НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«На правах рукопису»
УДК 621.391+004.7
Н34

«До захисту допущено»
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Експериментальне дослідження енергоефективності обробки даних в розподіленому дата центрі»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ТС-71мн
Прокопець Володимир Андрійович _____

Керівник:
завідувач кафедрою
Інформаційно-телекомунікаційних мереж ІТС,
професор, докт.техн.наук
Глоба Л.С. _____

Рецензент:
проф. каф. телекомунікацій,
д.т.н., проф., Романов О.І. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Л.О. Уривський

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

_____
(прізвище, ім'я, по батькові)

1. Тема дисертації _____
_____

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

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження _____
_____

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) _____
_____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____

_____

_____

_____

_____

_____

6. Перелік графічного (ілюстративного) матеріалу _____

_____

_____

7. Орієнтовний перелік публікацій _____

_____

_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

_____

РЕФЕРАТ

Робота містить 96 сторінок, 31 рисунок та 12 таблиць. Було використано 52 джерела.

Мета дослідження – підвищити енергоефективність та продуктивність обробки запитів користувача в гетерогенному ЦОД (центрі обробки даних) оператора зв’язку.

Завдання:

1. Дослідити технологію NFV (Network Function Virtualization), а також особливості її використання в мережі LTE. Визначити підхід щодо розрахунку обсягу навантаження, який надходитиме на ЦОД мобільних операторів зв’язку України.

2. Дослідити існуючі алгоритми балансування навантаження в серверному кластері, визначити їх переваги та недоліки.

3. Розробити удосконалений алгоритм балансування навантаження, який покращить енергоефективність обробки даних на основі даних, отриманих за запропонованим розрахунком обсягу навантаження.

4. Створити імітаційну модель серверного кластеру з функцією балансування навантаження в середовищі Matlab, базуючись на отриманих даних. Після проведеної апробації моделі дослідити ефективність запропонованого алгоритму на прикладі дата-центру з середнім розміром (100 вузлів).

5. Розробити крос-платформне ПЗ (програмне забезпечення), яке виконуватиме функції балансування навантаження. ПЗ має містити можливості автоматизованого формування логічних кластерів серед наявних фізичних серверів, які знаходяться як в одній фізичній мережі, так і в різних, а також підтримки функцій балансування навантаження серед усіх вузлів логічного кластера.

6. Провести апробацію розробленої імітаційної моделі.

Об’єкт дослідження: процес балансування навантаження в розподілених дата-центрах (серверних кластерах) операторів мобільного зв’язку.

Предмет дослідження: підходи щодо покращення ефективності гетерогенної територіально розподіленої віртуалізованої інфраструктури оператора мобільного зв’язку.

Методи дослідження:

1. Теоретичне дослідження – математичні методи системного аналізу, загальнодоступні специфікації технологій, наукові праці, статистики та прогнози операторів стільникового зв’язку;
2. Імітаційне моделювання - програма математичного моделювання Matlab;
3. Апробація на обладнанні - серверний кластер Технічного університету Дрездена для перевірки адекватності моделі алгоритмів балансування навантаження.

Новизна роботи:

Запропонований підхід до енергоефективного планування завдань складається, на відміну від відомих, з двох основних етапів:

1. Етап попередньої атестації;
2. Етап планування завдань.

Етап попередньої атестації проводиться у серверному кластері періодично, при його налаштуванні. Індивідуальне визначення функцій $P = f(CPU)$ для кожного вузла обробки та їх подальше використання у процесі планування завдань є ключовою особливістю запропонованого підходу.

Другий етап передбачає на відміну від відомих технічних рішень вирішення оптимізаційного завдання за критеріями енергоефективності та продуктивності обробки завдань з використанням індивідуально визначених

енергетичних моделей вузлів. Результатом цього процесу є розміщення кожного поточного завдання для обробки на сервер із оптимальними параметрами.

Запропонований підхід представлено у вигляді алгоритму планувальника завдань.

Обгрунтованість і достовірність наукових положень, висновків і рекомендацій:

1. Проведено аналіз існуючих підходів до підвищення енергоефективності обчислень у центрі обробки даних, який показав наявність низки методів, що мають на меті зниження енергоспоживання під час обробки даних, але не враховують одночасно параметр продуктивності обчислень.

2. Виділено клас методів підвищення енергоефективності обчислень, а саме енергоефективне планування завдань, у рамках якого доцільним є внесення змін з метою покращення показників енергоефективності та продуктивності обчислень.

3. Запропоновано підхід до підвищення енергоефективності обчислень для серверного кластера як інформаційно-телекомунікаційної одиниці інфраструктури ЦОД оператора зв’язку або провайдера онлайн-послуг, який відрізняється одночасним врахуванням параметрів енергоефективності та продуктивності при розподілі завдань.

4. Розроблено алгоритм підвищення енергоефективності обчислень на основі запропонованого підходу, який складається з етапу попередньої індивідуальної атестації серверного кластера та етапу енергоефективного розподілу завдань. Врахування індивідуальних математичних залежностей енергоспоживання серверів з урахуванням їх завантаженості є основною відмінною рисою запропонованого підходу.

5. Перевірено роботу запропонованого підходу експериментально та шляхом імітаційного моделювання. Проведено аналіз отриманих результатів, та визначено, що підхід проявляє більшу ефективність – до 25%

за показниками продуктивності та енергоефективності при оптимальних вагових коефіцієнтах – для великих гетерогенних кластерів.

Наукове значення роботи:

Запропоновані в дисертації підходи щодо удосконалення процесу балансування навантаження в розподілених дата-центрах (серверних кластерах) операторів мобільного зв'язку розширюють спектр можливостей балансування навантаження в серверному кластері, покращуючи енергоефективність обчислень, які в них виконуються, при збереженні поточного рівня QoS під час надання послуг зв'язку.

Практичне значення отриманих результатів:

Запропоновані в роботі підходи можуть бути використані для покращення енергоефективності існуючих масштабних дата-центрів мобільних операторів зв'язку. Зекономлена електроенергія матиме як економічний (зменшення операційних витрат на функціонування дата-центрів) так і екологічний (зменшення викидів в атмосферу при виробництві електроенергії) ефект. При цьому якісні показники обробки інформації не деградуватимуть і відповідно не нестимуть економічних збитків операторам зв'язку.

Особистий внесок магістранта:

Була виконана розробка імітаційної моделі з використанням середовища MATLAB. Проведена апробація моделі на реальному обладнанні серверного кластера (якісний і кількісний склад якого був відтворений у моделі MATLAB). Результати апробації вказують, що налаштування моделі було виконане правильно. Створене програмне забезпечення, яке реалізує підхід щодо балансування навантаження, яке дозволить покращити показники енергоспоживання та продуктивності в дата-центрі мобільних операторів зв'язку.

Апробація результатів дисертації:

Результаті досліджень, включених до магістерської дисертації були висвітлені на наступних конференціях:

- CADSM, February 26 – March 2, 2019, Polyana-Svalyava (Zakarpattya), Ukraine
- ACS 2018, Międzyzdroje, Poland, September 24-26, 2018
- Black Sea Com 2018, Batumi, Georgia
- Перспективи розвитку інформаційно-телекомунікаційних технологій та систем 2018, Київ, Україна.
- Проблеми телекомунікацій 2018, Київ, Україна
- UkrMiCo'2017, 11-15 September 2017, Odesa, Ukraine

Публікації:

1. "QoS and Energy Efficiency Improving in Virtualized Mobile Network EPC based on Load Balancing", Larysa Globa, Nataliia Gvozdetska, Volodymyr Prokopets, Oleksandr Stryzhak //ACS 2018, Międzyzdroje, Poland, September 24-26, 2018
2. "Prognostic-Reactive NFV Resource Allocation Method for Implementation in Virtualized Mobile Network EPC of Ukraine'', Larysa S. Globa, Volodymyr A. Prokopets, Nataliia A. Gvozdetska//Black Sea Com 2018, Batumi, Georgia
3. "Експериментальне дослідження енергоефективності обробки даних в розподіленому дата центрі", Прокопець В.А. // Перспективи розвитку інформаційно-телекомунікаційних технологій та систем 2018, Київ, Україна.
4. "Експериментальне дослідження енергоефективності обробки даних в розподіленому дата центрі", Прокопець В.А., Глоба Л.С. // Проблеми телекомунікацій 2018, Київ, Україна
5. "Power Consumption and Performance Balance (PCPB) scheduling algorithm for computer cluster", Alexander Schill, Larysa Globa, Oleksandr Stepurin, Nataliia Gvozdetska, Volodymyr Prokopets// UkrMiCo'2017, 11-15 September 2017, Odesa, Ukraine
6. "Energy-efficient Backfill-based Scheduling Approach for SLURM Resource Manager" Larysa Globa, Nataliia Gvozdetska and Volodymyr

Prokopets// CADSM, February 26 – March 2, 2019, Polyana-Svalyava (Zakarpattya), Ukraine

7. Наукоємні технології оптимізації та керування в інфокомунікаційних мережах : монографія / Під загальною редакцією В.М. Безрука, Л.С. Глоби, О.Є Стрижака. – К.: Інститут обдарованої дитини НАПН України, 2019. – 194 с. ISBN 978-617-7734-02-3

8. Наукоємні технології оптимізації та керування в інфокомунікаційних мережах : монографія / Під загальною редакцією В.М. Безрука, Л.С. Глоби, О.Є Стрижака. – К.: Інститут обдарованої дитини НАПН України, 2019. – 194 с. ISBN 978-617-7734-02-3

Ключові слова: балансування навантаження, обробка інформації в дата-центрах оператора зв’язку, віртуалізація мережевих функцій, енергоефективність.

ABSTRACT

The work contains 96 pages, 31 figures and 12 tables, 52 sources have been used.

The purpose of the study is to increase the energy efficiency and productivity of processing user requests in a heterogeneous data center (data center) of the telecommunications operator.

Task:

1. Explore NFV (Network Function Virtualization) technology for its use in the LTE network. Determine the approach to calculating the load that will be delivered to the data centers of mobile operators of Ukraine.
2. Examine the existing load balancing algorithms in the server cluster, identify their advantages and disadvantages.
3. Develop an improved load balancing algorithm that will improve the energy efficiency of data processing based on the data obtained from the proposed calculation of the load.
4. Create a simulation model of the server cluster with load balancing function in the Matlab environment based on the data received. After testing the model, examine the effectiveness of the proposed algorithm on an example of a data center with an average size (100 nodes).
5. Develop cross-platform software (software) that will perform load balancing functions. The software should include the possibility of automated logical clustering among existing physical servers located both in the same physical network and in different, as well as support for load balancing functions among all nodes in the logical cluster.
6. Conduct the testing of the developed simulation model.

Object of research: the process of balancing the load in distributed data centers (server clusters) of mobile operators.

Subject of research: approaches to improving the efficiency of the heterogeneous, territorially distributed virtualized infrastructure of the mobile communications operator.

Research methods:

1. Theoretical research - mathematical methods of system analysis, publicly available specifications of technologies, scientific works, statistics and forecasts of cellular operators;
2. Imation simulation - a program of mathematical modeling Matlab;
3. Test on the equipment - a server cluster of the Technical University of Dresden to verify the adequacy of the model load balancing algorithms.

Novelty of work:

The proposed approach to energy efficient planning of tasks is, unlike the known, from two main stages:

1. The stage of preliminary attestation;
2. Task planning stage.

The stage of preliminary attestation is performed in the server cluster from time to time, when it is configured. The individual definition of the functions $P = f(CPU)$ for each processing node and their subsequent use in the process of task planning is a key feature of the proposed approach.

The second stage involves, in contrast to the known technical solutions, the solution of the optimization task by the criteria of energy efficiency and processing of tasks using individually determined energy models of nodes. The result of this process is to place each of the current tasks for processing on the server with optimal parameters.

The proposed approach is presented as a task scheduler algorithm.

