезпечення розгортання, керування та управління віртуалізованими мережевими функціями в операторському середовищі.

Початкова концепція NFV дійсно передбачала розташування віртуалізованих функцій у центрах обробки даних. Однак, згодом стало очевидним, що такий підхід не є універсальним і не підходить для всіх випадків.

NFV надає гнучкість щодо фізичного розташування віртуалізованих функцій, підкреслюючи, що вони можуть бути розташовані там, де це є найбільш ефективним і економічно вигідним. Це означає, що постачальники послуг мають можливість розгортати NFV в різних місцях, включаючи центри обробки даних, мережеві вузли та приміщення клієнтів. Цей підхід, відомий як розподілений NFV, був акцентований

з самого початку розробки та стандартизації NFV, і це підтверджується останніми документами від NFV ISG (Industry Specification Group).

Розподілений NFV надає більшу гнучкість у розташуванні віртуалізованих функцій і дозволяє їх розгортання там, де це найбільш доцільно. Наприклад, деякі функції можуть бути ефективними в центрах обробки даних, тоді як інші можуть бути більш корисними у мережевих вузлах або навіть безпосередньо у приміщеннях клієнтів. Розподілений підхід до NFV дозволяє використовувати відповідні ресурси і місцезнаходження для забезпечення найбільшої ефективності та вигоди від використання віртуалізованих функцій мережі [12].

У деяких ситуаціях, постачальникам послуг вигідно розміщувати віртуалізовану функціональність безпосередньо на території своїх клієнтів. Це надає їм різноманітні переваги, які охоплюють економічні аспекти, продуктивність та можливості віртуалізації функцій [13].

Підтвердження концепції D-NFV було здійснено публічним PoC (Proof of Concept) ETSI NFV ISG, проведеним Cyan, Inc., RAD, Fortinet і Certes Networks у Чикаго в червні 2014 року за підтримки CenturyLink. Цей PoC базувався на спеціальному обладнанні D-NFV від RAD, яке використовувало брандмауер наступного покоління Fortinet (NGFW) та механізм віртуального шифрування/дешифрування Certes Networks як віртуалізовані мережеві функції (VNF), а управління системою здійснювалося за допомогою Cyan Blue Planet, що керує всією екосистемою. Рішення RAD D-NFV включало мережевий кінцевий блок (NTU) рівня 2 та рівня 3 з D-NFV X86 серверним модулем, який функціонує як механізм віртуалізації на стороні клієнта. Протягом 2014 року RAD також заснував D-NFV Alliance, екосистему постачальників і міжнародних системних інтеграторів, які спеціалізуються на нових програмах NFV [16].

При проектуванні та розробці програмного забезпечення для віртуалізованих мережевих функцій (VNF), постачальники можуть структурувати його на компоненти програмного забезпечення, що представляють реалізацію архітектури. Ці компоненти можуть бути упаковані в одне або кілька зображень, що відображають розгортання архітектури програмного забезпечення. Ці компоненти програмного забезпечення,

визначені постачальником, називаються компонентами VNF (VNFC). Кожен VNF може бути реалізований з одного або декількох VNFC, і за загальною умовою можна стверджувати, що екземпляри VNFC відповідають 1:1 зображенням віртуальних машин (VM).

Компоненти VNFC повинні бути здатні до масштабування, як вертикального, так і горизонтального. За допомогою гнучкого розподілу віртуальних ЦП для кожного екземпляра VNFC, рівень керування мережею може здійснювати вертикальне масштабування VNFC для досягнення необхідної пропускної здатності або продуктивності в межах однієї системи або платформи. Також, шляхом активації декількох екземплярів VNFC на різних платформах, рівень керування мережею може здійснювати горизонтальне масштабування VNFC. Це дозволяє досягати вимог щодо продуктивності та архітектури, зберігаючи стабільність інших функцій VNFC.

Віртуалізація мережевих функцій (NFV) є важливим доповненням до програмно-визначеної мережі (SDN). SDN - це підхід до розробки мережевого обладнання та програмного забезпечення, який розділяє та абстрагує їх елементи. Це досягається шляхом відокремлення площини керування від площини даних, при цьому площина керування централізована, а компоненти пересилання даних залишаються розподіленими. Площина керування взаємодіє як зверху, так і знизу. Зверху вона надає абстрактне уявлення про мережу для додатків та програм вищого рівня, використовуючи високорівневі API та новітні керувальні парадигми, наприклад, мережу на основі намірів. Знизу вона програмує поведінку компонентів пересилання даних, використовуючи API рівня фізичного мережевого обладнання, яке розподілене по мережі.

NFV і SDN можуть взаємодіяти для поліпшення управління інфраструктурою NFV та створення більш динамічного мережевого середовища. Є можливість реалізувати віртуальну мережу (VNF) як окрему сутність, використовуючи існуючі мережеві парадигми та процеси. Однак використання концепцій SDN для впровадження та керування інфраструктурою NFV має численні переваги, особливо при управлінні мережевими службами (NS), які складаються з різних типів мережевих функцій (NF), таких як фізичні мережеві функції (PNF) і VNF, і

розташовані у різних просторових розташуваннях інфраструктури NFV. У цьому випадку використовуються мультивендорні платформи, які об'єднують SDN і NFV у взаємодіючі системи [18].

Система NFV потребує централізованого оперування та системи керування, яка приймає запити оператора, пов’язані з NS або VNF, перетворює їх у відповідну обробку, зберігання та мережеву конфігурацію, необхідну для запуску NS або VNF. Після запуску VNF і мереж, до яких він потенційно підключений, необхідно перевіряти на ємність і використання, а також адаптувати, якщо необхідно [19].

Усі функції керування мережею в інфраструктурі NFV можна виконати за допомогою концепцій SDN, NFV можна вважати одним із основних випадків використання SDN у середовищах постачальників послуг [20]. Наприклад, у межах кожного сайту інфраструктури NFV VIM може покладатися на контролер SDN для встановлення та налаштування накладених мереж, що з’єднують (наприклад, VXLAN) VNF та PNF, що утворюють NS. Тоді контролер SDN за потреби налаштує комутатори і маршрутизатори інфраструктури NFV, а також мережеві шлюзи. Подібним чином диспетчер глобальної інфраструктури (WIM) може покладатися на контролер SDN для налаштування накладених мереж для з’єднання NS, які розгорнуті в різних географічних інфраструктурах NFV.

З 2018 року багато постачальників VNF почали переносити багато своїх VNF на контейнерну архітектуру. Такі VNF, також відомі як Cloud-Native Network Functions (CNF), використовують багато інновацій, які зазвичай розгортаються в інфраструктурі Інтернету. До них належать автоматичне масштабування, підтримка безперервної доставки/моделі розгортання DevOps і підвищення ефективності за рахунок спільного використання спільних сервісів на платформах. Завдяки виявленню та оркестровці послуг мережа на основі CNF стане більш стійкою до збоїв інфраструктурних ресурсів. Використання контейнерів і, таким чином, відмова від накладних витрат, притаманних традиційній віртуалізації, шляхом усунення гостьової ОС може значно підвищити ефективність ресурсів інфраструктури.

До мереж 1 Гбіт/с мережева віртуалізація не страждала від накладних витрат програмних рівнів або рівнів гіпервізора, що забезпечують міжз’єднання. Зі

зростанням високої пропускної здатності, 10 Гбіт/с і вище, швидкість пакетів перевищує можливості обробки мережевих стеків. Щоб продовжувати пропонувати обробку з високою пропускною здатністю, розгортаються деякі комбінації програмного та апаратного забезпечення. у так званій «мережі в коробці», пов'язаній або з апаратно-залежним контролером мережевого інтерфейсу (NIC), використовуючи розширення SRIOV гіпервізора, або з використанням технології швидкого шляху між NIC і корисним навантаженням (віртуальними машинами або контейнерами).

Наприклад, у випадку Openstack мережу забезпечує Neutron, який використовує багато функцій ядра Linux для роботи в мережі: iptables, iprout 2, L2 bridge, L3 routing або OVS. Оскільки ядро Linux не може підтримувати швидкість пакетів 10G, то використовуються деякі технології обходу для швидкого шляху. Основні технології обходу засновані або на обмеженому наборі функцій, таких як Open vSwitch (OVS) із реалізацією простору користувача DPDK, або на повній функції та розвантаженні обробки Linux, наприклад віртуальний прискорювач 6WIND.

Серед вимог обробки навантажень для комп'ютерів, які живляться від батареї (наприклад, ноутбуки та смартфони), або для дуже тривалих/великих обчислень (наприклад, суперкомп'ютери)

- пряме споживання електроенергії : потужність, необхідна безпосередньо для роботи комп'ютера;
- непряме споживання електроенергії : потужність, необхідна для охолодження, освітлення тощо.

Станом на 2018 рік енергоспоживання зростає як важливий показник для обчислювальних завдань усіх типів і в усіх масштабах, починаючи від вбудованих пристроїв Інтернету речей (IoT) і закінчуючи системами на чіпі та серверними фермами . Цю тенденцію часто називають зеленими обчисленнями .