The validity and reliability of scientific provisions, conclusions and recommendations:

1. An analysis of existing approaches to improving the energy efficiency of computations in the data center has been carried out, which has shown a number of methods aimed at reducing energy consumption during data processing, but do not take into account simultaneously the parameter of computing productivity.

2. A class of methods for increasing energy efficiency of calculations, namely, energy-efficient planning of tasks, in which it is expedient to make changes in order to improve energy efficiency and computational efficiency indicators, is selected.

3. An approach to increasing the energy efficiency of computing for a server cluster as an information and telecommunication unit of a data center infrastructure of a telecommunication operator or an online service provider is proposed, which is different at the same time taking into account the parameters of energy efficiency and productivity when assigning tasks.

4. An algorithm for increasing the energy efficiency of the calculations based on the proposed approach, which consists of the stage of preliminary individual attestation of the server cluster and the stage of energy efficient allocation of tasks, is developed. The consideration of the individual mathematical dependencies of the power consumption of servers, taking into account their load capacity, is the main distinguishing feature of the proposed approach.

5. The work of the proposed approach is tested experimentally and by simulation modeling. The analysis of the obtained results is carried out, and it is determined that the approach is more effective - up to 25% for performance and energy efficiency indicators at optimal weight ratios - for large heterogeneous clusters.

The scientific significance of work:

The approaches proposed in the dissertation to improve the load balancing process in distributed data centers (server clusters) of mobile operators expands the range of load balancing capabilities in the server cluster by improving the energy efficiency of the calculations that are performed on them while maintaining the current QoS level during communication service provision.

The practical value of the results obtained:

The approaches proposed in the work can be used to improve the energy efficiency of existing large-scale data centers of mobile operators. The saved energy will have both an economic effect (reducing operating costs for the functioning of the data centers) and the ecological (reduction of atmospheric emissions in the production of electricity) effect. At the same time, the quality indicators of information processing will not degrade and accordingly will not incur economic losses to communication operators.

Master student's personal contribution:

The simulation model was developed using the MATLAB environment. We tested the model on the real equipment of the server cluster (qualitative and quantitative composition of which was reproduced in the MATLAB model). Approval results indicate that the model configuration was performed correctly. Software has been developed that implements a load balancing approach that will improve the energy and performance in the data center of mobile operators.

Approbation of the results of the dissertation:

Results of studies included in the master's thesis highlighted in the following conferences:

- CADSM, February 26 - March 2, 2019, Polyana-Svalyava (Zakarpattya), Ukraine
- ACS 2018, Międzyzdroje, Poland, September 24-26, 2018
- Black Sea Com 2018, Batumi, Georgia
- Prospects for Information and Telecommunication Technologies and Systems Development 2018, Kyiv, Ukraine.
- Problems of telecommunications 2018, Kiev, Ukraine
- UkrMiCo'2017, 11-15 September 2017, Odessa, Ukraine

Publications:

1. "QoS and Energy Efficiency Improvement in Virtualized Mobile Network EPC based on Load Balancing", Larysa Globa, Natalia Gvozdetska,

Volodymyr Prokopets, Oleksandr Stryzhak // ACS 2018, Międzyzdroje, Poland, September 24-26, 2018

2. "Prognostic-Reactive NFV Resource Allocation Method for Implementation in Virtualized Mobile Network EPC of Ukraine", Larysa S. Globa, Volodymyr A. Prokopets, Natalia A. Gvozdetska // Black Sea Com 2018, Batumi, Georgia

3. "Experimental study of energy efficiency of data processing in distributed time center", Prokopets VA // Prospects for Information and Telecommunication Technologies and Systems Development 2018, Kyiv, Ukraine.

4. "Experimental study of energy efficiency of data processing in distributed time center", Prokopets VA, Globa L.S. // Problems of telecommunications 2018, Kyiv, Ukraine

5. "Power Consumption and Performance Balance (PCPB) scheduling algorithm for a computer cluster", Alexander Schill, Larysa Globa, Oleksandr Stepurin, Natalia Gvozdetska, Volodymyr Prokopets // UkrMiCo'2017, 11-15 September 2017, Odessa, Ukraine

6. "Energy-efficient Backfill-Based Scheduling Approach for SLURM Resource Manager" by Larysa Globa, Natalia Gvozdetska and Volodymyr Prokopets // CADSM, February 26 - March 2, 2019, Polyana-Svalyava (Zakarpattya), Ukraine

7. Knowledge-based technologies of optimization and control in infocommunication networks: monograph / Under the general editorship of VM Bezruka, L.S. Globe, O. Y. Stryzhak. - K .: Gifted Child Institute of National Academy of Sciences of Ukraine, 2019. - 194 p. ISBN 978-617-7734-02-3

8. Knowledge-based technologies of optimization and control in infocommunication networks: monograph / Under the general editorship V.M. Bezruka, L.S. Globe, O. Y. Stryzhak. - K .: Gifted Child Institute of National Academy of Sciences of Ukraine, 2019. - 194 p. ISBN 978-617-7734-02-*3*

Keywords: load balancing, information processing in the data center of the telecommunication operator, virtualization of network functions, energy efficiency.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CAPEX	Capital Expenditures	Капітальні витрати
EPC	Evolved Packer Core	Покращене пакетне ядро
HSS	Home Subscriber Server	Сервер домашніх абонентів
LTE	Long Term Evolution	Довгострокова еволюція
MIMO	Multiple Input Multiple Output	Багато входів багато виходів
MME	Mobility Management Entuty	Вузол керування мобільністю
MNP	Mobile Number Portability	Переносимість мобільних номерів
NAT	Network Address Translation	Перетворення мережевих адрес
Nb-IoT	Narrow-band Internet of Things	Вузькосмуговий інтернет речей
NFV	Network Function Virtualization	Віртуалізація мережевих функцій
OFDM	Otrhogonal Frequency Division Multiplexing	Мультиплексування з ортогональним частотним розподілом каналів
OPEX	Operational Expenditures	Операційні витрати
PCRF	Policy Control and Charging Rules Function	Вузол виставлення рахунків абонентам за надані послуги зв’язку
PDN	Packet Data Network	Мережа пакетних даних
PGW	PDN Gateway	Шлюз до мереж інших операторів
SDN	Software Defined Networks	Програмно визначені мережі
SGW	Serving Gateway	Шлюз обслуговування
UMTS	Universal Mobile Telecommunication system	Універсальна мобільна телекомунікаційна система
vEPC	Virtualized EPC	Віртуалізована EPC
VR	Virtual Reality	Віртуальна реальність
ПЗ		Програмне забезпечення
ЦОД		Центр Обробки Даних

ВСТУП

Обгрунтування актуальності обраної теми дослідження

З огляду на значне зростання мобільного трафіку даних, з одного боку, а також кількості та складності послуг, з іншого боку, постачальники послуг, зокрема оператори мобільного зв'язку, все більше віртуалізують частини своєї мережі за допомогою технологій NFV, хмарних обчислень для побудови економічно ефективних та еластичних мобільних мереж та їх застосування у якості хмарних сервісів [1].

Динамічно змінювана архітектура мережі вимагає застосування контролю показників функціонування мережі в так званому «end-to-end» середовищі. Технології віртуалізації та динамічної реконфігурації мережі дозволяють знизити витрати, підвищити ефективність, мають значний вплив на забезпечення продуктивності всієї мережі та додають функціональність в її програмне забезпечення.

Сучасні телекомунікаційні системи складаються з великої кількості вузькоспеціалізованого фірмового обладнання, яке, найчастіше, може виконувати невеликий обсяг конкретних операцій, наприклад, забезпечення NAT (Network Address Translation), функцій обмеження швидкості доступу, здійснення «батьківського» контролю та фільтрацію вмісту, firewall, тощо. Такий розподіл функцій мережевої апаратури спричинений декількома факторами [2]:

1. Кожен постачальник мережевого обладнання спеціалізується на деяких функціях, обмежених визначеною ділянкою мережевої топології (наприклад, виступає виробником обладнання для мережі доступу, або ядра мережі). Зазвичай, більшість вендорів не має задовільних технічних рішень для інших частин мережевої топології.

2. Практично будь-яке вузькоспеціалізоване обладнання, задля якомога кращого виконання своїх функцій має складну і специфічну

структуру, яка не достатньо уніфікована з іншим обладнанням, навіть того ж самого виробника.

Ці фактори спричинюють необхідність підтримки гетерогенних мультивендорних мереж операторами зв'язку, які експлуатують подібні мережі. При цьому, запуск будь-якої нової послуги вимагає значних капітальних затрат (CAPEX - Capital Expenditures) для закупки нового комплекту вузькоспеціалізованого обладнання, вартість якого, як правило, значно перевищує вартість серверів загального призначення. Крім того, додатково необхідно придбати високовартісне програмне забезпечення (ПЗ) та вкладати кошти в переналаштування існуючих мережевих елементів у відповідності до нової топології. Разом з тим потрібно збільшувати операційні витрати (OPEX - Operational Expenditures) у вигляді плати за оренду додаткових площ для обладнання, додаткове енергоживлення, логістику, монтаж та пусконалагодження.

Існуюча модель розвитку, яка передбачає, що мережа складається з вузькоспеціалізованих пристроїв, та в якій впровадження нових послуг неможливе без введення в експлуатацію нового обладнання, не є оптимальною. Високі капітальні та операційні витрати спричинені затратним за часом проектуванням архітектури мережі. Оператор в майбутньому не матиме можливості вводити в експлуатацію нові послуги так швидко, як того вимагатиме ринок. Мережа має бути динамічною, сприяти як впровадженню нових сервісів та послуг, так і швидкій деактивації неактуальних послуг без втрат як для оператора, так і для користувача.

Функції, які виконують мережеві пристрої, значно різняться своїми характеристиками і можливостями їх віртуалізації. Частина з них, наприклад, переміщення мережевих пакетів з одної точки в іншу, застосовує фізичні мережеві пристрої – комутатори або маршрутизатори, їх неможливо відокремити від фізичної апаратури. Деякі інші функції, з погляду на сучасний розвиток інформаційних та комунікаційних технологій, навіть доцільно перенести в хмару.

Віртуалізація мережевих функцій (англ. Network Functions Virtualization, NFV) – це концепція мережевої архітектури, що пропонує використовувати технології віртуалізації для цілих класів функцій мережевих вузлів у вигляді складових елементів, які можуть бути з'єднані разом або пов'язані в ланку для створення телекомунікаційних послуг (сервісів).

Будь-яка віртуалізована мережева функція складається з однієї або декількох віртуальних машин, що використовують відповідне програмне забезпечення. Для реалізації безперебійної роботи сервісу, навіть в умовах значного навантаження, використовують високопродуктивні сервери, комутатори та сховища великих об'ємів.

Віртуалізація мобільних мереж в Україні має значні перспективи розвитку, оскільки в останні роки були введені в експлуатацію мережі UMTS (Universal Mobile Telecommunication System) та LTE (Long Term Evolution), запущені сервіси NB-IoT (Narrow-band Internet of Things), MNP (Mobile Number Portability) та інші. Все це вимагало від мобільних операторів значних часових та матеріальних ресурсів. Використання технології віртуалізації мережевих ресурсів дозволить у майбутньому значно швидше впроваджувати і налагоджувати новітні технології та сервіси. Окрім того, зростуть експлуатаційні показники мережі завдяки багатократному резервуванні вузлів та швидкої адаптації топології ЦОД до експоненційно зростаючого навантаження. В майбутньому, після введення в експлуатацію мереж 5G (5 Generation) стануть доступними сервіси віртуальної реальності – VR (Virtual Reality) та відео 8К (об'єм трафіку якого в 16 разів перевищує Full-HD). Таким чином, навантаження на ЦОД з віртуалізованими мережевими функціями ядра мережі оператора стільникового зв'язку зросте багатократно.

Водночас, обробка даних потребує значних енергозатрат. Згідно джерел [3, 4] кількість енергії, що була спожита центрами обробки даних (серверами, сховищами, апаратурою зв'язку та охолодження) по всьому світу в останні роки мала такий розподіл: становила близько 70,8 ТВт у 2000 році;

152,5 ТВт – у 2005 році (спостерігалось зростання споживання на 115%); зросла до 271,8 ТВт у останні роки. При цьому частка цієї потужності у загальній кількості спожитої у світі становить в середньому 1,5%.

У зв'язку з необхідністю підвищення енергоефективності та зростанням вимог до швидкості обробки інформації гостро постає проблема розробки методів енергоефективної обробки даних у Центрах обробки даних (ЦОД), які б водночас позитивно впливали на продуктивність обчислень.

Таким чином, актуальною задачею є поєднання програмних методів зменшення енергоспоживання на серверній стороні і методу динамічного розподілу ресурсів, що дозволить значно підвищити ефективність роботи ЦОД.

Для підвищення енергоефективності та продуктивності обробки даних у ЦОД операторів зв'язку необхідно перш за все розподіляти навантаження між вузлами серверного кластера так, щоб досягти балансу між дотриманням вимог до енергоефективності та продуктивності обробки даних.

Проблема розподілу задач (балансування навантаження) вирішується за допомогою планувальника завдань (брокера) [5, 6]. У даній роботі запропоновано підхід до підвищення енергоефективності обробки даних, що застосовується саме на етапі планування завдань та представляє собою алгоритм роботи брокера-планувальника завдань. Ефективність методів перевіряється шляхом його застосування у віртуалізованому ядрі мобільної мережі одного з операторів України.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Опис середовища гетерогенної віртуалізованної інфраструктури оператора зв’язку.

Наразі актуальним стандартом передачі даних в мережах операторів зв’язку є LTE. Використовуючи сучасні технології обробки сигналу, такі як OFDM, Massive MIMO, Carrier Aggregation, спектрально ефективну модуляцію QAM-256, ця технологія забезпечує швидкість завантаження до 1 Гбіт/с. Наразі LTE є основним мобільним протоколом в більшості країн світу. Структура LTE мережі представлена на рис. 1.1. [7]

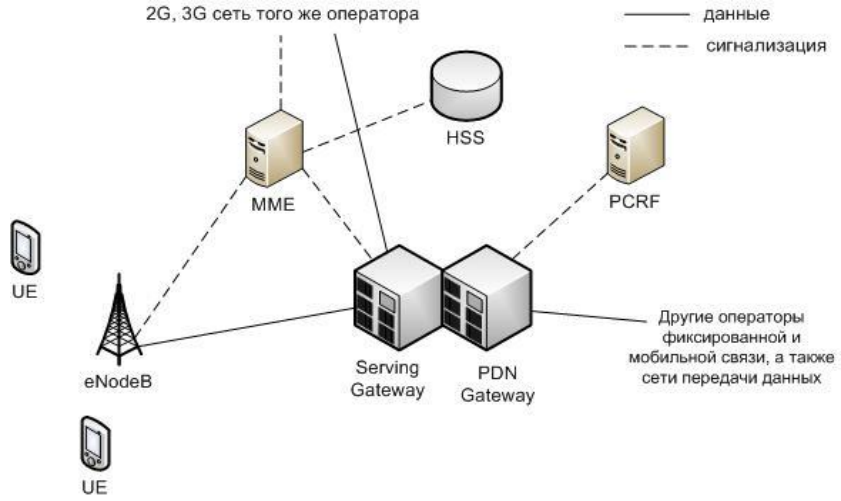


Рисунок 1.1 Структура мережі LTE

Фізичні вузли мережі можуть бути віртуалізовані та мають таке призначення:

HSS – є великою базою даних і призначений для зберігання даних про абонентів. Окрім того, HSS генерує дані, необхідні для виконання процедур шифрування, аутентифікації і т.д. Мережа LTE може включати один або декілька HSS в залежності від географічної структури мережі та кількості абонентів.

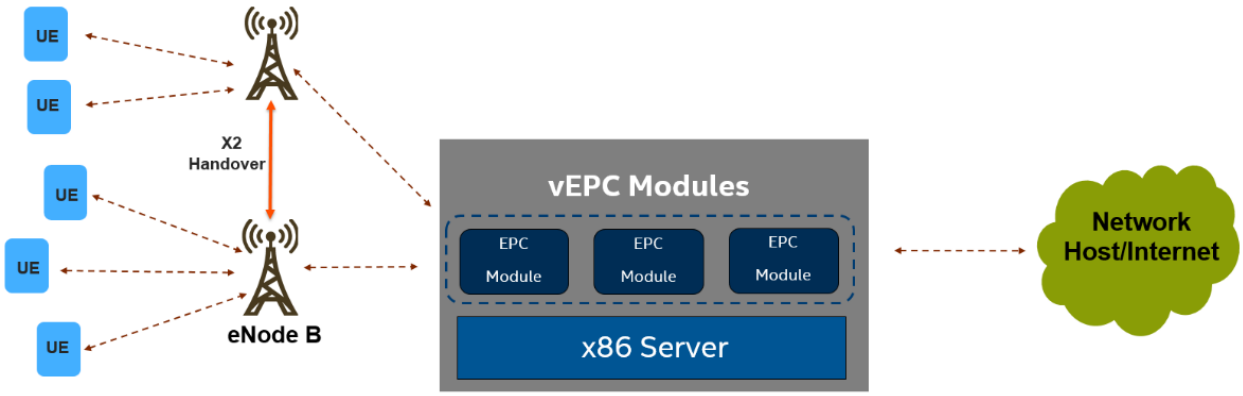
MME – призначений для обробки сигналізації, зв’язаної з керуванням мобільністю абонентів у мережі.

SGW – призначений для обробки і маршрутизації пакетних даний, що поступають з/в підсистему базових станцій. Має пряме з'єднання з мережами 2 та 3 поколінь того ж оператора.

PGW – шлюз до мереж передачі даних інших операторів для мережі LTE. Основна задача PGW – маршрутизація трафіку мережі LTE іншим мережам передачі даних, таких як Інтернет-трафік, а також мережам GSM та UMTS.

PCRF – призначений для виконання білінгових послуг, таких як нарахування плати за надані послуги зв'язку, а також за забезпечення якості з'єднання у відповідності до заданих конкретним абонентом характеристик.

Наразі технологія NFV з використанням SND (Software Defined Networks) може провести віртуалізацію як частини функціональних вузлів EPC (Evolved Packer Core) так і всіх, утворюючи vEPC (віртуалізовану EPC) [8]. Структура мережі LTE з використанням технології vEPC зображена



на рис. 1.2:

Рисунок 1.2 Структура мережі LTE з використанням vEPC

Як видно з рис. 1.2, все ядро мережі представлено модулями vEPC, які інстальовані на звичайному серверному обладнанні. При цьому один сервер може виконувати як одну визначену віртуалізовану функцію, так і всі одночасно. Схематичне зображення структури такого сервера зображено на рис. 1.3:

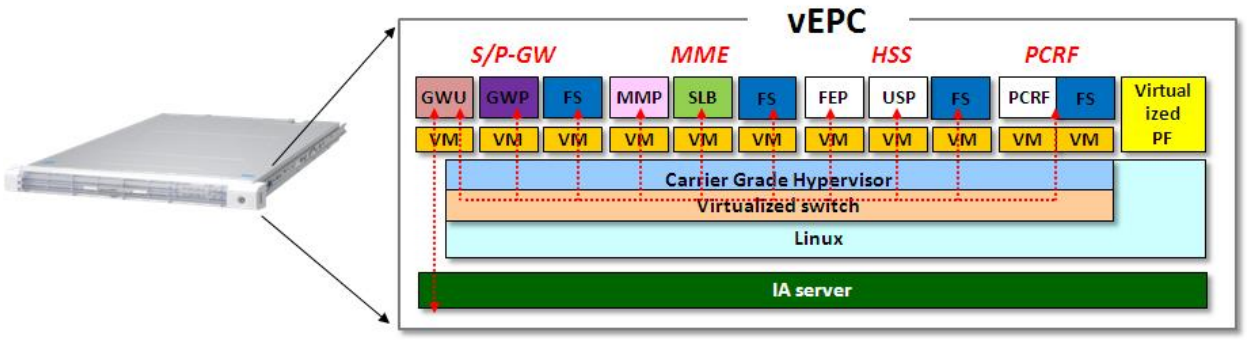


Рисунок 1.3 Структура програмної частини серверного обладнання, яке використовуються для імплементації технології vEPC.

З рис. 1.3 можна побачити, що кожна віртуалізована функція є віртуальною машиною (VM – virtual machine). При цьому конфігурація віртуальних машин (які функції використовуються, а які ні) виконується гіпервізором [9].

В залежності від розмірів та завантаженості мережі, кількість серверних блоків може варіюватися від одиниць до тисяч. Причому, при збільшенні розміру мережі збільшується її ступінь гетерогенності – неоднаковості фізичних характеристик окремих вузлів, таких як продуктивність та енергоефективність. Це спричинено тим, що розвиток мережі не відбувається в один момент часу. Поступово, із збільшенням навантаження, надходить необхідність розширення кластеру. При цьому, нові машини будуть більш продуктивними, ніж застарілі існуючі. Або, наприклад, при виході з ладу машини існує велика ймовірність, що вона буде замінена на більш продуктивну. Таким чином, можна стверджувати, що ступінь гетерогенності кластеру збільшуватиметься із плином часу, оскільки одночасна заміна всіх машин з метою збільшення продуктивності кластеру не буде виконуватися з економічних причин. І перед оператором зв’язку постає задача балансування навантаження між вузлами дата-центру, на якому розташоване віртуалізоване ядро мережі зв’язку.

1.2. Підходи щодо розподілення навантаження в дата центрах оператора зв’язку

Існуючі підходи можна класифікувати за рівнем їх застосування в системі, динамічністю прийняття рішення, сферою використання, тощо. Одним із можливих доцільних варіантів класифікації є варіант, запропонований у роботі [10]. Дану класифікацію наведено на рис.1.4

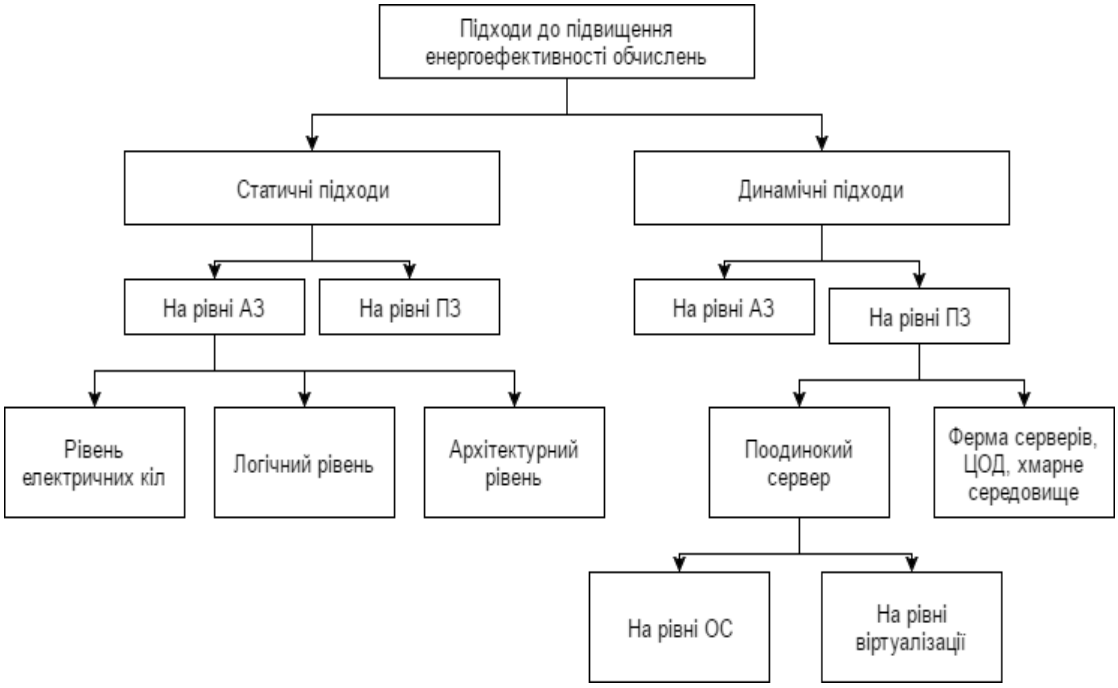


Рисунок 1.4 Класифікація підходів до вирішення проблеми енергоефективності обробки даних

За наведеною класифікацією, існуючі підходи до підвищення енергоефективності обробки даних можна розділити на статичні та динамічні за характером прийняття рішень. Аналіз існуючих рішень обох підходів дає змогу зрозуміти, що динамічні рішення мають вищу ефективність.