Менш поширені показники обчислювальної ефективності також можуть бути доречними в деяких випадках:

- розмір передачі: пропускна здатність може бути обмежуючим фактором. Стиснення даних використовується за для зменшення об'єму даних, які потрібно

передати. Відображення зображення або зображень (наприклад, логотип Google) може призвести до передачі десятків тисяч байтів (у цьому випадку 48 КБ) порівняно з передачею шести байтів для тексту "Google". Це важливо для обчислювальних завдань, пов’язаних із вводом-виводом;

− зовнішній простір: необхідний простір на диску чи іншому зовнішньому пристрої пам’яті; можливе тимчасове зберігання, поки виконується алгоритм, або довгострокове зберігання, яке необхідно перенести для подальшого використання;

− час відповіді (затримка): це особливо актуально в програмах реального часу , коли комп’ютерна система повинна швидко реагувати на зовнішню подію ;

− загальна вартість володіння: особливо якщо комп’ютер призначений для одного певного алгоритму.

В обчислювальній техніці масштабованість є характеристикою комп’ютерів, мереж, алгоритмів , мережевих протоколів , програм і додатків. Прикладом є пошукова система , яка повинна підтримувати зростаючу кількість користувачів і кількість тем, які вона індексує [3]. Webscale − це комп’ютерний архітектурний підхід, який передає можливості великих хмарних обчислювальних компаній у корпоративні центри обробки даних.

ВИСНОВОКИ ДО РОЗДІЛУ 1

Проаналізовано основні проблеми та особливості процесу обробки навантажень в інформаційній мережі, а також вимоги різних типів навантаження на інформаційну систему.

Завдяки NFV постачальники послуг можуть запускати мережеві функції на стандартному апаратному забезпеченні замість спеціального обладнання. Крім того, оскільки мережеві функції віртуалізовані, кілька функцій можна запускати на одному сервері. Це означає, що потрібно менше фізичного обладнання, що дозволяє консолідувати ресурси, що призводить до фізичного простору, потужності та загального зниження витрат.

NFV надає постачальникам гнучкість для запуску VNF на різних серверах або переміщення їх за потреби, коли попит змінюється. Ця гнучкість дозволяє постачальникам послуг швидше надавати послуги та програми.

Наприклад, якщо клієнт запитує нову мережеву функцію, він може створити нову віртуальну машину для обробки цього запиту. Якщо функція більше не потрібна, віртуальну машину можна вивести з експлуатації. Це також може бути безпечним способом перевірити цінність потенційної нової послуги.

РОЗДІЛ 2
ІСНУЮЧІ ПІДХОДИ ПІДВИЩЕННЯ ЕНЕРГОЕФЕКТИВНОСТІ ОБРОБКИ НАВАНТАЖЕНЬ В ІНФОРМАЦІЙНІЙ МЕРЕЖІ

2.1. Загальні положення енергоефективної оброки навантажень

Енергоефективна обробка навантажень слугує для зниження споживання електроенергії, а також збільшення продуктивності. У енергоефективну обробку навантажень в інформаційній мережі входять наступні аспекти:

- віртуалізація та консолідація серверів;
- енергоефективне обладнання;
- енергоефективне керування обладнанням;
- керування температурою та охолодженням;
- оптимізація мережевих протоколів;
- моніторинг та аналітика.

Віртуалізація та консолідація серверів: Віртуалізація є ключовою стратегією для енергоефективної обробки навантажень в інформаційній мережі. Вона дозволяє запускати кілька віртуальних серверів на одному фізичному сервері, що знижує кількість фізичних серверів та споживання електроенергії. Крім того, консолідація серверів полягає в об'єднанні різних функцій та додатків на одному сервері, що також знижує споживання енергії та вимоги до інфраструктури.

Енергоефективне обладнання: Вибір енергоефективного обладнання є важливим аспектом енергоефективної обробки навантажень. Енергоефективні сервери, комутатори та маршрутизатори використовують менше енергії для виконання своїх завдань. При виборі обладнання слід звертати увагу на сертифікати енергоефективності, такі як ENERGY STAR, а також дослідження про енергоефективність виробників.

Енергоефективне керування обладнанням: Ефективне керування обладнанням включає управління споживанням електроенергії та вимкненням

обладнання що знаходиться в режимі простою. Наприклад, використання технології "wake-on-LAN" дозволяє вмикати-вимикати комп'ютери за запитом, що знижує споживання електроенергії в періоди не активності. Також можна встановити системи автоматичного вимкнення та перехід у режим сну для обладнання, яке не використовується протягом тривалого часу.

Керування температурою та охолодженням: Контроль температури в серверних приміщеннях та використання енергоефективних систем охолодження допомагають знизити споживання енергії. Ефективне розташування серверних шаф, використання вентиляційних систем з енергоефективними вентиляторами та оптимізація простору для забезпечення вільного потоку повітря можуть покращити енергоефективність охолодження.

Оптимізація мережевих протоколів: Використання енергоефективних мережевих протоколів є важливим для зниження споживання електроенергії в інформаційній мережі. Наприклад, Ethernet Energy Efficient Ethernet (EEE) дозволяє зменшити споживання енергії в мережевому обладнанні шляхом автоматичного переходу в режим низького споживання енергії в періоди низького навантаження або не активності.

Моніторинг та аналітика: Впровадження систем моніторингу та аналітики дозволяє спостерігати за споживанням електроенергії в реальному часі та аналізувати дані для виявлення можливостей оптимізації. Це може включати використання систем управління енергією, аналіз споживання електроенергії на різних рівнях (сервер, мережа, приміщення) та виявлення проблемних зон.

Впровадження цих загальних положень енергоефективної обробки навантажень в інформаційній мережі призводить до таких результатів:

- зниження споживання електроенергії;
- підвищення ефективності ресурсів;
- зменшення негативного впливу на довкілля;
- зниження витрат на електроенергію;
- підвищення надійності;
- підвищення продуктивності мережі.

Згідно документу Міжнародної спілки електрозв'язку ITU-T L.1300 важливим є комплексний підхід, що враховує всі аспекти енергоефективної обробки навантажень. Дослідниками з компанії Google [1] була запропонована концепція «енерго-пропорційні обчислення». Дана концепція розглядає ідеалізовану модель обчислювального вузла, загальне споживання енергії яке пропорційне його завантаженості. Дослідники також проаналізували залежність енергоефективності обчислень від рівня завантаженості сервера (Рис. 2.1).

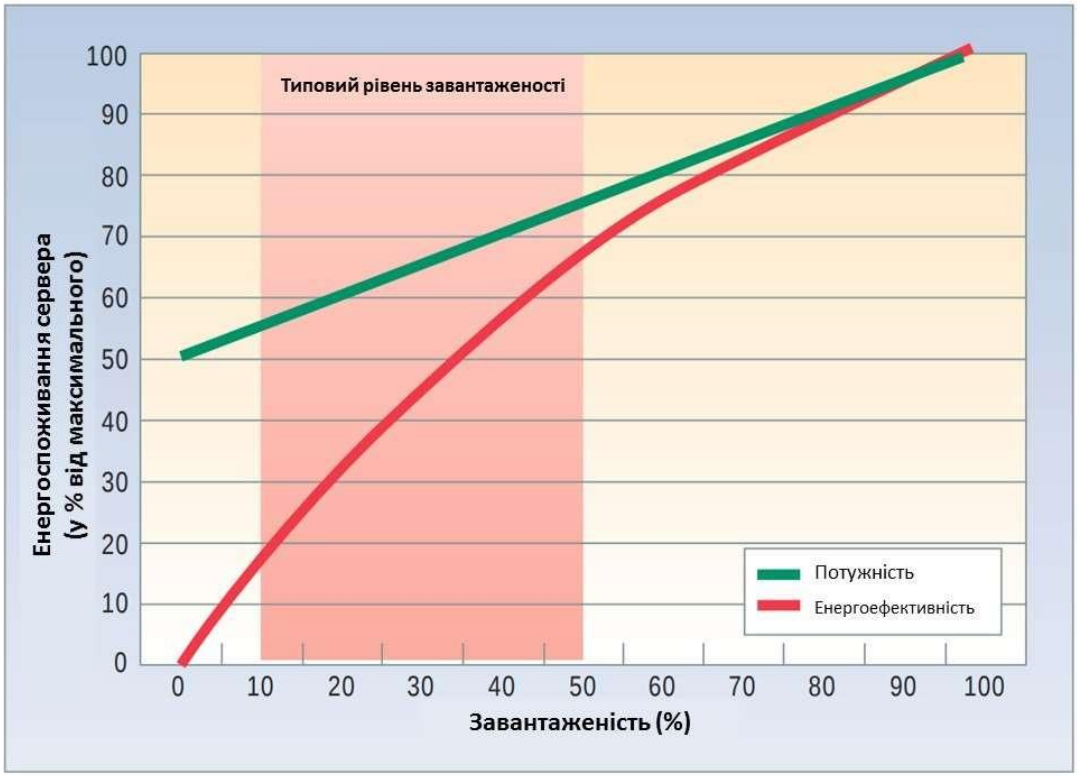

Рис. 2.1. Залежність енергоефективності обчислень від завантаженості обчислювального вузла [1]

Виходячи з графіку зображеному на (Рис. 2.1), коли завантаженість сервера є низькою, його енергоспоживання не є нульовим. Такий результат відповідає режиму простою. Також аналізуючи графік важко не помітити, що чим більша завантаженість сервера, тим більша його енергоефективність. Виходячи з цього основна ідея енергоефективних обчислень полягає в тому щоб уникнути простою обладнання та на максимум використовувати обладнання.

Для наближення до моделі енерго-пропорційних обчислень розроблено багато підходів. Серед таких можна виділити такі підходи як:

- Підходи на рівні апаратного забезпечення [1];
- Горизонтальне масштабування та консолідація [10];
- Енергоефективний розподіл навантаження [11].

2.2. Підхід на рівні апаратного забезпечення

На рівні апаратного забезпечення існує кілька методів, які впливають на енергоефективності та продуктивність в інформаційних мережах:

Використання енергоефективних компонентів: При виборі компонентів для побудови серверів і мережевого обладнання, слід звертати увагу на їх енергоефективність. Наприклад, процесори з низьким рівнем енергоспоживання, які підтримують технології керування енергією, такі як Intel SpeedStep або AMD Cool'n'Quiet, дозволяють знизити енергоспоживання при низькому навантаженні. Використання енергоефективних модулів пам'яті та жорстких дисків також сприяє зниженню споживання енергії.

Розумне керування енергією: Сучасні сервери та мережеве обладнання мають функції керування енергією, які дозволяють автоматично знижувати споживання енергії в періоди не активності або низького навантаження. Наприклад, системи управління енергією Advanced Power Management (APM) та Intelligent Platform Management Interface (IPMI) дозволяють налаштовувати режими сну та глибокого сну для різних компонентів системи.

Енергоефективне охолодження: Охолодження серверних приміщень є важливим аспектом енергоефективної обробки навантажень. Використання енергоефективних систем охолодження, таких як системи з використанням холодної води, вентиляційні системи з енергоефективними вентиляторами або використання систем охолодження з повітрям зовні приміщення, може значно знизити споживання електроенергії. Крім того, використання технологій гарячої айл-головки (hot aisle/cold

aisle) та принципу градієнту температури допомагає оптимізувати охолодження на рівні серверних шаф.

Регулювання швидкості вентиляторів: Багато серверів та мережевого обладнання мають вентилятори для охолодження компонентів. Регулювання швидкості обертання вентиляторів відповідно до температури і навантаження дозволяє знизити споживання енергії, забезпечуючи достатню вентиляцію при необхідності.

Використання віртуалізації та консолідації: Виртуалізація серверів дозволяє об'єднати декілька віртуальних машин на одному фізичному сервері. Це дозволяє використовувати обладнання більш ефективно, зменшуючи кількість фізичних серверів і, відповідно, споживання електроенергії.

Використання енергоефективних мережевих пристроїв: Вибір мережевих комутаторів, маршрутизаторів та інших пристроїв з низьким рівнем енергоспоживання може сприяти зниженню споживання електроенергії в мережі. Деякі пристрої мають функції автоматичного вимкнення портів в разі неактивності або регулювання енергії в залежності від навантаження.

Ефективне розташування обладнання: Оптимальне розташування серверів та мережевого обладнання в приміщенні може сприяти кращому охолодженню та енергоефективності. Розміщення обладнання в так званих "гарячих айл-головках" (ряди серверів звернені один до одного) та забезпечення належного простору для вентиляції між серверами допомагає уникнути перегріву та забезпечити ефективну роботу системи охолодження.

Ці підходи на рівні апаратного забезпечення можуть бути комбіновані та адаптовані відповідно до конкретних потреб інформаційної мережі. Вони спрямовані на зниження споживання електроенергії, покращення енергоефективності та забезпечення оптимальної продуктивності мережі.

2.3 Горизонтальне і вертикальне масштабування

Основне розходження між горизонтальним масштабуванням і вертикальним масштабуванням полягає в тому, що горизонтальне масштабування передбачає додавання більшої кількості машин або вузлів у систему, в той час як вертикальне масштабування передбачає додавання більшої потужності (CPU, RAM, сховища тощо) до існуючого обладнання. Іншими словами, горизонтальне масштабування передбачає зменшення масштабу, у той час як вертикальне масштабування передбачає збільшення. Горизонтальне масштабування зазвичай використовується для обробки зростаючих обсягів трафіку або робочого навантаження, в той час як вертикальне масштабування для обробки ресурсномістких завдань або додатків, що вимагають більшої обчислювальної потужності.

Що таке горизонтальне масштабування?

Горизонтальне масштабування, часто відомий як "масштабування", - це процес збільшення числа вузлів і машин у пулі ресурсів. Щоб послідовний фрагмент логіки оброблявся паралельно на безлічі пристроїв, горизонтальне масштабування вимагає поділу його на більш дрібні фрагменти і делегування логіки новій машині. Простіше кажучи, масштабування по горизонталі полягає в наймі нових співробітників для додаткового набору завдань.

Переваги такого підходу:

- спрощення масштабування за рахунок апаратних доповнень;
- підвищена гнучкість;
- менший час простою;
- пропонує надмірність.

Також він має і недоліки. Серед недолік можна виділити таке: більш високі початкові витрати, а також його важче підтримувати

Що таке вертикальне масштабування?

Вертикальне масштабування, часто відомий як "збільшення масштабу", - це процес збільшення потужності існуючої системи, такий як центральний процесор або оперативна пам'ять, для задоволення зростаючих потреб. Оскільки немає

необхідності змінювати логіку, вертикальне масштабування простіше. Замість цього ви виконуєте той самий код тільки на машинах з більшою продуктивністю.

Пам'ять, сховище або швидкість мережі можна масштабувати по вертикалі. Вертикальне масштабування також може означати повну заміну сервера або перенесення робочого навантаження з застарілого сервера на оновлений. Простіше кажучи, вертикальне масштабування - це підвищення кваліфікації існуючих співробітників для додаткового набору клієнтів / завдань.

Переваги такого підходу:

- економічно ефективний;
- менша складність;
- простіше в обслуговуванні.

Недоліки:

- більше можливостей для простою;
- набагато менша гнучкість;
- єдина точка відмови.

2.4. Енергоефективний розподіл навантаження

Окрім консолідації та горизонтального масштабування, енергоефективне планування є важливим підходом для підвищення ефективності використання енергії. Основна мета цього процесу - встановити відповідність між ресурсами обробки даних та завданнями, які потребують обробки. Простіші методи планування включають FCFS (першим прийшов - першим обслугований) та RoundRobin (RR) [11]. FCFS розподіляє завдання в порядку їх надходження. Якщо недостатньо ресурсів для виконання наступного завдання, алгоритм очікує їх звільнення. RR розподіляє ресурси між завданнями по колу. Ці підходи не враховують енергоефективність під час розподілу завдань.

Згідно з діаграмою (рис. 2.1), чим більше завантаженість сервера, тим більша його енергоефективність. Тому для енергоефективного розподілу завдань важливо

встановити відповідність між обчислювальними ресурсами та завданнями, щоб мінімізувати простої обладнання.

Автори роботи [12] розглядають дворівневий підхід до розподілу завдань CTES - Cooperative Two-Tier Energy-Aware Scheduling, призначений для обробки завдань у режимі реального часу. На першому рівні розподіляються ресурси одного сервера між завданнями таким чином, щоб кожне завдання отримало мінімальну необхідну потужність для обробки. Усі залишкові ресурси розподіляються між всіма завданнями. На другому глобальному рівні кластер серверів розподіляє завдання за допомогою однієї з технік, що описані у дослідженні: Predictive Energy, Predictive Total Time, ET2. Ці техніки враховують як показники енергоефективності, так і продуктивності під час розподілу завдань. Однак, недоліком цього алгоритму є те, що автори використали лінійну модель енергоспоживання сервера (залежність потужності від завантаження центрального обчислювального вузла). На практиці така модель не точно відображає реальну залежність споживання потужності від завантаження сервера. Замість цього, у цьому дослідженні пропонується попередня атестація кожного сервера кластера для отримання реальної залежності потужності споживання від завантаження для кожного обчислювального вузла.

Стратегія Backfill [13] є однією з найбільш поширених стратегій енергоефективного планування задач. Початковий варіант цієї стратегії був розроблений компанією IBM і послужив основою для розвитку численних модифікацій та поліпшень цього підходу [14].

Стратегія Backfill є оптимізаційним підходом до розподілу навантаження, що дозволяє краще використовувати наявні апаратні ресурси шляхом розподілу обчислювальних задач у нестандартному порядку, а не в порядку їх надходження до черги. Основна ідея стратегії Backfill полягає у "щільному заповненні" вільних ресурсів за допомогою задач із черги, які, враховуючи їх обсяг, можуть бути виконані. Застосовуючи стратегію Backfill, планувальник спочатку резервує ресурси для "великих" задач у черзі і обчислює час їх початку виконання, враховуючи звільнення необхідних ресурсів у системі. Потім планувальник оцінює, чи можна використати незайняті ресурси до початку виконання "великих" задач для виконання "швидких"

задач з черги до моменту приходу "великої" задачі. Таким чином, планувальник вибирає задачі з меншим пріоритетом, але меншого обсягу, для заповнення "вільних місць" на ресурсах обробки. Застосування стратегії Backfill дозволяє скоротити час виконання обчислювальних задач за рахунок більш ефективного використання доступних ресурсів, як показано на рисунку 2.3 в порівнянні зі стратегією FCFS.