Як статичні, так і динамічні підходи можуть бути застосовані як на рівні апаратного забезпечення (АЗ), так і на рівні програмного забезпечення (ПЗ). При цьому, дані підходи класифікують також за областю застосування (хмарне середовище, поодинокий сервер, тощо).

Динамічні підходи на рівні програмного забезпечення можуть бути застосовані як до поодинокого сервера (на рівні його операційної системи, наприклад), так і для набору серверів – серверного кластера. При цьому серверний кластер може бути гомогенним або гетерогенним. Ця властивість кластера залежить від фізичних характеристик вузлів, що входять до нього. Гетерогенний серверний кластер складається з вузлів, що мають різні фізичні параметри. Для таких кластерів постає проблема вибору найбільш доцільного вузла обробки серед множини можливих для розміщення певної конкретної задачі, тобто проблема балансування або планування навантаження (англ.scheduling).

Крім того, рішення, націлені на підвищення енергоефективності, можуть бути застосовані на різних рівнях організації обчислювальної системи (логічному, архітектурному, операційної системи (ОС) тощо) [11].

Підходи, що застосовуються на рівні апаратного забезпечення, є більш складними в реалізації та вимагають значних додаткових фінансових вкладень, у той час як програмні підходи можуть бути застосовані за умов обмежених матеріальних та технічних ресурсів.

В обчислювальних системах ефективність обробки даних залежить у тому числі від характеру вхідного навантаження на вузол обробки, поточного стану обчислювальної системи та інших факторів, що змінюються динамічно під час роботи системи. Тому, застосування динамічних підходів здатне забезпечити більшу продуктивність у загальній енергоефективності системи, ніж статичних [12].

Серед існуючих алгоритмів балансування навантаження найбільш простим та розповсюдженим є алгоритм Round Robin. Згідно цього алгоритму всі завдання розподіляються по всім активним серверам за циклічним принципом. Цей алгоритм швидкий, адже не потребує знання про поточний розподіл ресурсів серед серверів. Але, оскільки завдання та вузли обробки (сервери) за даним алгоритмом не мають жодного пріоритету,

алгоритм не здатний призначити завдання на обробку найбільш відповідному для цього серверу. Це є основним недоліком Round Robin [13].

Однією з успішних модифікацій цього алгоритму є Round Robin зі зваженими коефіцієнтами (WeightedRoundRobin) [13]. На відміну від простого алгоритму Round Robin, дана модифікація використовує знання про обчислювальні потужності вузлів обробки і, відповідно, до значень обчислювальних потужностей, присвоює вагові коефіцієнти серверам, що в подальшому враховується при розподілі завдань між ними.

Алгоритм LC [14] розподіляє кожне наступне вхідне завдання на обробку на сервер, що має найменшу кількість поточних з'єднань. Цей алгоритм застосовується здебільшого у кластерах, де всі вузли мають однакову продуктивність. Це динамічний алгоритм планування, оскільки він бере до уваги параметр поточної завантаженості вузла.

Алгоритм FIFO розподіляє кожне наступне вхідне завдання на перший доступний сервер, тобто такий, що має достатньо ресурсів для обробки даного завдання. Цей алгоритм не враховує можливу різницю у параметрах вузлів обробки, не враховуючи, в тому числі, до уваги параметр енергоефективності. Основною перевагою даного підходу є його простота і легкість впровадження.

Усі вищеописані алгоритми не беруть до уваги параметр енергоспоживання кластерного вузла.

Серед енергоефективних підходів до планування навантаження у серверному кластері найбільш близькими до запропонованого у роботі підходу є такі: CTES [15]; EDRP [16]; Min_C [17]; The Most-Efficient-Server First Scheme [18]; «Алгоритм P» [18].

Серед відомих алгоритмів балансування, що враховують параметр енергоспоживання, слід відмітити алгоритм CTES – Cooperative Two-Tier Energy-Aware Scheduling [15]. Автори розглядають дворівневий підхід до планування завдань із регулюванням швидкості їх виконання, з метою досягнення оптимального використання процесору, замість міграції завдань

на інші вузли. Використовуються декілька стратегій планування з прогнозом виконання завдань для оптимального призначення їх на доступні машини. Результати моделювання, представлені в роботі, показують, що цей підхід зменшує загальне споживання енергії у серверному кластері.

Недоліком даного алгоритму є те, що автори вважали модель енергоспоживання сервера (функцію залежності енергоспоживання від навантаження центрального обчислювального вузла) лінійною. Насправді, така модель не точно відображає характер залежності спожитої потужності від завантаженості сервера. Натомість у даній роботі пропонується проведення попередньої атестації кожного сервера кластеру для отримання реальної залежності потужності споживання від завантаженості для кожного вузла кластеру.

Іншим прикладом алгоритму, що має на меті підвищення енергоефективності обробки задач у хмарних системах, EDRP – Energy and Deadline aware Resource Provisioning [16]. Автори зосередились на проблемі мінімізації затрат на хмарні системи, підвищуючи ефективність використання енергії, але гарантуючи терміни виконання користувацьких задач, визначених в умовах щодо якості обслуговування (SLA). Вони беруть до уваги два типи завдань, незалежні пакетні передачі і завдання із залежностями.

Їхня модель розрахунку споживання електроенергії у момент часу t включає статичне Pxstatic(t) і динамічне Pxdynamic(t) енергоспоживання. Обидві характеристики розраховуються на основі відсотку завантаження процесору Utilx (t), в якому враховуються тільки параметри використовуваної машини Q-t. Недоліком запропонованого алгоритму є те, що автори не враховують відмінностей між машинами, на яких виконуються завдання, і машинами, що знаходяться в режимі очікування.

У роботі [17] описано стратегію розподілу завдань, яка має назву Min_C. Дана стратегія враховує різноманітність завдань, що приходять на обробку та відповідну різноманітність ресурсів, яких вони потребують.

Недоліком описаного у статті [17] алгоритму є те, що енергетична модель, яку використали автори, хоч і має нелінійний характер, близький до дійсності, проте використана модель має єдиний вигляд для всіх машин в незалежності від їхніх характеристик. Натомість, у даній роботі пропонується індивідуально визначати модель енергоспоживання для кожної машини окремо.

У роботі [18] описано алгоритм планування навантаження The Most-Efficient-Server First Scheme (MES-first). Згідно даного підходу, центральний планувальник задач сортує вузли кластера, базуючись на їх енергоспоживанні. Завдання для обробки розподіляються спершу на найбільш енергоефективні сервери, згодом на менш енергоефективні, згідно позиції сервера у відсортованому списку. Розподіл завдань припиняється, якщо не лишається завдань у черзі на обробку, або якщо черги до всіх серверів заповнені. Для ЦОД, у яких параметри всіх вузлів є однаковими, цей алгоритм має на меті розподілити якомога більше завдань на сервери, що знаходяться в активному стані, до моменту досягнення точки насичення, а вже потім залучати сервери, що знаходяться у режимі сну.

Вузол, на якому розміщується центральний планувальник, містить сортований список доступних серверів та їхні енергетичні профілі. У такому списку найбільш енергоефективні сервери розміщені у верхній частині списку. В момент прибуття завдання, воно призначається на обробку на сервер із вершини списку. Сервер, що отримує завдання на обробку, оновлює свій енергетичний профіль у центральному вузлі із планувальником. Після заповнення найбільш енергоефективного сервера завдання розподілятимуться на сервер, що займає наступну позицію у списку.

Недоліком запропонованого у роботі [18] підходу є те, що автори не враховують параметра продуктивності обробки завдань при їхньому розподілі. Може виникнути ситуація, за якої найбільш енергоефективні сервери виявляться найменш продуктивними. При розподілі задач виключно за параметром енергоефективності існує ймовірність значного програшу у

швидкості обробки завдань та недотримання, відповідно, прийнятих SLA (англ.Service Level Agreement – домовленість про якість сервісу).

У роботі [19] описано підхід до енергоефективного розподілу завдань, що має назву «Алгоритм Р». Даний алгоритм автори пропонують застосовувати при розподілі завдань між кількома незалежними серверними кластерами. Згідно запропонованого авторами підходу, вхідним параметром для «Алгоритму Р» є значення H_{diff}, що визначає допустиму різницю між мінімальним та максимальним часом очікування завдання у черзі до кластера, за якого алгоритм буде давати економію електроенергії. Алгоритм вибиратиме кластер, у якому завдання буде оброблено із найменшими затратами енергії. Для кожного кластера планувальник навантаження C_j оцінює вартість електроенергії, що буде затрачена на обробку завдання, і параметр $h_j = (\sum_{k=1}^{N} S_k)/W_j$ - відношення загальної площі завдання до ширини кластера (цей параметр визначає оцінку часу перебування завдання у черзі). Якщо $max(h_j) - min(h_j) > H_{diff}$, то завдання розподіляється на обробку на кластер з мінімальним значенням h_j.Інакше, завдання розподіляється на кластер із мінімальною вартістю енергії.

Автори роботи [19] розглядали у якості оцінюваних параметрів час очікування завдання у черзі та вартість електроенергії, вважаючи її різною для різних кластерів. У випадку застосування підходу у межах одного серверного кластера, оцінюваними параметрами можуть виступати час знаходження завдання у черзі до кожного із серверів та вартість обробки завдання на кожному окремому сервері.

Серед недоліків вищеописаного підходу можна виділити те, що автори не розглядають параметр продуктивності кожного окремого сервера як оцінюваний. Вибір вузла обробки здійснюється на основі поточного стану кластера та завантаженості його вузлів. Такий підхід може негативно вплинути на продуктивність обчислювальної системи в цілому.

У роботі [20] запропонований метод динамічного розподілу, базований на квантовому генетичному алгоритмі. Даний алгоритм пропонується

використовувати як високорівневий інструмент резервування ресурсів. Автори роботи [20] пояснюють використання квантового генетичного алгоритму, який є розвитком генетичного алгоритму, його високою продуктивністю для задач цілочисленного програмування. Недоліком використання генетичного алгоритму в задачі динамічного розподілу ресурсів є висока обчислювальна складність функції пристосованості для складної багаторозмірної задачі, та погана масштабованість генетичного алгоритму під складність задачі. Це пов’язане із значною кількістю елементів, схильних до спотворень у випадку великого розміру області пошуку рішень.

У роботі [21] запропонована архітектура системи в площині керування, яка дозволяє виконувати швидкий та безпечний розподіл потоків серед блоків віртуалізованих функцій. Ця архітектура дозволяє вирішувати декілька важливих задач одночасно:

1. Проблема «перегонів»;
2. Обмеження перенавантажень;
3. Розміщення різноманітних мережевих функцій з мінімальними змінами в системі.

Недоліком даного підходу є значне збільшення часу обробки кожного пакету даних (до 10%) за рахунок збільшення об’єму та складності коду, який виконується на контролюючому вузлі.

У роботі [22] запропонований підхід, який отримав назву «розділення – об’єднання», який полягає в розділенні вхідного потоку даних на декілька частин, їхню подальшу обробку декількома віртуальними машинами та об’єднання в єдиний вихідний потік даних. Завдяки такому підходу досягається гнучкість та масштабованість системи у виконанні процесів будь-якої складності. Недоліком такого підходу є підвищені вимоги до взаємного узгодження віртуальних машин як на фізичному рівні (затримка) так і на програмному.