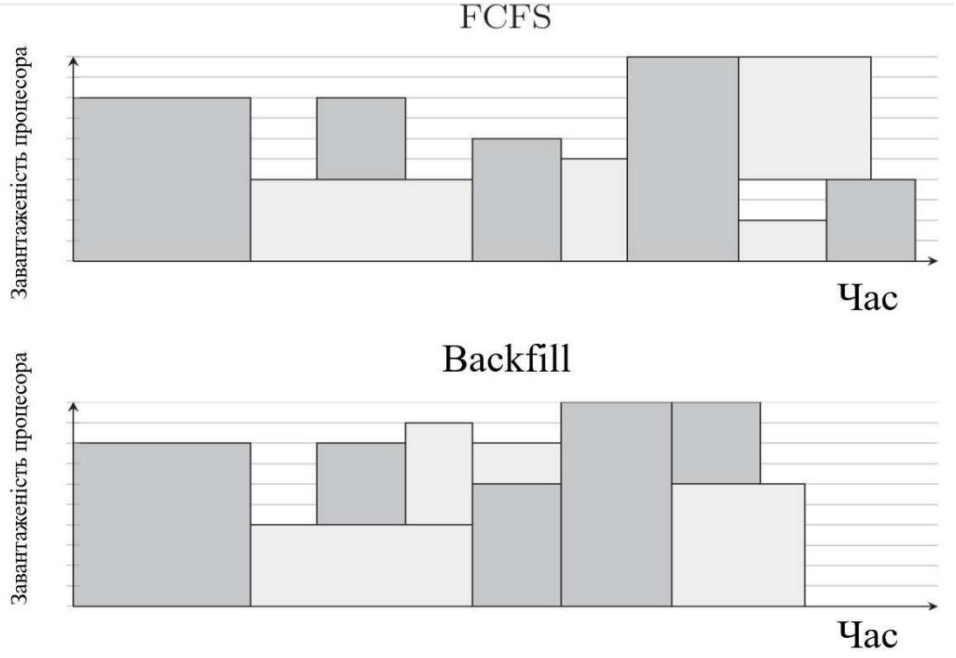

Рис. 2.2. Принцип роботи стратегії Backfill у порівнянні з FCFS

Стратегія Backfill та її модифікації дійсно сприяють підвищенню енергоефективності шляхом забезпечення високого рівня завантаженості серверів. Однак, сама по собі ефективність цього підходу без застосування додаткових технік значно не перевищує ефективності найпростіших підходів, таких як FCFS та RR. Це пояснюється тим, що усі ресурси системи знаходяться в активному стані та споживають електроенергію, навіть коли вони не використовуються (так само, як це відбувається і при використанні підходів FCFS та RR). Для досягнення вищої енергоефективності часто застосовуються додаткові техніки, такі як управління потужністю, динамічне вимкнення ресурсів та інші, які спрямовані на оптимізацію споживання електроенергії в системі.

Алгоритм The Most-Efficient-Server First (MES-first), запропонований у роботі [15], є ще одним відомим підходом до енергоефективного розподілу навантаження.

Згідно з цим підходом, завдання з черги спочатку розподіляються на найбільш енергоефективні сервери, а потім на менш енергоефективні, відповідно до їх позиції у відсортованому списку. Розподіл завдань припиняється, якщо в черзі не залишилося завдань для обробки або якщо черги до всіх серверів заповнені. Цей підхід показує найбільшу ефективність, коли використовується разом з підходами масштабування ресурсів, де менш енергоефективні сервери знаходяться в режимі сну, доки активні енергоефективні сервери забезпечують обробку поточного навантаження.

Недоліком запропонованого у роботі [15] підходу є те, що, на відміну від стратегії Backfill, його цілью не є досягнення максимального рівня завантаженості кожного окремого сервера. Як показано на зображені (Рис. 2.1), недостатня завантаженість вузла обробки призводить до зниження енергоефективності його роботи. Це може відбутися через неоптимальний розподіл завдань і використання ресурсів. Для досягнення оптимальної енергоефективності і продуктивності системи важливо збалансувати завантаженість між серверами та враховувати їхню енергоефективність.

Отже, алгоритм Backfill для енергоефективного планування навантаження є відповідним для вирішення завдання підвищення енергоефективності обробки даних, оскільки він враховує концепцію енерго-пропорційних обчислень. Цей алгоритм може бути використаний у вашому дисертаційному дослідженні. Проте, існує потреба вдосконалити Backfill з метою поліпшення його роботи у випадках недостатньої завантаженості системи і пошуку оптимального балансу між енергоефективністю та продуктивністю обробки даних при помірній завантаженості системи.

ВИСНОВКИ ДО РОЗДІЛУ 2

Було проведено огляд існуючих підходів до підвищення енергоефективності процесу обслуговування навантаження в інформаційно-комунікаційних мережах (ІКМ) та проаналізовано їх ефективність з урахуванням визначених показників, таких як енергоефективність, продуктивність та доступність системи. Серед цих підходів виділяються апаратні та програмні методи, які можна класифікувати за підходами до горизонтального масштабування, консолідації та енергоефективного розподілу навантаження.

Однак, підходи на рівні апаратного забезпечення мають свої недоліки, такі як складність впровадження та високі витрати. Методи масштабування та консолідації можуть знизити загальне енергоспоживання системи, але вони можуть негативно впливати на продуктивність та доступність, особливо при змінних навантаженнях, що є критичним для гарантування якості обслуговування в ІКМ. Крім того, існуючі методи масштабування та консолідації не враховують індивідуальні особливості енергоспоживання обчислювальних вузлів, що призводить до неефективного використання ресурсів.

Серед підходів до енергоефективного планування навантаження особливу увагу заслуговує алгоритм Backfill, який мінімізує простої обчислювальних вузлів шляхом щільного розподілу робіт. Проте, ефективність цього підходу зменшується при низькій інтенсивності навантаження, оскільки він не забезпечує ефективне використання ресурсів у таких умовах.

Існує також проблема, що жоден з існуючих методів не вирішує завдання підвищення ефективності обслуговування навантаження з урахуванням усіх трьох показників (енергоефективність, продуктивність та доступність системи), які є важливими для навантаження в ІКМ. Тому є потреба в систематизації та формалізації процесу обслуговування навантаження в ІКМ з урахуванням цих показників.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО ІСНУЮЧИХ РІШЕНЬ ЩОДО ЗАБЕЗПЕЧЕННЯ ЕНЕРГОЕФЕКТИВНОСТІ ОБРОБКИ НАВАНТАЖЕНЬ

3.1. Дослідження Autopilot та аналіз його особливостей

У багатьох публічних і приватних хмарних системах користувачам потрібно вказати обмеження на кількість ресурсів (ядра ЦП і ОЗП), щоб забезпечити їх робоче навантаження. Завдання, яке перевищує ліміти, може бути придушено або припинено, що призведе до затримки або відхилення запитів кінцевих користувачів, тому люди-оператори, природно, проявляють обережність і запитують ліміт, більший, ніж вимагається для завдання. У великих масштабах це призводить до величезної сукупної втрати ресурсів.

Щоб вирішити цю проблему, Google використовує автопілот для автоматичного налаштування ресурсів, регулюючи як кількість одночасних завдань у завданні (горизонтальне масштабування), так і ліміти ЦП/пам’яті для окремих завдань (вертикальне масштабування). Автопілот працює так само, як і людина-оператор: його головна мета — зменшити провисання- різниця між обмеженням і фактичним використанням ресурсу - водночас мінімізуючи ризик того, що завдання буде припинено через помилку браку пам'яті (OOM) або його продуктивність погіршиться через дроселювання ЦП.

Автопілот використовує алгоритми машинного навчання, застосовані до історичних даних про попередні виконання завдань, а також набір точно налаштованих евристик, щоб пройти цю лінію. На практиці завдання з автопілотом мають слабину лише на 23%, у порівнянні з 46% для завдань, керованих вручну. Крім того, Autopilot зменшує кількість робочих місць, які серйозно постраждали від OOM, у 10 разів.

Незважаючи на його переваги, забезпечення широкого впровадження автопілота вимагало значних зусиль, зокрема надання потенційних рекомендацій легко видимим клієнтам, які ще не погодилися, автоматичне переміщення певних категорій вакансій і додавання підтримки для спеціальних рекомендацій. На момент написання статті роботи з автопілотом займають понад 48% використання ресурсів Google у всьому автопарку.

Багато типів публічних і приватних хмарних систем вимагають від своїх користувачів декларувати, скільки екземплярів буде потрібно їхньому робочому навантаженню під час виконання, а також ресурси, необхідні для кожного з них: у публічних хмарних платформах користувачі повинні вибрати тип і кількість віртуальних машин (ВМ), які вони будуть орендувати, у кластері Kubernetes користувачі встановлюють кількість реплік подів і ліміти ресурсів для окремих подів, у Google просять користувачів вказати кількість контейнерів, які їм потрібні, і ліміти ресурсів для кожного з них.

Такі ліміти роблять хмарні обчислення можливими, дозволяючи хмарній інфраструктурі забезпечити належну ізоляцію продуктивності. Але ліміти - це (здебільшого) неприємність для користувача. Важко оцінити, скільки ресурсів потрібно для оптимального виконання завдання: правильна комбінація потужності процесора, пам'яті та кількості одночасно запущених реплік. Навантажувальні тести можуть допомогти отримати початкову оцінку, але ці рекомендації будуть застарілими, оскільки потреби в ресурсах змінюються з часом, оскільки багато кінцевих користувачів, які обслуговують завдання, мають денний або тижневий графік навантаження, а трафік змінюється в більш тривалих часових масштабах, коли сервіс стає більш-менш популярним.

Нарешті, ресурси, необхідні для обробки певного навантаження, змінюються з появою нових функцій, оптимізації та оновлень базового програмного або апаратного стеку. Перевищення запитуваних ресурсів може призвести до низької продуктивності, якщо центральний процесор обмежений, або до завершення завдання через нестачу пам'яті (переповнення пам'яті). Ні те, ні інше не є добре.