У роботі [23] запропонований підхід, який включає в себе 2 компоненти:

1. Створення опису для кожного потоку даних, в якому зазначаються вимоги до масштабування та стійкості системи. Такий опис передається разом із самим потоком даних.

2. Створення додаткового блоку під назвою «Система управління інфраструктурою орендарів» (англ. Tenant Infrastructure Management System), основною задачею якого є зчитування інформації з опису кожного потоку даних та, відповідно, створення моделі розподілу. Менеджер ресурсів, використовуючи цю модель, виділяє відповідну кількість обчислювальних ресурсів.

Недоліком такого підходу є низька гнучкість (наразі, розроблена система для роботи з лише двома вимогами, які зазначаються в описі потоку: масштабованість та стійкість системи).

В роботі [24] запропонований алгоритм VNR (Virtual Network Reconfiguration – реконфігурація віртуальної мережі), метою якого є підвищення стійкості системи. Це досягається розбиттям ресурсів однієї машини на підмножину віртуальних машин, об’єднаних в топологію «зірка». При перевантаженні однієї з цих віртуальних машин виконується процедура перевизначення топології (будується нова топологія з тимчасовим виключенням перевантаженої віртуальної машини). Недоліком такого підходу є відсутність ефективності при значній гетерогенності виконуваних віртуальних функцій на різних віртуальних машинах.

1.3. Вимоги до створюваного програмного забезпечення брокера

Основні вимоги, що висуваються до рішення з балансування навантаження в серверному кластері наведено нижче:

1. Операційна система: UNIX, Windows

2. Підтримувані алгоритми балансування навантаження: PCPB, Round Robin

3. Надійність: відсутність в архітектурі програми вразливих місць, вихід з ладу яких унеможливлює роботу всієї програми

4. Безпека: шифрування передаваних даних

5. Простота інсталяції та налаштування

6. Універсальність використання ПЗ для будь-яких задач

1.4. Програмні технології побудови середовища гетерогенної віртуалізованної інфраструктури оператора зв'язку

З урахуванням наведених вище вимог, доцільним є використання в програмному забезпеченні брокера наступних технологій:

1.4.1. Docker

Docker — інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів. Зокрема, Docker дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції і потім переносити і клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування і підтримки контейнерів.

Таким чином, в контейнер вміщується образ операційної системи UNIX з усіма необхідними для роботи програми бібліотеками та інструментами. Це значно спрощує процес інсталяції та налаштування програми, а також робить її універсальною та крос-платформною. ПЗ Docker можна інсталювати на будь-якій операційній системи сімейства UNIX, Windows та MacOS. До інших переваг Docker відносять:

1. Використання легковагих контейнерів для ізоляції процесів від інших процесів і основної системи.

2. Оскільки контейнери використовують свою власну самодостатню файлову систему, не важливо де, коли і в якому оточенні вони запускаються.

3. Ізоляція на рівні файлової системи: кожен процес виконується у повністю окремій кореневій ФС;

4. Ізоляція ресурсів: споживання системних ресурсів, таких як витрата пам'яті і навантаження на CPU, можуть обмежуватися окремо для кожного контейнера з використанням cgroups;

В розробленому ПЗ з балансування навантаження Docker використовується для розміщення всіх основних компонентів програми в одному універсальному контейнері, для використання якого на будь-якому комп’ютері необхідно лише мати встановлене ПЗ Docker.

1.4.2. Dockerfile та Docker-compose

Побудова стандартного docker-контейнера з включенням в нього бібліотек, виконуваних файлів виконується шляхом виконання декількох команд в консолі. Цей метод не є оптимальним, оскільки значно ускладнює процес інсталяції у недосвідчених користувачів. Технологія Dockerfile дозволяє записати всі ці команди в один виконуваний файл і спростити процес інсталяції (необхідно ввести лише одну команду замість багатьох)

В свою чергу, недоліком даної технології є ускладнення команди при необхідності додаткових налаштувань контейнера: призначення портів, прав адміністратора тощо – все це повинно передаватися, як аргумент команди. Рішенням цієї проблеми є технологія Docker-compose, яка дозволяє записати всі передавані налаштування в один файл. Головною функцією Docker-compose є зв’язування контейнерів між собою, а також просту інсталяцію та запуск всієї системи однією командою:

docker-compose up –build

1.4.3. SSH

SSH (Secure SHell - «безпечна оболонка») - мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань (наприклад, для передачі файлів). Схожий за функціональністю з протоколом Telnet і rlogin, проте шифрує весь трафік, в тому числі і паролі, що передаються [25].

Криптографічний захист протоколу SSH не фіксований, можливий вибір різних алгоритмів шифрування. Клієнти і сервери, що підтримують цей протокол, доступні для різних платформ. Крім того, протокол дозволяє не тільки використовувати безпечний віддалений shell на машині, але і туннелювати графічний інтерфейс — X Tunnelling (тільки для Unix-подібних ОС або застосунків, що використовують графічний інтерфейс X Window System). Так само ssh здатний передавати через безпечний канал (Port Forwarding) будь-який інший мережевий протокол

SSH використовується для захищеного зв’язку між вузлами системи та брокером.

1.4.4. RAFT

Особливістю створеного ПЗ є універсальність: будь-який вузол може виконувати функції як брокера так і обробника запитів. При цьому лише один вузол може виконувати функції балансування навантаження в певний момент часу. У випадку виходу з ладу брокера будь-який інший вузол повинен зайняти його місце. Таке рішення дозволить значно підвищити відмовостійкість системи.

Для цього ПЗ використовує алгоритм RAFT суть якого полягає у використанні трьох ролей для кожного вузла:

1. Leader – лідер
2. Candidate – кандидат

3. Follower – послідовник

В системі обов’язково повинен бути тільки один лідер, який обирається загальним голосуванням. З моменту ініціалізації роботи алгоритма кожен вузол має статус послідовника і певний випадковий таймер, після якого він отримує статус кандидата. При цьому кандидат розсилає всім іншим запит на голосування. Інші вузли голосують за того кандидата, від якого вони отримали перший запит. Якщо ж він не отримав більшості (це випадок, коли на кластері виникли відразу кілька кандидатів і голоси розділилися), то кандидат чекає випадковий час і ініціює нову процедуру голосування. Процедура голосування повторюється, поки не буде обраний лідер [26, 27].

Якщо звичайний вузол довго не отримує повідомлень від лідера, то він переходить в стан «кандидат» і посилає іншим вузлам запит на голосування. При цьому якщо кандидат отримує повідомлення від лідера, то він знімає свою кандидатуру і повертається в нормальний стан

Особливістю RAFT є те, що тільки лідер може направляти запити на інші вузли. Це дозволяє підвищити доступність системи завдяки наявності консенсусу між всіма її вузлами. Окрім того, значно підвищується відмовостійкість системи, оскільки всі функції багатократно дублюються, то вивести з ладу таку систему можливо лише вивівши з ладу всі вузли, які входять до цієї системи [28].

1.4.5. Python

Python – високорівнева мова програмування, на якій розроблено ПЗ. До переваг цієї мови програмування відносять:

1. Чистий синтаксис (для виділення блоків слід використовувати відступи);
2. Стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);

3. Зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

4. Відкритий код (можливість редагувати його іншими користувачами)

5. Наявність великої кількості бібліотек, які значно поширюють можливості з проведення будь-яких досліджень, в тому числі, з використанням алгоритмів машинного навчання.

Вибір цієї мови програмування спричинений її простотою, можливостями та її поширеності.

Висновки

1. Розглянуто середовище гетерогенної віртуалізованної інфраструктури оператора зв’язку на прикладі мережі стандарту LTE. Проаналізовано можливості віртуалізації окремих компонентів мережі LTE з використанням технологій SDN та NFV.

2. Проаналізовано класифікацію підходів до вирішення проблеми енергоефективності обробки даних. Задача балансування навантаження з використанням брокера відноситься до динамічних підходів на рівні програмного забезпечення. Розглянуті існуючі підходи щодо балансування навантаження дозволили визначити як основний недолік відсутність одночасного врахування продуктивності та енергоефективності серверів.

3. Визначено доцільність створення алгоритму балансування навантаження, який буде ефективним у випадку значного ступеню гетерогенності серверного кластеру та дозволить зменшити енергоспоживання системи при збереженні та покращенні основних показників якості обслуговування.

4. Визначено основні вимоги до ПЗ брокера з використанням підходу до балансування навантаження з одночасним урахуванням продуктивності та енергоефективності вузлів серверного кластера.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ЗАПРОПОНОВАНОГО МЕТОДУ ДИНАМІЧНОГО РОЗПОДІЛУ РЕСУРСІВ

У процесі аналізу існуючих підходів до підвищення енергоефективності обробки задач у ЦОД було визначено, що для підвищення загальної енергоефективності та продуктивності обробки даних, найбільш доцільним є внесення змін у процес планування завдань. Крім того, для опису енергоефективного підходу до планування завдань, доцільно на початковому етапі розглянути його застосування у серверному кластері, як одиниці інформаційної інфраструктури ЦОД.

Нехай технологія віртуалізації NFV застосовується до ядра мобільної мережі LTE (Long Term Evolution) України.

2.1. Опис експериментального дослідження енергоефективності ЦОД, що виконує функції ядра мережі оператора стільникового зв'язку України.

Експериментальне дослідження проводиться засобами імітаційного моделювання середовища Matlab за таким планом:

Підготовчі кроки:

Крок 1. Визначення розподілу трафіку в мережі.

Крок 2. Побудова топології мережі.

Крок 3. Моделювання добової кривої навантаження.

Крок 4. Визначення реального навантаження на мережах LTE кожного міста.

Основні кроки:

Крок 5. Проведення моделювання. Аналіз результатів [29, 30]

В моделі для спрощення топології допущено, що мережа LTE розгорнута в 5 найбільших містах в різних частинах України. Особливості топології такої мережі представлено в п.2.2.

2.2. Підготовчі кроки експериментального дослідження.

Крок 1. Визначення розподілу трафіку в мережі

Для урахування об’єму навантаження на вузли в залежності від географічного розташування останніх необхідно ввести корегуючі географічні коефіцієнти навантаження. Для цього розглянемо таблицю 2.1, в якій наведена кількість населення в містах Київ, Харків, Одеса, Дніпро, Львів станом на 2014 рік за статистикою з відкритих джерел [31]. Вважатимемо, що послугами зв'язку користується 80% населення, з них, згідно прогнозу Vodafone-Україна, 20% матиме LTE-сумісні телефонні апарати.

Таблиця 2.1 Кількість населення в найбільших містах України станом на 2014 рік

Місто	**Область, АРК, міськрада**	**Чисельність населення, 2014**	**Абонентів LTE**
Київ	м. Київ	2 868 702	458 992
Харків	Харківська	1 451 132	232 181
Одеса	Одеська	1 017 022	162 724
Дніпро	Дніпропетровська	993 094	158 895
Львів	Львівська	729 038	116 646

Додатково необхідно врахувати особливості профілю користувача в кожному регіоні. Молоді люди у віковій категорії 15-24 років споживають набагато більше трафіку, ніж представники інших вікових категорій. У табл. 2.2 показана доля LTE трафіку згідно статистики оператора «Мегафон» у 15 найбільших ВНЗ Новосибірська [32, 33].

Таблиця 2.2 Порівняльний аналіз структури спожитого трафіку серед студентів ВНЗ із загальномережевими показниками

Середня доля LTE трафіку в мережі «Мегафон»	**Середня доля LTE трафіку в мережі "Мегафон" серед студентів ВНЗ**	**Середній об'єм спожитого трафіку у мережі «Мегафон»**	**Середній об'єм спожитого трафіку у мережі «Мегафон» серед студентів ВНЗ**	**Середній об'єм спожитого трафіку у мережі LTE «Мегафон»**	**Середній об'єм спожитого трафіку у мережі LTE «Мегафон» серед студентів ВНЗ**	**Відношення спожитого LTE трафіку серед студентів ВНЗ до загальномережевого**
43,00%	57,26%	4286 МБ	6852 МБ	1843 МБ	3923 МБ	2,13

З табл. 2.2 видно, що у середньому студент споживає у 2,13 рази більше LTE трафіку, ніж медіанне значення по мережі.

Використовуючи дані Держстату [34], визначимо кількість студентів у кожному місті, та кількість абонентів LTE мережі серед них та занесемо отримані дані в табл. 2.3.

Таблиця 2.3 Порівняльний аналіз кількості LTE абонентів серед найактивніших споживачів трафіку

Місто	**Чисельність населення, 2014**	**Відсоток студентів**	**Кількість студентів**	**Кількість LTE абонентів серед студентів**	**Відсоток LTE-суміних пристроїв**
Київ	2 868 702	13,40%	384 406	172 983	45%
Харків	1 451 132	11,90%	172 685	77 708	45%
Одеса	1 017 022	4,30%	43 732	19 679	45%
Дніпро	993 094	3,80%	37 738	16 982	45%
Львів	729 038	4,50%	32 807	14 763	45%

Враховуючи, що середній об’єм спожитого трафіку згідно дослідженням GlobalData складатиме 1Гб/міс [35], а для абонентів-студентів

ВНЗ, відповідно – 2,13 Гб, та приймаючи географічний коефіцієнт Львова таким, що дорівнює одиниці, розрахуємо значення географічних коефіцієнтів інших міст. Отримані дані занесемо в табл. 2.4.

Таблиця 2.4 Географічний розподіл навантаження

Місто	**Абонентів LTE**	**Кількість LTE абонентів серед студентів**	**Кількість інших абонентів**	**Спожитий трафік абонентами – студентами,** *ГБ/міс*	**Спожитий іншими абонентами трафік,** *ГБ/міс*	**Сумарний спожитий трафік,** *ГБ/міс*	**Значення географічного коефіцієнта (Vtraff.city -Vtraff.Lviv)**
Київ	458 992	172 983	286 010	368453,2	286009,6	654462,8	4,91
Харків	232 181	77 708	154 473	165518,3	154473	319991,3	2,40
Одеса	162 724	19 679	143 044	41917,07	143044,1	184961,2	1,39
Дніпро	158 895	16 982	141 913	36171,46	141913,1	178084,6	1,34
Львів	116 646	14 763	101 883	31445,23	101883,1	133328,3	1,00

Оскільки мережа LTE складається з декількох вузлів, то в моделі прийняті коефіцієнти навантаження кожного вузла:

HSS – 0,05. HSS є великою базою даних і призначений для зберігання даних про абонентів. Окрім того, HSS генерує дані, необхідні для виконання процедур шифрування, аутентифікації і т.д. Мережа LTE може включати один або декілька HSS в залежності від географічної структури мережі і кількості абонентів. HSS облуговує службовий трафік, тому об’єм трафіку, що обробляється HSS складає 5% від загального.

MME – 0,05. Призначений для обробки сигналізації, зв’язаної з керуванням мобільністю абонентів у мережі. Як і HSS обслуговує службовий трафік, тому його об’єм складає 5% від загального.

SGW – 0,85. Призначений для обробки і маршрутизації пакетних даних, що поступають з/в підсистему базових станцій. Має пряме з’єднання з мережами 2 та 3 поколінь того ж оператора. SGW обслуговує приблизно 85%

всього трафіку, оскільки існує певна кількість абонентів, що не мають сумісного з 4G обладнання.

PGW – 0,7. Шлюз до мереж передачі даних інших операторів для мережі LTE. Основне завдання PGW – маршрутизація трафіку мережі LTE іншим мережам передачі даних, таких як Інтернет-трафік, а також мережам GSM та UMTS. Оскільки існує незначний у загальних масштабах об’єм даних, які оброблюються SGW, призначений для з’єднання абонентів однієї мережі, тому PGW оброблятиме 70% від загального об’єму трафіку.

PCRF – 0,05. Призначений для виконання білінгових послуг, таких як нарахування плати за надані послуги зв’язку, а також за забезпечення якості з’єднання у відповідності до заданих конкретному абоненту характеристиками. PCRF обслуговує службовий трафік в об’ємі 5% від загального.

В табл. 2.5 представлено розподіл навантаження між вузлами мережі одного міста згідно їхнього опису в пункті застосування NFV-технології для віртуалізації ядра мережі LTE.

Таблиця 2.5 Розподіл навантаження вузлами мережі

Назва вузла мережі	**Об'єм трафіка, що оброблюється вузлом (% від загального)**
HSS	5%
MME	5%
SGW	85%
PGW	70%
PCRF	5%

При чому віртуалізовані мережеві функції EPC LTE можуть бути імплементовані на серверному кластері як в комбінованому (одна машина може виконувати одночасно декілька функцій) так і в стандартному режимі (одна машина виконує тільки одну функцію).

Розподіл, наведений в табл. 2.5 в поєднанні з можливими режимами імплементації vEPC (стандартний або комбінований) робить актуальною

задачею перевірку роботи створюваного алгоритму балансування навантаження на кластерах різного розміру, як невеликого (5-20 машин) так і значно більшого (понад 100).

Крок 2. Побудова топології мережі

Серверні конфігурації визначено таким чином, щоб забезпечити безперебійне функціонування мережі міста при максимально можливих для цього регіону навантаженнях із запасом в 10%. LTE мережа кожного міста складається з одного комплекту усіх функціональних елементів, віртуалізованих в дата-центрі.

м. Київ – 1 SGW, 1 PGW, 1 MME, 1 HSS, 1 PCRF;

м. Львів – 1 SGW, 1 PGW, 1 MME, 1 HSS, 1 PCRF;

м. Харків – 1 SGW, 1 PGW, 1 MME, 1 HSS, 1 PCRF;

м. Одеса – 1 SGW, 1 PGW, 1 MME, 1 HSS, 1 PCRF;

м. Дніпро – 1 SGW, 1 PGW, 1 MME, 1 HSS, 1 PCRF.

Така топологія дозволить підтримувати в робочому стані LTE мережу міста у випадку зникнення зв’язку з елементами мережі оператора, розташованих в інших містах [36]. Зазначений підхід дозволить виконувати реконфігурацію мережі у випадках, коли функціональні вузли мережі одного міста перевантажені, коли з’являються черги на обслуговування та погіршуються показники QoS.

Таким чином, топологія мережі матиме вигляд, представлений на рис. 2.1.

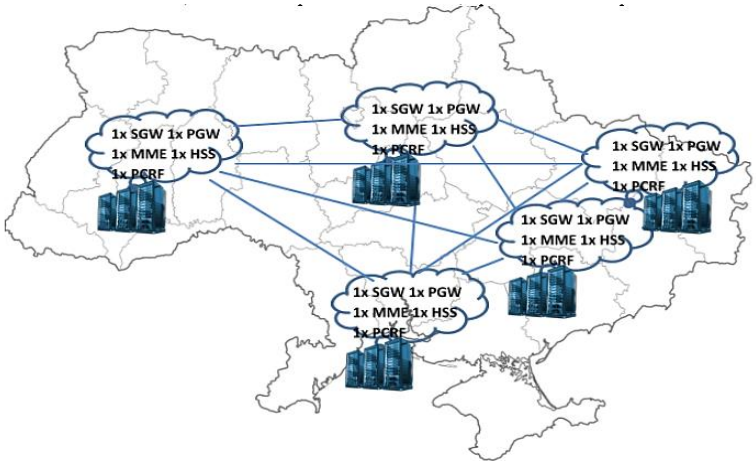


Рисунок 2.1 Топологія мережі, яка моделюється

Крок 3. Моделювання добової кривої навантаження

Приклад розподілу навантаження на реальний кластер протягом доби зображено на рис. 2.2 [37]. Даний розподіл було використано при моделюванні.

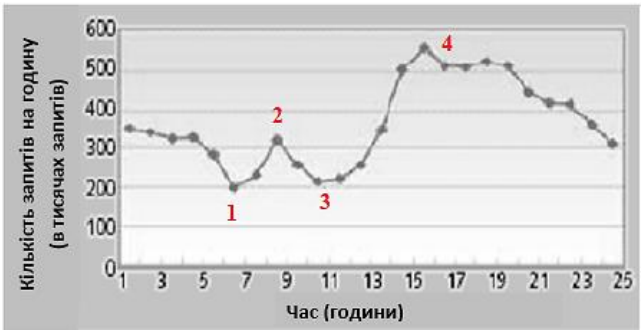


Рисунок 2.2 Приклад розподілу навантаження на реальний серверний кластер протягом доби [37]

З рис. 2.2 можна зробити висновок, що добова крива навантаження на серверний кластер має декілька проміжків зростання та спадання:

— від початку доби до першого мінімуму добового навантаження, позначеного цифрою 1, навантаження на серверний кластер має спадаючий характер, досягається мінімальне добове значення;

— від першого мінімуму до першого піку навантаження, позначеного цифрою 2, навантаження має тенденцію зростання;

— від першого піку до другого мінімуму, позначеного цифрою 3, навантаження спадає;

— від другого мінімуму до другого піку, позначеного цифрою 4, навантаження значно зростає, досягається максимальне добове значення;

— від другого піку навантаження до кінця доби навантаження спадає.

З урахуванням описаних вище закономірностей часового розподілу добового навантаження на серверний кластер, визначимо середні значення навантаження для моментів часу, що дорівнюють часу доби в 1,2,3,…,24 год. Сумарне добове навантаження при цьому дорівнює сумі споживання трафіку

по містах за місяць (колонка «Сумарний спожитий трафік, ГБ/міс» табл. 2.4) поділене на кількість днів в місяці (30).

Всі отримані значення заносяться в табл. 2.6.

Таблиця 2.6 Добовий розподіл навантаження

Час доби, *год*	**Навантаження,** *Гб/год*	**Час доби,** *год*	**Середнє погодинне навантаження,** *Гб/год*
1	1500	13	2125
2	1250	14	2437,5
3	875	15	2625
4	750	16	2812,5
5	625	17	2843,75
6	875	18	2937,5
7	1062,5	19	3000
8	1687,5	20	3250
9	1875	21	2875
10	2125	22	2500
11	2625	23	2187,5
12	2250	24	1687,5

Для кращого сприйняття змін середнього погодинного навантаження телекомунікаційної мережі України на рис. 2.3 візуалізовано дані, наведені у табл. 2.5, із зазначенням характерних точок.

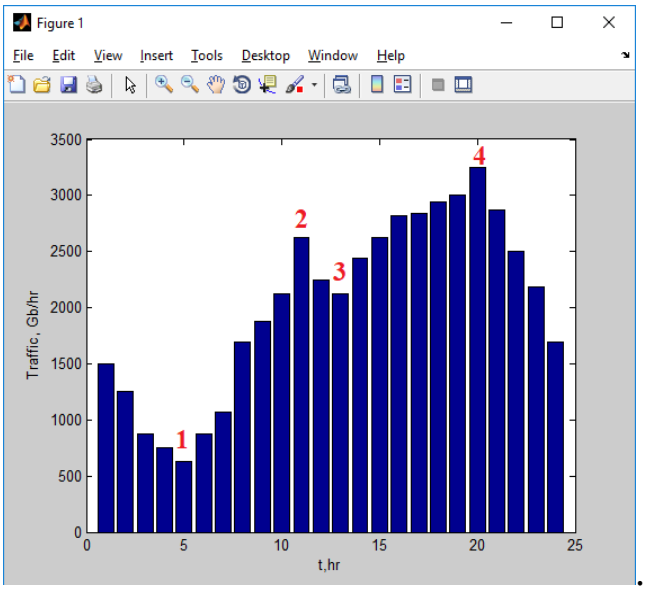


Рисунок 2.3 Гістограма погодинного споживання трафіку LTE мережі

З рис. 2.3. видно, що погодинна характеристика споживання трафіку має декілька характерних точок, які відповідають локальним мінімумам та максимумам споживання.