Як наслідок, раціональний користувач свідомо переоцінює ресурси, необхідні для роботи, що призводить до нераціонального використання фізичних ресурсів. Один аналіз місячного відстеження завдань, виконаних на одному з кластерів Google, показав середнє використання пам'яті на рівні 50%, інший аналіз кластера YARN компанії Al-ibaba показав, що пікове використання пам'яті завдань ніколи не перевищувало 80%.

У відповідь на труднощі з конфігурацією ресурсів поширеною моделлю є використання горизонтального автомасштабування, яке масштабує завдання, додаючи або видаляючи репліки у відповідь на зміни трафіку кінцевого користувача або середнього завантаження процесора.

Усі основні хмарні провайдери (AWS, Azure та GCP) пропонують функцію горизонтального автомасштабування, вона також доступна в деяких хмарних проміжних програмах, таких як Kubernetes. Менш поширеним є використання вертикального автомасштабування для налаштування кількості ресурсів, доступних для кожної репліки. Ці два методи також можна комбінувати.

Autopilot - це основний «автомасштабувальник» , який Google використовує у своїй внутрішній хмарі. Він забезпечує як горизонтальне, так і вертикальне автомасштабування. Ця стаття фокусується на вертикальному масштабуванні пам'яті Autopilot, оскільки про нього повідомляється рідше.

Цілі та обмеження Автопілота випливають з інфраструктури Borg Google, і він налаштований під робоче навантаження Google.

Обчислювальна інфраструктура Google складається з багатьох кластерів, розташованих у різних географічних точках. Середній кластер має приблизно 10 000 фізичних машин і виконує багато різних типів робочих навантажень одночасно. Одна фізична машина може одночасно виконувати обчислення, що вимагають багато пам'яті та важкі для процесора пакетні обчислення, зберігати та обслуговувати запити до резидентної бази даних, а також обслуговувати латентно чутливі запити кінцевих користувачів.

Завдання складається з одного або декількох завдань. Завдання виконується на одній фізичній машині, одна машина виконує декілька завдань одночасно. Завдання -

це логічна сутність, яка відповідає сервісу з певною функціональністю (наприклад, файловій системі або сервісу автентифікації), завдання виконують файлова система або служба автентифікації), завдання виконують фактичну роботу, таку як обслуговування запитів кінцевих користувачів або запитів на доступ до файлів.

Під час такого розгортання нові завдання поступово замінюють завдання, що працюють зі старим бінарним кодом. Робочі навантаження, які виконуються, можна розділити на дві категорії: обслуговуючі та пакетні. Обслуговуючі завдання, як правило, мають на меті суворі гарантії продуктивності щодо часу відповіді на запит (наприклад, мета рівня обслуговування щодо затримки запитів або SLO ≤ 50 мс при 95%ile).

Такі жорсткі вимоги до затримок унеможливлюють прийняття будь-яких рішень щодо розподілу ресурсів у смузі пропускання, окрім рішень ядра ОС. ядром ОС, тому завданням, що обслуговуються, виділяються ресурси, які вони запитують, у явному вигляді. На противагу цьому, пакетні завдання мають на меті "швидко" завершити і вийти, але, як правило, не мають жорстких термінів завершення. Обслуговування завдань є основною рушійною силою нашої інфраструктури, в той час як серійні завдання, як правило, заповнюють вільні або тимчасово залишкову або тимчасово невикористану потужність.

Подія виходу з пам'яті (out-of-memory, OOM) завершує виконання окремого завдання. Деякі завдання є достатньо толерантними до подій виходу з пам'яті, деякі зовсім не толерантні, а деякі знаходяться посередині. Загалом, завдання складаються з більшої кількості завдань і мають менший досвід роботи з державою менше погіршення сервісу після завершення окремого завдання, а отже є більш толерантними до OOM. Деякі завдання вимагають низької, повторюваної затримки в обслуговуванні запитів, а деякі - ні. Автопілот вибирає значення за замовчуванням на основі заявленого розміру, пріоритету та класу завдання, але дозволяє користувачам змінювати їх. Borg виселяє завдання у випадках для того, щоб виконати оновлення безпеки та ОС, звільнити місце для більш пріоритетних завдань, а також для того, щоб більш ефективно розподіляти роботу на машини більш ефективно. Свідомо розподіляючи тягар забезпечення відмовостійкості сервісів між обчислювальним

кластером інфраструктурою обчислювального кластера та нашими додатками, які, як очікується, будуть виконувати додаткові завдання, щоб обійти виселення та збої в роботі обладнання. збоїв обладнання. Borg публікує очікувану максимальну частоту цих виселень, а також різницю між спостережуваною і фактичною частотою виселень.

І спостережуваною швидкістю виселень, що дає свободу експериментувати з налаштуваннями ресурсів завдання - випадковий OOM.

Щоб досягти прийнятної та передбачуваної продуктивності, завдання що виконуються на машині, мають бути ізольовані одна від одної.

Як і у випадку з Kubernetes, Borg запускає кожне завдання в окремому контейнері Linux і локальний агент встановлює обмеження на ресурси контейнера для досягнення ізоляції продуктивності за допомогою cgroups. На відміну від традиційного справедливого розподілу ресурсів на рівні ОС, це гарантує, що продуктивність завдання є узгодженою між різними виконаннями одного і того ж бінарного файлу, на різних машинах (якщо апаратне забезпечення однакове) і різних сусідів (завдання, які спільно заплановані на одній машині).

У інфраструктурі процесор та оперативна пам'ять є ключовими ресурсами для управління. Використовуючи термін "ліміт" для позначення максимально дозволену кількість кожного ресурсу, яка може бути споживатися. Оскільки Borg зазвичай розглядає завдання робочих місць як взаємозамінні копії, всі завдання зазвичай мають однакові ліміти.

Завдання виражає ліміт свого процесора у нормалізованих міліядерних процесорах, і це обмеження забезпечується стандартним механізмом ядра Linux cgroups. Якщо існує невелика суперечка (яка вимірюється загальним використанням процесора), завданням дозволяється використовувати процесор понад встановлені ліміти. Однак, як тільки виникають суперечки, обмеження застосовуються, і деякі завдання можуть бути призупинені, щоб працювати в межах своїх лімітів

3.2. Дослідження фреймворку OpenStack NEAT на базі OpenStack

Хмарні обчислення зробили революцію в галузі інформаційно-комунікаційних технологій (ІКТ), дозволивши надавати еластичні обчислювальні ресурси на вимогу за принципом "pay as you go". Однак хмарні центри обробки даних споживають величезну кількість електроенергії, що призводить до високих операційних витрат і викидів вуглекислого газу (CO2) в навколишнє середовище. За оцінками, енергоспоживання центрів обробки даних у всьому світі зросло на 56% з 2005 по 2010 рік і становило від 1,1% до 1,5% світового споживання електроенергії в 2010 році. Крім того, викиди вуглекислого газу в ІКТ-індустрії оцінюються в 2% від глобальних викидів, що еквівалентно викидам авіаційної промисловості.

Щоб вирішити проблему високого енергоспоживання, необхідно усунути неефективність і марнотратство в тому, як електроенергія постачається до обчислювальних ресурсів, і в тому, як ці ресурси використовуються для обслуговування робочих навантажень додатків. Це може бути здійснено шляхом вдосконалення фізичної інфраструктури центрів обробки даних, а також алгоритмів розподілу та управління ресурсами. Нещодавній прогрес у проектуванні центрів обробки даних призвів до значного підвищення ефективності інфраструктури. Як повідомляє проект Open Compute, центр обробки даних Facebook в Орегоні досягнув ефективності використання електроенергії а це означає, що приблизно 91% енергоспоживання дата-центру припадає на обчислювальні ресурси. Тому зараз важливо зосередитися на оптимізації розподілу та використання ресурсів для обслуговування робочих навантажень додатків.

Одним з методів покращення використання ресурсів та зменшення енергоспоживання є динамічна консолідація віртуальних машин (ВМ), що забезпечується міграцією в реальному часі - можливістю перенесення ВМ між фізичними серверами (хостами або вузлами) з близьким до нуля часом простою. Динамічна консолідація ВМ складається з двох основних процесів: міграція ВМ з недовантажених хостів для мінімізації кількості активних хостів та вивантаження ВМ

з перевантажених хостів, щоб уникнути погіршення продуктивності ВМ, що може призвести до порушення вимог QoS.

Хости, що простоюють, автоматично перемикаються в режим низького енергоспоживання, щоб усунути статичну енергію та зменшити загальне енергоспоживання. За потреби хости активуються для розміщення нових віртуальних машин або віртуальних машин, що мігрують з інших хостів. Незважаючи на те, що на тему динамічної консолідації віртуальних машин опубліковано велику кількість досліджень, існує дуже мало програмних реалізацій з відкритим вихідним кодом.

У цій роботі розглядається архітектура та реалізація OpenStack Neat – програмного фреймворку з відкритим вихідним кодом для розподіленої динамічної консолідації віртуальних машин у хмарних дата-центрах на базі платформи OpenStack.

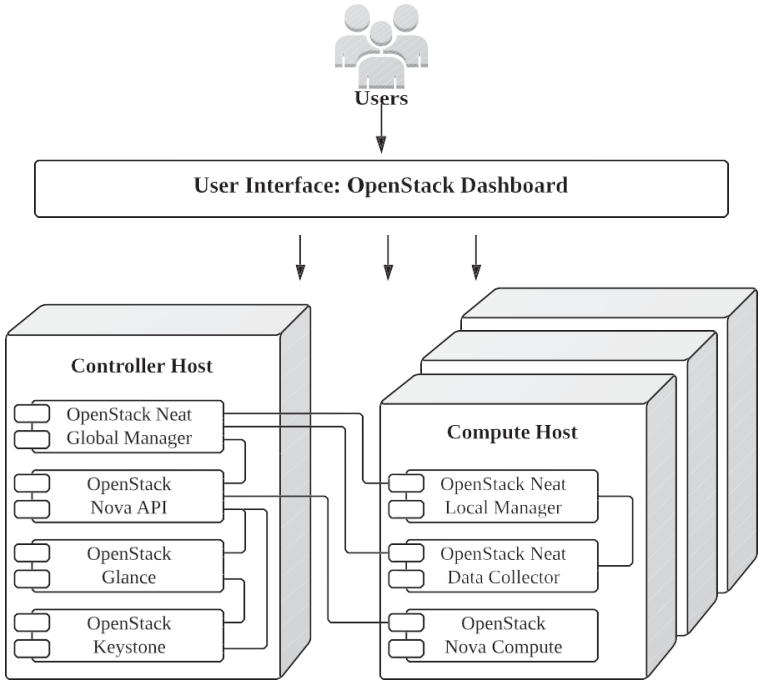

Рис. 3.1. розгортання основних сервісів OpenStack

На рисунку 3.1 зображено типове розгортання основних сервісів OpenStack, сервісів OpenStack Neat та їх взаємодію. Розгортання може включати декілька екземплярів обчислювальних і контролерних хостів. Фреймворк OpenStack Neat розроблений і реалізований як прозорий додаток до OpenStack, що означає, що

інсталяцію OpenStack не потрібно модифікувати або спеціально налаштовувати, щоб отримати вигоду від OpenStack Neat. Фреймворк працює незалежно від базової платформи OpenStack і застосовує процеси консолідації віртуальних машин шляхом виклику публічних API OpenStack. Мета фреймворку OpenStack Neat подвійна:

Перша: надання повноцінного програмного забезпечення з відкритим вихідним кодом для динамічної консолідації віртуальних машин, яке може бути застосоване до існуючих хмар OpenStack та друга: надання розширюваної програмної основи для проведення досліджень з динамічної консолідації віртуальних машин.

OpenStack Neat розроблено та реалізовано відповідно до розподіленого підходу до динамічної консолідації віртуальних машин, представленого та оціненого в наших попередніх роботах. Цільовим середовищем є інфраструктура як послуга (IaaS), наприклад, Amazon EC2, де провайдер не знає про додатки та робочі навантаження, що обслуговуються віртуальними машинами, і може лише спостерігати за ними ззовні. Називаючи цю властивість середовищ IaaS агностичністю до додатків.

Запропонований підхід до розподіленої динамічної консолідації ВМ полягає в розбитті проблеми на чотири пункти:

- прийняття рішення про те, чи вважається хост недовантаженим, і чи слід перенести з нього всі ВМ, а хост перевести в режим низького енергоспоживання;
- прийняття рішення про те, що хост перевантажений, і деякі ВМ слід перенести з нього на інші активні або реанімовані хости, щоб уникнути порушення вимог QoS;
- вибір ВМ для міграції з перевантаженого хоста;
- розміщення обраних для міграції ВМ на інших активних або реактивних хостах.

Цей підхід має дві основні переваги порівняно з традиційними повністю централізованими алгоритмами консолідації ВМ:

- перший розділення проблеми спрощує її аналітичну обробку, дозволяючи розглядати підпроблеми незалежно;

- другий підхід може бути реалізований частково розподілено, виконуючи алгоритми виявлення недовантаження/перевантаження та вибору ВМ на обчислювальних хостах, а алгоритм розміщення ВМ на хості-контролері, який за бажанням може бути реплікований.

Розподілені алгоритми консолідації ВМ забезпечують природне масштабування системи до тисяч обчислювальних вузлів, що є важливим для великих хмарних провайдерів. Наприклад, Rackspace, відомий IaaS-провайдер, наразі управляє десятками тисяч серверів. Більше того, кількість серверів постійно зростає: Rackspace збільшив загальну кількість серверів у другому кварталі 2012 року до 84 978 порівняно з 82 438 серверами наприкінці першого кварталу.

Крім того, для сприяння дослідницьким зусиллям та майбутнім досягненням в області динамічної консолідації ВМ, розробники пропонують набір тестів для оцінки та порівняння алгоритмів динамічної консолідації ВМ, що складається з OpenStack Neat як базового програмного забезпечення, трас реальних робочих навантажень від PlanetLab, метрик продуктивності та методології оцінки.

Дослідницькі роботи, пов'язані з цією статтею, можна розділити на дві категорії:

- практично реалізовані та загальнодоступні програмні системи з відкритим вихідним кодом;
- теоретичні роботи, присвячені різним підходам до динамічної консолідації віртуальних машин.

Є велика кількість досліджень, присвячених темі динамічної консолідації віртуальних машин і при цьому існує дуже мало програмних реалізацій, доступних у відкритому доступі в Інтернеті. Наскільки нам відомо, найбільш ранньою реалізацією менеджера консолідації ВМ з відкритим вихідним кодом є проект Entropy. Entropy - це менеджер консолідації ВМ для однорідних кластерів з відкритим вихідним кодом, розроблений Hermenier та інше і випущений під ліцензією Lesser General Public License.

Entropy побудовано на основі Xen і орієнтовано на дві мети, а саме підтримання конфігурації кластера, при якій усім віртуальним машинам виділяється достатньо

ресурсів та мінімізація кількості активних хостів. Для оптимізації розміщення віртуальних машин Entropy періодично застосовує двоетапний підхід. Спочатку розв'язується задача програмування з обмеженнями, щоб знайти оптимальне розміщення ВМ, яке мінімізує кількість активних хостів.

Потім розв'язується інша оптимізаційна задача, щоб знайти цільову конфігурацію кластера з мінімальною кількістю активних хостів, яка також мінімізує загальну вартість реконфігурації, яка пропорційна вартості міграції ВМ. Замість того, щоб періодично оптимізувати розміщення віртуальних машин, як це робить Entropy, OpenStack Neat виявляє умови недовантаження та перевантаження хостів і динамічно вирішує їх, що дозволяє системі мати більш тонкий контроль над станами хостів. Хоча Entropy може знайти більш оптимальне розміщення віртуальних машин, обчислюючи глобально оптимальне рішення, всі аспекти оптимізації розміщення віртуальних машин повинні обчислюватися центральним контролером, що обмежує масштабованість системи.

Феллер та інше запропонували та реалізували фреймворк для розподіленого управління віртуальними машинами для приватних хмар під назвою Snooze, який має відкритий вихідний код і випускається під ліцензією General Public License версії 2.

На додаток до функціональності, що надається існуючими платформами управління хмарами, такими як OpenStack, Eucalyptus та OpenNebula, Snooze реалізує динамічну консолідацію ВМ як одну з своїх базових функцій. Інша відмінність полягає в тому, що Snooze реалізує ієрархічне управління розподіленими ресурсами. Ієрархія управління складається з трьох рівнів: локальні контролери на кожному фізичному вузлі, менеджери груп, які керують набором локальних контролерів, і лідер групи, який динамічно обирається з набору менеджерів груп і виконує завдання глобального управління. Розподілена структура забезпечує відмовостійкість та самовідновлення, уникаючи єдиних точок відмови та автоматично обираючи нового лідера групи, якщо поточний виходить з ладу.

Snooze також інтегрує моніторинг використання ресурсів віртуальними машинами та хостами, який можна використовувати за допомогою політик консолідації віртуальних машин. Ці політики призначені для реалізації на рівні

менеджерів груп і тому можуть застосовуватися лише до підмножин хостів. Такий підхід частково вирішує проблему масштабованості консолідації ВМ ціною втрати можливості оптимізації розміщення ВМ на всіх вузлах дата-центру.

OpenStack Neat забезпечує масштабованість за рахунок розподіленого виявлення недовантаження/перевантаження та вибору ВМ, а також потенційної реплікації контролерів розміщення ВМ. На відміну від Snooze, він здатний застосовувати глобальні алгоритми розміщення ВМ для обраних міграційних ВМ з урахуванням повного набору хостів. Ще одна відмінність полягає в тому, що OpenStack Neat прозоро інтегрується з OpenStack, зрілою хмарною платформою з відкритим вихідним кодом, широко прийнятою і підтримуваною в індустрії, що забезпечує довгостроковий розвиток платформи.

3.3. Призначення, основні функції та склад алгоритму Round Robin

Алгоритм балансування навантаження завжди намагається вирішити конкретну проблему. Крім усього іншого, слід враховувати характер завдань, алгоритмічну складність, апаратну архітектуру, на якій працюватимуть алгоритми, а також необхідну стійкість до помилок. Тому потрібно знайти компроміс, щоб найкращим чином відповідати вимогам конкретної програми.

Серед цілей, для досягнення яких використовується балансування, слід виділити такі:

- ефективність алгоритмів балансування навантаження критично залежить від характеру завдань. отже, чим більше інформації про завдання доступно на момент прийняття рішення, тим більший потенціал для оптимізації;
- досконале знання часу виконання кожного із завдань дозволяє досягти оптимального розподілу навантаження. на жаль, насправді це ідеалізований випадок. знати точний час виконання кожного завдання - вкрай рідкісна ситуація;

− досягнення мінімального часу виконання запиту: передбачає забезпечення найменшої можливої затримки між початком обробки запиту (або його постановкою у чергу на обробку) та його завершенням;

− іншою особливістю завдань, критично важливим для розробки алгоритму балансування навантаження, є їх здатність розбиватися на підзадачі під час виконання.