Оскільки часовий крок в 1 годину для умов експерименту є занадто великим, то його необхідно зменшити до розмірності 1 хвилини. Використання однієї інтерполяційної формули для великої кількості вузлів, як у випадку інтерполяційних формул Ньютона або Лагранжа, є недоцільним. Такий інтерполяційний поліном (многочлен) характеризується значною інтенсивністю коливань, і його значення між вузлами сильно відрізнятимуться від значень інтерпольованої функції. Таким чином, при невисоких значеннях порядку точність інтерполяції є недостатньою [38, 39]. При використанні більш високих порядків починає проявляти себе феномен Рунге, який полягає в збільшенні до неприпустимих значень похибки між функцією і її інтерполюючим поліномом в точках, які не є точками інтерполяції,.

На рис. 2.4 показано ефект феномену Рунге при проведенні інтерполяції поліномом вищого порядку, де червона крива – це функція Рунге.

Блакитна крива – інтерполяція поліномом 5-го порядку (використовуючи п'ять рівновіддалених точок інтерполяції).

Зелена крива – інтерполяція поліномом 9-го порядку (використовуючи дев'ять рівновіддалених точок інтерполяції).

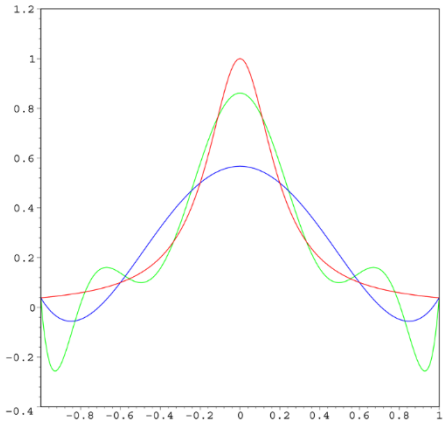


Рисунок 2.4 Феномен Рунге

На рис. 2.4 видно, що в точках інтерполяції, помилка між функцією та інтерполючим поліномом (за визначенням) нульова. Між точками інтерполяції (особливо в регіоні близькому до крайніх точок *1, -1*), похибка між функцією і інтерполючим поліномом для поліномів більш високого порядку стає надто великою.

Однією з можливостей обійти такий недолік є застосування сплайн-інтерполяції [20]. Ідея сплайн-інтерполяції полягає в побудові поліномів між парами сусідніх вузлів інтерполяції, причому для кожної пари вузлів будується свій поліном. В задачах інтерполяції, інтерполяція сплайном краща, ніж інтерполяція многочленом, оскільки дає схожі результати навіть при менших степенях поліномів, а також при її використанні не виникає феномена Рунге.

Після проведення інтерполяції кубічними сплайнами за допомогою Matlab отримаємо похвилинний графік добового навантаження, який приймемо за базове навантаження, який зображено на рис. 7.8.

На рис. 2.5 зображено результат інтерполяції сплайнами, який точно відображає залежність зміни споживання трафіку протягом дня з точністю до хвилини. При цьому кумулятивна похибка інтерполяції складає менше 0,1%.

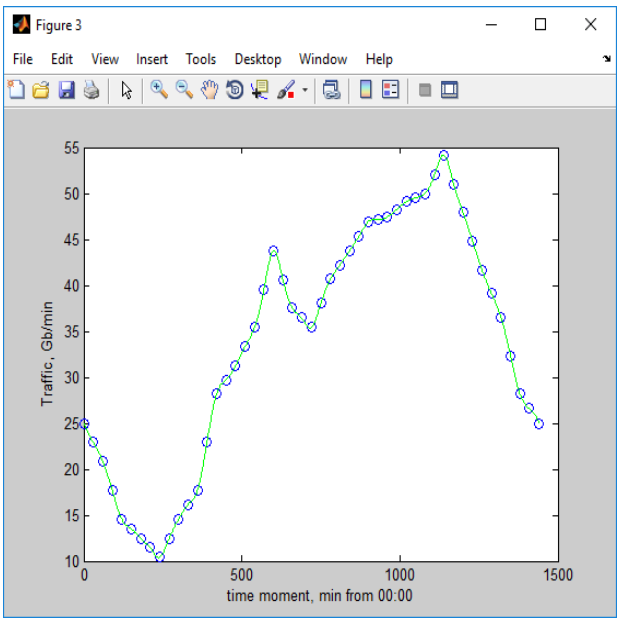


Рисунок 2.5 Базова похвилинна крива навантаження

Крок 4. Визначення реального навантаження на мережах LTE кожного міста

Враховуючи те, що реальне навантаження не буде точно співпадати з базовим, оскільки останнє є усередненням статистичних даних, приймемо допущення, що реальне навантаження можна вирахувати таким чином:

$$Yreal = Ybasic + rand\ (-0.05 \ldots 0.05) * Ybasic. \quad (2.1)$$

Іншими словами, до базового навантаження вноситься випадкове відхилення від базового значення, що відповідає реальній ситуації на будь-якій ділянці мережі.

На рис.2.6 показано розподілення сумарного навантаження по містам у відповідності до значень географічних коефіцієнтів та внесеного відхилення навантаження від базового.

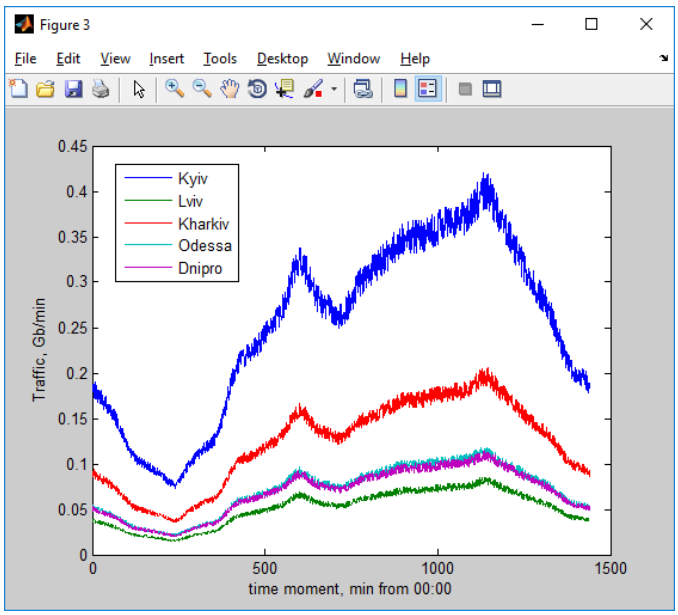


Рисунок 2.6 Розподіл реального навантаження для визначених міст

На рис. 2.6 можна побачити, що абсолютний динамічний діапазон споживання трафіку для кожного міста різний та вимагатиме впровадження індивідуальних налаштувань роботи алгоритмів резервування обчислювальних ресурсів.

Крок 5. Проведення моделювання. Аналіз результатів

Опис виконання процесу моделювання, а також аналіз його результатів наведено в пунктах 2.3 та 2.4 цього дослідження.

2.3. Основна частина експериментального дослідження.

2.3.1. Введення основних термінів та понять

У даній роботі використані такі терміни та поняття:

Обчислювальний кластер або серверний кластер – набір комп'ютерів, що працюють разом і можуть розглядатись як єдина система.

Вузол серверного кластера – одиниця обчислювального кластера, представлена фізично однією ЕОМ:

$$ComputerCluster = \{N_j\},$$

де N_j - j-й вузол обчислювального кластера.

FLOPS – одиниця вимірювання продуктивності комп'ютера (кількість операцій з плаваючою комою, що можуть бути виконані за секунду) [40].

Завдання – набір елементарних операцій, що мають бути обробленими неподільно в обчислювальному кластері:

$$Task = \{task_i\}$$

Робота – завдання разом з вимогами до її виконання:

$$Jobs = \{job_i\} \rightarrow \left\{task_i,\ \left\{V_{req_i},\ k_{core_{req_i}},\ t_{i\_max}\right\}\right\},$$

де V_{req_i} – об'єм оперативної пам'яті для виконання завдання; $k_{core_{req_i}}$ – кількість ядер процесора, що вимагає завдання $task_i$ для свого виконання;

t_{i_max} – максимальний час, за який завдання $task_i$ має бути оброблено. [41]

2.3.2. Постановка завдання енергоефективного планування в математичному вигляді

Обчислювальний кластер складається з n вузлів $\{N_j\}$. Кожен вузол N_j характеризується:

V_j – об'ємом доступної оперативної пам'яті;

$flops_j$ – продуктивністю вузла N_j, що має k_{core_j} обчислювальних ядер.

$P_j = f_j(CPU_j)$ – функцією енергоспоживання від завантаженості процесора $f_j(CPU_j)$, що експериментально визначена для кожного вузла N_j.

Нове завдання $task_i$ приходить до системи в момент τ.

Кожне завдання потребує певних значень вищеназваних параметрів: $\{V_{req},\ k_{core_{req}},\ t_{max}\}$

$$job_i \rightarrow task_i, \{V_{req_i},\ k_{core_{req_i}},\ t_{i_{max}}\}$$

Необхідно розробити алгоритм планування завдань такий, що

$P_\sum \rightarrow min, t_{task\_i} \rightarrow min,$

де $P_{\sum}$ – сумарна потужність спожита усім кластером,

$t_{\sum}$ – час виконання набору із m задач [42].

2.3.3. Опис запропонованого алгоритму енергоефективного планування завдань

Запропонований підхід до енергоефективного планування завдань складається з двох основних етапів:

1. Етап попередньої атестації;
2. Етап планування завдань.

Етап попередньої атестації проводиться у серверному кластері періодично, при його налаштуванні. Цей етап передбачає попереднє визначення функцій *P = f(CPU)* індивідуально для кожного вузла кластера та формування описів вузлів, що розміщено на центральному вузлі із планувальником (брокером). Отримання залежності $P_j = f_j(CPU_j)$ детально описано у роботах [43, 44]. Індивідуальне визначення функцій *P = f(CPU)* для кожного вузла обробки та їх подальше використання у процесі планування завдань є ключовою особливістю запропонованого підходу.

Другий етап передбачає вирішення оптимізаційного завдання за критеріями енергоефективності та продуктивності обробки завдань з використанням індивідуально визначених енергетичних моделей вузлів. Результатом цього процесу є розміщення кожного поточного завдання для обробки на сервер із оптимальними параметрами.

Запропонований підхід представлено у вигляді алгоритму планувальника завдань, що складається з таких кроків:

Крок 0. Попередня атестація кластера

Крок 1. Оцінка стану кластера в момент τ_{k-1}

Крок 2. Виключення із розгляду всіх непідходящих вузлів (за відсутністю ресурсів для обробки)

Крок 3. Знаходження масиву значень сумарного енергоспоживання кластера $P_\Sigma = \{P_{\Sigma_j}\}$ *при розміщенні завдання* $task_i$ *на кожен з вузлів* N_j

Крок 4. Сортування масиву вузлів N_j *за теоретичним вкладом у загальну спожиту потужність:* $N_P = \{N_{P_j}\}$

Крок 5. Сортування масиву вузлів, що лишилися, за продуктивністю (FLOPS): $N_{FLOPS} = \{N_{FLOPS_j}\}$

Крок 6. Призначення вагових коефіцієнтів (балів) за продуктивність та енергоефективність кожному вузлу

Крок 7. Розміщення завдання $task_i$ *на вузол із найбільшим сумарним ваговим коефіцієнтом* [45].

Детальний опис кожного кроку алгоритму наведено далі.

Крок 0.

В процесі початкового налаштування кластера та його підготовки до роботи, згідно запропонованого підходу, необхідно виміряти залежності спожитої кожним сервером електроенергії від завантаженості його центрального вузла обробки. Ця залежність може бути виміряна із використанням навантажувального тесту (наприклад, представленого у джерелі [14]) та представлена у вигляді функції:$P = f(CPU)$,

де CPU – Central Processor Utilization – навантаженість центрального процесора (вузла обробки).

У даній роботі запропоновано проводити подальшу інтерполяцію функцій $P = f(CPU)$ поліномами ступеня n, що докладно описується у далі.