Бажано, щоб алгоритм балансування мав наступні характеристики:

− передбачуваність: необхідно чітко розуміти, в яких ситуаціях і при якому навантаженні алгоритм буде ефективним для досягнення поставлених цілей;

− рівномірне завантаження ресурсів системи: алгоритм повинен розподіляти навантаження рівномірно між ресурсами системи, уникати перевантаження окремих компонентів та максимізувати використання ресурсів;

− масштабованість: алгоритм повинен зберігати свою ефективність при збільшенні навантаження, забезпечуючи працездатність системи навіть при зростанні обсягів роботи.

Алгоритм кругового обслуговування, відомий також як Round Robin, передбачає послідовну обробку запитів у циклі: кожен запит передається послідовно від одного сервера до іншого, поки не буде досягнутий останній сервер, а потім цикл повторюється з початку. Round-robin (RR) − це один із алгоритмів, які використовують планувальники процесів і мереж в обчислювальній техніці. Як термін є в основному використовуваним, терміни, є визначеними для всіх процесів у двох частинах і в циклічному порядку, обробляючи всі процеси без пріоритету (також відомий як циклічний виконавчий).

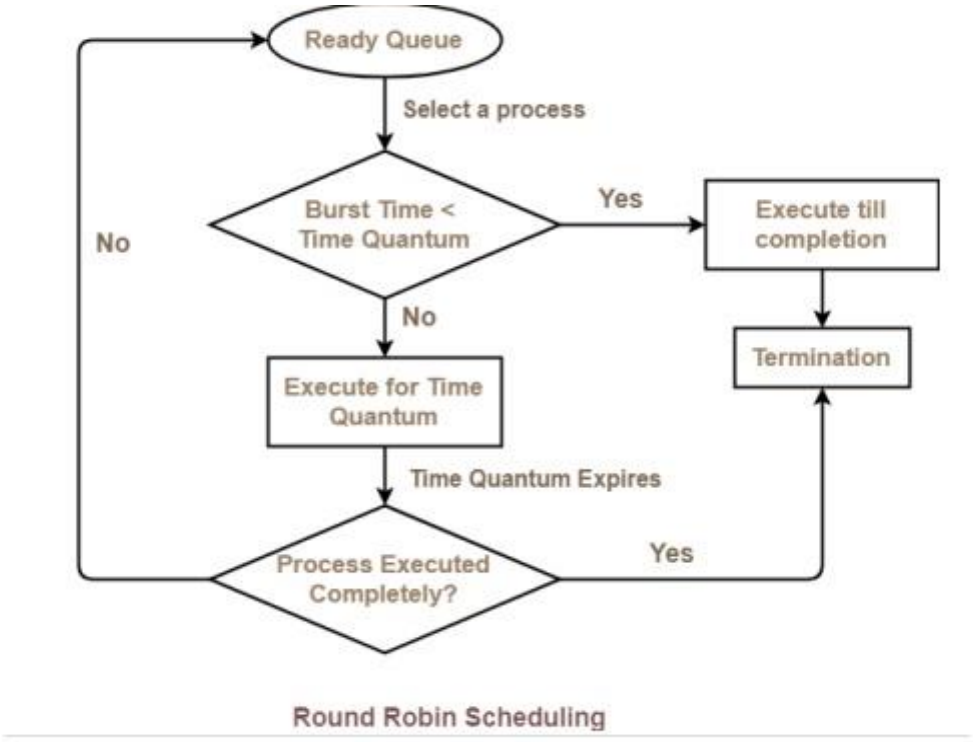

Рис. 3.2. Блок-схема алгоритму Round Robin

У централізованій бездротовій пакетній радіомережі, де багато станцій використовують один частотний канал, алгоритм планування на центральній базовій станції може зарезервувати часові проміжки для мобільних станцій по колу і забезпечити рівність. Однак, якщо використовується адаптація каналу зв'язку, передача певного обсягу даних "дорогим" користувачам займе набагато більше часу, ніж іншим, оскільки умови каналу відрізняються. Було б ефективніше зачекати з передачею, поки умови каналу не будуть покращені, або принаймні, надати пріоритет при плануванні менш дорогим користувачам. Циклічне планування цього не використовує. Вищої пропускної здатності та ефективності використання спектру системи можна досягти за допомогою залежного від каналу планування, наприклад, пропорційно-справедливого алгоритму, або планування за максимальною пропускною здатністю. Зауважте, що останнє характеризується небажаним "голодуванням планування". Цей тип планування є одним з базових алгоритмів для операційних систем в комп'ютерах, який можна реалізувати за допомогою структури даних з круговою чергою.

Найпоширенішою імплементацією цього алгоритму є, звичайно, метод балансування Round Robin DNS. Round-robin DNS — це альтернативний метод балансування навантаження, який не потребує спеціального програмного чи апаратного вузла. У цій техніці кілька IP-адрес пов’язуються з одним доменним іменем ; клієнтам надається IP у циклічному порядку. IP-адреса призначається клієнтам із коротким терміном дії, тому клієнт, швидше за все, використає іншу IP-адресу під час наступного доступу до запитуваної послуги Інтернету. Цей список приблизно може виглядати, наприклад, так:

example.com xxx.xxx.xxx.2
www.example.com xxx.xxx.xxx.3
З кожним ім'ям зі списку можна асоціювати кілька IP-адрес
example.com xxx.xxx.xxx.2
www.example.com xxx.xxx.xxx.3
www.example.com xxx.xxx.xxx.4
www.example.com xxx.xxx.xxx.5
www.example.com xxx.xxx.xxx.6

DNS-сервер послідовно проходить через всі записи в таблиці і при кожному новому запиті повертає наступну IP-адресу в послідовності. Наприклад, на перший запит повертається IP-адреса xxx.xxx.xxx.2, на другий запит - xxx.xxx.xxx.3, і так далі. Це призводить до того, що всі сервери в кластері отримують однакову кількість запитів.

Незважаючи на те, що циклічний DNS його легко реалізувати, він має низку недоліків, таких як ті, що виникають через кешування записів у самій ієрархії DNS, а також кешування та повторне використання адрес на стороні клієнта, поєднанням яких може бути важко керувати. Не варто покладатися лише на доступність служби Round-robin DNS. Якщо служба за однією з адрес у списку дає збій, DNS продовжить передавати цю адресу, і клієнти все одно намагатимуться отримати доступ до непрацюючої служби.

Round-robin DNS може бути не найкращим вибором для балансування навантаження сам по собі, оскільки він просто змінює порядок записів адрес щоразу, коли запитується сервер імен. Оскільки він не враховує час транзакції, навантаження на сервер і перевантаження мережі, він найкраще працює для служб із великою кількістю рівномірно розподілених підключень до серверів еквівалентної потужності. В іншому випадку він просто розподіляє навантаження.

Існують методи подолання таких обмежень. Наприклад, модифіковані DNS-сервери (такі як lbnamed) можуть регулярно опитувати дзеркальні сервери щодо доступності та коефіцієнта завантаження. Якщо сервер не відповідає, як потрібно, його можна тимчасово видалити з пулу DNS, доки він не повідомить, що знову працює в межах специфікацій.

Також, коли використовується алгоритм Round Robin для балансування навантаження, не приділяється уваги завантаженості окремих серверів у кластері. Припустімо, що один з вузлів насичений на 100%, тоді як інші вузли мають лише 10-15% завантаженості. Алгоритм Round Robin не враховує можливість такої ситуації, тому перевантажений вузол все одно отримує запити. В такому випадку немає мови про справедливість, ефективність та передбачуваність.

У зв'язку з описаними вище обставинами, область застосування алгоритму Round Robin має свої обмеження.

3.4 Розробка методичних рекомендацій щодо забезпечення енергоефективності обробки навантажень в інформаційній мережі

На підставі проведених досліджень, сформульовано основні методичні рекомендації. При виборі конкретного алгоритму балансування слід керуватися двома основними факторами. По-перше, враховувати специфіку конкретного проекту, а по-друге, орієнтуватися на поставлені цілі, які ми плануємо досягти.

Серед цілей, для досягнення яких використовується балансування, особливу увагу слід звернути наступним аспектам:

Справедливість: гарантувати, що кожен запит отримує необхідні системні ресурси для обробки, і уникнути ситуацій, коли один запит обробляється, а інші залишаються у черзі.

Ефективність: максимальне використання ресурсів серверів, що обробляють запити. Варто уникати ситуацій, коли деякий сервер простоює, не отримуючи запитів на обробку. Слід зазначити, що досягнення абсолютної ефективності не завжди можливе у реальних умовах.

Скорочення часу виконання запиту: забезпечення мінімального часу між постановкою запиту у чергу на обробку і його завершенням.

Скорочення часу відгуку: мінімізація часу, необхідного для надання відповіді користувачеві.

Додатково, важливо, щоб алгоритм балансування мав наступні властивості:

Передбачуваність: чітке розуміння умов і завантажень, при яких алгоритм буде ефективним для досягнення поставлених цілей.

Рівномірне завантаження ресурсів системи.

Масштабованість: здатність алгоритму зберігати працездатність при збільшенні навантаження.

Враховуючи ці рекомендації, можна вибрати найбільш відповідний алгоритм балансування для конкретного проекту.

На основі досліджень рекомендовано OpenStack, тому що OpenStack можна налаштувати на використання спеціальних алгоритмів консолідації віртуальних машин і прозоро інтегрувати з існуючими розгортаннями OpenStack без необхідності змінювати їхню конфігурацію. Окрім того інструментарій із надійною репутацією на ринку хмарних рішень, який регулярно оновлюється разом із розвитком технологій. Однак якість рішення на базі цієї платформи залежить від розробників самої хмари, тому пропозиції конкретних компаній є надійнішим показником, ніж факт використання OpenStack.

Проте OpenStack легко модифікувати, тому його часто використовують компанії, які орієнтуються на вирішення конкретних завдань клієнтів.

Загальна недоліком у всіх досліджених алгоритмах щодо підвищення енергоефективності обробки в інформаційній мережі полягає в тому, що кожен з них враховує лише окремі аспекти та показники ефективності цього процесу. Це вказує на необхідність систематизації та формалізації процесу обробки навантаження в мережі. Таким чином, використання будь-якого окремого рішення не є доцільним. Проте, OpenStack може використовувати додаткові модулі та фреймворки, що дозволить розширити його можливості та підвищити ефективність обробки навантаження в мережі.

ВИСНОВОК ДО РОЗДІЛУ 3

Під час аналізу існуючих методів для підвищення енергоефективності розподіленої обробки навантаження, були виявлені певні недоліки. Статичні підходи не враховують динамічні зміни інтенсивності навантаження, а динамічні підходи на рівні апаратного забезпечення мають складну і дорогу реалізацію. Динамічні підходи на рівні програмного забезпечення, що стосуються масштабування обчислювальних ресурсів, не враховують доступність системи, можуть негативно впливати на продуктивність при зміні інтенсивності навантаження і не використовують індивідуальні характеристики енергоспоживання обчислювальних вузлів, що призводить до неефективного використання ресурсів.

Алгоритм планування навантаження Round-robin був визнаний одним з методів енергоефективного розподілу навантаження. Його основною перевагою є незалежність від протоколу високого рівня та низька вартість реалізації. Проте, ефективність цього підходу значно обмежена через його обмежену сферу застосування і відсутність урахування індивідуальних характеристик енергоспоживання та продуктивності обчислювальних вузлів. Загалом, існуючі підходи до підвищення енергоефективності обробки навантаження вирішують лише окремі аспекти та показники ефективності, що підкреслює необхідність систематизації та формалізації цього процесу в інформаційних мережах.

Запропоновані рекомендації в роботі можуть бути використані для поліпшення енергоефективності існуючих дата-центрів або кластерів серверів для обробки даних. Зменшення споживання електроенергії матиме позитивний вплив як на економічні аспекти (зниження операційних витрат на функціонування серверів), так і на екологічні (зменшення викидів при виробництві електроенергії). При цьому якість обробки інформації не зазнає погіршення, що уникне економічних збитків для компаній.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження, спрямоване на вирішення актуальної науково-практичної проблеми, а саме підвищення енергоефективності обробки навантаження в інформаційній мережі. Отримані результати дослідження мають теоретичний характер, а саме:

Аналіз специфікацій та рекомендацій Міжнародної спілки електрозв'язку та Європейського інституту телекомунікаційних стандартів дозволив визначити основні завдання, які впливають на обчислювальне навантаження в сучасних інформаційно-комунікаційних мережах (ІКМ). Шляхом аналізу вимог, пов'язаних з обробкою такого навантаження, були визначені ключові показники ефективності процесу обробки навантаження в ІКМ: енергоефективність (EZ), продуктивність (CZ) обслуговування та коефіцієнт готовності системи ($p$avail), що вказують на її доступність. З використанням цих показників був сформульований критерій ефективності системи (Kopt).

Був проведений аналіз наявних підходів до підвищення ефективності обробки навантаження в інформаційній мережі (з урахуванням енергоефективності, продуктивності та доступності системи). Виявлено, що існуючі підходи не враховують всі три зазначенні показники одночасно, що має критичне значення для обробки навантаження в ІКМ з урахуванням необхідності забезпечення якості надання послуг. Крім того, існуючі підходи не використовують індивідуальні характеристики енергоспоживання обчислювальних вузлів, що призводить до неоптимального використання обчислювальних ресурсів. На основі проведених досліджень, слід виділити програмний комплекс OpenStack.

OpenStack можна налаштувати на використання спеціальних алгоритмів консолідації віртуальних машин і прозоро інтегрувати з існуючими розгортаннями OpenStack без необхідності змінювати їхню конфігурацію. Окрім того інструментарій із надійною репутацією на ринку хмарних рішень, який регулярно оновлюється разом із розвитком технологій. Однак якість рішення на базі цієї платформи залежить від

розробників самої хмари, тому пропозиції конкретних компаній є надійнішим показником, ніж факт використання OpenStack.

Проте OpenStack легко модифікувати, тому його часто використовують компанії, які орієнтуються на вирішення конкретних завдань клієнтів.

При аналізі питання підвищення енергоефективної обробки в інформаційній мережі було виявлено загальний недолік у всіх досліджених алгоритмах. Кожен з цих алгоритмів вирішує задачу з урахуванням лише обмеженої кількості аспектів процесу та показників ефективності. Це підкреслило необхідність систематизації та формалізації процесу обробки навантаження в інформаційній мережі.

На основі результатів дослідження, можна сказати що експлуатувати одне з запропонованих рішень не є доцільним і краще використовувати додаткові модулі, або фреймворки, наприклад, як у OpenStack, їх кількість значно покращує рейтинг продукту на світовому ринку, а отже і клієнтів по всій планеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Horizontal and Vertical Scalability of Machine Learning Methods | Biletskyy | PROBLEMS IN PROGRAMMING. Home Page. URL: https://doi.org/10.15407/pp2019.02.069 (date of access: 07.06.2023)

2. ISO - International Organization for Standardization. ISO. URL: https://www.iso.org/home.html (date of access: 1.06.2023).

3. System for Cross-domain Identity Management: Protocol. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/draft-ietf-scim-api-19 (date of access: 16.06.2023).

4. Інформаційні технології. Stud. URL: https://stud.com.ua/35734/informatika/informatsiyni_tehnologiyi (date of access: 07.06.2023).

5. DDoS-атака | ESET. ESET. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/distributed-denial-of-service/ (дата звернення: 03.06.2023).

6. What Is Quality of Service (QoS)? - Huawei. URL: https://support.huawei.com/enterprise/en/doc/EDOC1100086518 (date of access: 31.05.2023).

7. Beloglazov, A., & Buyya, R. (2015). OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. Concurrency and Computation: Practice and Experience, 27(5), pp. 1300-1343. [Електронний ресурс]: https://doi.org/10.1002/cpe.3314. (date of access: 05.06.2023).

8. What Is SASE - Secure Access Service Edge?. Cisco. URL: https://www.cisco.com/c/en/us/products/security/what-is-sase-secure-access-service-edge.html (date of access: 10.06.2023).

9. Sudarsono, G. A., Nugroho, E. P., & Putra, R. R. J. (2019, November). Scheduling the cluster server node shutdown based on the hierarchical and k-means clustering combinations. In Journal of Physics: Conference Series (Vol. 1280, No. 3, p.

032039). IOP Publishing. [Електронний ресурс]: https://doi.org/10.1088/1742-6596/1280/3/032039. (date of access: 04.06.2023).

10. Gandhi, A., Gupta, V., Harchol-Balter, M., & Kozuch, M. A. (2010). Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation, 67(11), pp. 1155-1171.

11. Overview of Enhanced Intel SpeedStep® Technology for Intel®... Intel. URL: https://www.intel.com/content/www/us/en/support/articles/000007073/processors.html (date of access: 05.06.2023).

12. Hosseinimotlagh, S., Khunjush, F., & Hosseinimotlagh, S. (2014, February). A cooperative two-tier energy-aware scheduling for real-time tasks in computing clouds. In 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (pp. 178-182). IEEE.

13. IBM Spectrum LSF 10.1.0 Document, (2022), [Електронний ресурс] available at: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=jobs-backfill-scheduling (date of access: 06.06.2023).

14. Round-Robin Scheduling. Google Books. URL: https://web.archive.org/web/20200718160256/https://books.google.com.eg/books?id=5Z1cbwAACAAJ&dq=Round-robin+scheduling&hl=en&sa=X& (date of access: 04.06.2023).

15. Autopilot overview | Google Kubernetes Engine (GKE) | Google Cloud. Google Cloud. URL: https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview (date of access: 30.05.2023).

16. Network Functions Virtualisation (NFV). ETSI. URL: https://www.etsi.org/technologies/nfv (date of access: 28.05.2023)

17. View on 5G Architecture: 5G PPP Architecture Working Group, (2019), [Електронний ресурс]: https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture- White-Paper_v3.0_PublicConsultation.pdf (date of access: 05.06.2023).

18. Європейський інститут телекомунікаційних стандартів. ETSI Standard GR NFV 003, (2020), “Network Functions Virtualisation (NFV); Terminology for Main

Concepts in NFV",[Електронний ресурс]: https://www.etsi.org/deliver/etsi_gr/-NFV/001_099/003/01.05.01_60/gr_NFV003v0105 01p.pdf (date of access: 23.05.2023).

20. Міжнародна спілка електрозв’язку. ITU-T Y.3300 - Framework of software-defined networking, (2014), [Електронний ресурс] available at: https://www.itu.int/rec/T-REC-Y.3300-201406-I/en. (date of access: 24.05.2023).

21. Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. Security and communication networks, 9(18), pp. 4603-4633. [Електронний ресурс]: https://doi.org/10.1002/sec.1737. (date of access: 06.06.2023).