Після визначення залежностей $P = f(CPU)$, згідно запропонованого підходу, відбувається формування опису кожного вузла. До опису входять такі дані:

— Функція енергоспоживання сервера від CPU;

— Загальна продуктивність сервера;

— Кількість ядер центрального процесора сервера;

— Об’єм оперативної пам’яті сервера.

Приклад сформованого опису вузла наведено на рис. 2.7. (функція $P = f(CPU)$ подана у вигляді коефіцієнтів полінома ступеня 4).

Node #1	
P = f(CPU)	0 2,1 -16 35,1
FLOPS	35,7*10^9
k_cores	4
V_RAM	4*10^9

Рисунок 2.7 Приклад опису вузла сформований
у результаті виконання кроку 0

Крок 1.

У момент часу τ нове завдання (робота) job_i надходить до кластера. У момент часу τ_{k-1} (попередня оцінка) необхідно виміряти такі параметри кожного вузла N_j:

1. $\Delta V_{j_{avail}} = V_j - V_{j\_used} -$ доступний об'єм оперативної пам'яті $j -$ го вузла;

де V_{j_used} – оперативна пам'ять, зайнята на момент часу τ_{k-1};

2. $\Delta k_{core\,j_{avail}} = k_{core\,j} - k_{core\,j_{used}} -$ кількість ядер процесора, доступних на вузлі N_j;

де $k_{core\,j_{used}}$ – кількість ядер вузла N_j, що зайняті іншими завданнями на момент часу τ_{k-1};

3. $P_\Sigma = \sum_j P_j\,|\tau_{k-1} -$ сумарна потужність, що споживається кластером в момент часу τ_{k-1}

Крок 2.

Якщо доступний об'єм оперативної пам'яті менший за той, що вимагається для виконання завдання $task_{i,}$ ($\Delta V_{j_{avail}} \le V_{req_i}$), вузол N_j кластера виключається із розгляду доступних для обробки даного завдання вузлів. Аналогічно, якщо кількість незадіяних у роботі ядер вузла N_j менша, ніж цього вимагає завдання $task_{i,}$ вузол N_j виключається із розгляду доступних для обробки даного завдання вузлів.

У результаті таких операцій отримуємо масив $\{N_{avail\,j}\}$ теоретично доступних для розміщення завдання $task_i$ вузлів такий, що $\{N_{avail\,j}\} \subset \{N_j\}$.

Крок 3.

Для кожного j-го вузла обчислювального кластера залежність потужності споживання від навантаженості процесора є відомою функцією: $P_j = f_j(CPU_j)$. Цю залежність було виміряно і збережено при проведенні попередньої атестації кластера.

Припустивши, що робота job_i була призначена для виконання вузлу N_j, та знаючи залежність $P_j = f_j(CPU_j)$ для цього вузла, обчислимо теоретичне значення сумарної спожитої кластером потужності у разі розміщення завдання на обробку на j-й вузол:

$$P_{\Sigma_l}|\tau_k \& N_j = P_{\Sigma}|\tau_{k-1} + \Delta P_{j_i}|\tau_k$$

Проведемо такі обчислення для кожного вузла кластера.

Таким чином, маємо масив значень $P_{\Sigma} = \{P_{\Sigma_j}\}$, отриманий шляхом теоретичного розміщення роботи job_i на кожний з вузлів N_j.

Крок 4.

Проведемо сортування масиву доступних для розміщення завдання $task_i$ вузлів $\{N_{avail\,j}\}$ за їх теоретично розрахованим на кроці 3 вкладом у загальне сумарне енергоспоживання всього обчислювального кластера $\Delta P_{j_i}|\tau_k$. Нехай сортування буде також проведене в порядку спадання. В результаті отримаємо масив N_{avail_p}:

$$N_{avail_p} = \{N_{p_{j_{max}}}, \dots N_{p_j min}\}$$

Крок 5.

Проведемо сортування масиву доступних для розміщення завдання $task_i$ вузлів $\{N_{avail\,j}\}$ за їх доступною продуктивністю $flops_j$ у порядку спадання. Доступна продуктивність умовно визначається як:

$$flops_j = FLOPS(1 - \frac{CPU(\%)}{100}),$$

де $FLOPS$ – загальна продуктивність вузла обробки.

В результаті отримаємо масив $N_{flops_{avail}}$:

$$N_{avail_{flops}} = \{N_{flops_{j_{max}}}, \dots N_{flops_{jmin}}\}$$

Крок 6.

В залежності від позиції вузла із номером j у сортованому масиві $N_{avail_{flops}}$ та N_{avail_p}, він отримує два вагових коефіцієнта (дві оцінки): $mark_p$ і $mark_{flops}$ відповідно. При чому $mark_p$, $mark_{flops} = \overline{1,n}$, де n – кількість вузлів у кластері. $mark_p$ дорівнює позиції j-го вузла у сортованому масиві N_{avail_p}, $mark_{flops}$ – позиції цього вузла у сортованому масиві $N_{avail_{flops}}$ відповідно.

Крок 7.

Сумарна оцінка кожного вузла:

$$mark_{\Sigma_j} = mark_{flops_j} + mark_{p_j}$$

Вузол, для якого $mark_{\Sigma_j} = \max_j mark_{\sum j}$ вибирається для здійснення обробки завдання $task_i$.

Блок-схема для запропонованого алгоритму наведена на рис. 2.8. На рис. 2.8 пункти зображеної блок-схеми позначають такі дії:

Пункти 1-2 блок-схеми: До кластера надходить нове завдання *task_i*, що повинно бути розміщено на один з вузлів обробки. Вузли, що не мають достатньо ресурсів, виключаються з розгляду.

Пункти 3-4 блок-схеми: Проводиться сортування вузлів за продуктивністю та теоретичним внеском у загальне енергоспоживання кластера.

Пункт 5 блок-схеми: Кожен з вузлів кластера отримує бал (ваговий коефіцієнт) за продуктивність та енергоефективність. Підраховується сумарний бал за обома параметрами.

Пункт 6 блок-схеми: Завдання розміщується на вузол, що має найвищий сумарний бал.

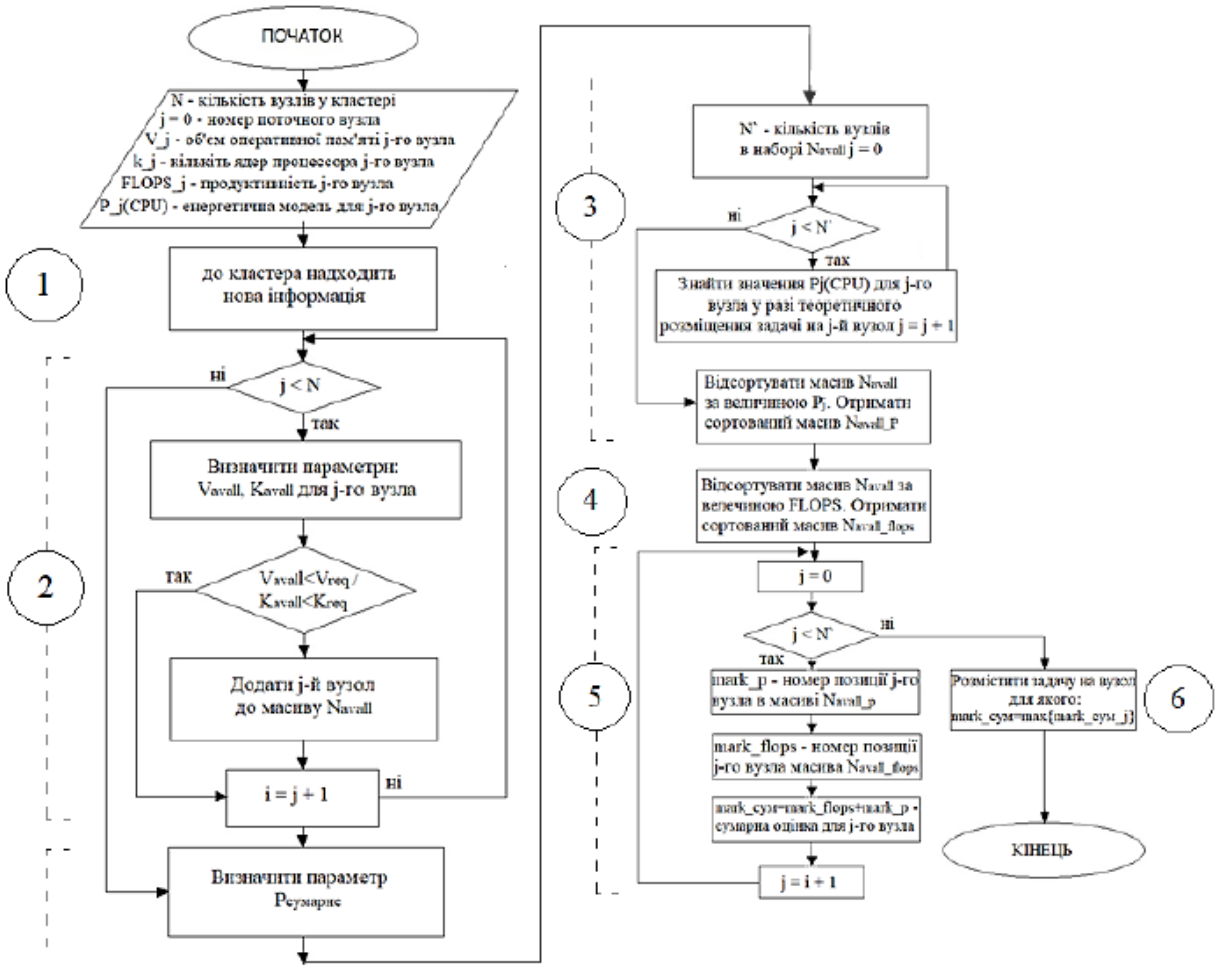


Рисунок 2.8 Блок-схема запропонованого алгоритму

2.4. Експериментальна перевірка роботи запропонованого підходу

Для перевірки роботи запропонованого підходу до підвищення енергоефективності обчислень у ЦОД було проведено натурний експеримент.

Метою експерименту було порівняння запропонованого підходу у формі алгоритму планування задач із існуючим широко використовуваним алгоритмом. Для порівняння було обрано алгоритм RoundRobin (RR). Принцип роботи цього алгоритму описано у пункті 1 розділу. Простота його реалізації, широке використання у реальних умовах ЦОД та наявність відкритого коду планувальника RR стали переважними якостями вибору даного алгоритму для порівняння.

План експерименту:

1. Побудувати гетерогенний серверний кластер.
2. Підготувати пакет завдань для перевірки роботи RR та запропонованого алгоритму (однаковий для обох випадків).
3. Отримати залежності між спожитою кожним вузлом серверного кластера потужністю та завантаженість процесора даного вузла.
4. Перевірити роботу алгоритму RR в умовах експерименту (проведення опорного експерименту).
5. Перевірити роботу запропонованого алгоритму в тих самих умовах.
6. Порівняти роботу алгоритмів за критеріями енергоефективності та продуктивності роботи кластера.

2.4.1. Постановка експерименту

Для проведення експерименту було використано 5 машин (серверів). Серед них 3 належали до одного типу (мали подібні фізичні характеристики), а 2 інші – до другого типу. Характеристики машин кожного типу наведені у табл. 2.7.

Таблиця 2.7 Фізичні характеристики серверів, використаних у експерименті

	Сервери типу 1 (x3)	**Сервери типу 2 (x2)**
Об'єм оперативної пам'яті	4 Gb	8 Gb
Тип та частота процесора	CPU Intel Core 2 Quad Q9400 2.66GHz	CPU Intel Core i5-43 3GHz
Продуктивність	35.7GFLOPS	40-50GFLOPS

На всі машини була встановлена операційна система Ubuntu OS. З цих 5 машин було утворено серверний кластер. Один вузол був обраний головним у кластері, саме на ньому розміщувався брокер (планувальник)

задач. Інші підлеглі вузли були рівноправними між собою та мали прямий зв’язок із головним. Топологію кластера зображено на рис. 2.9.

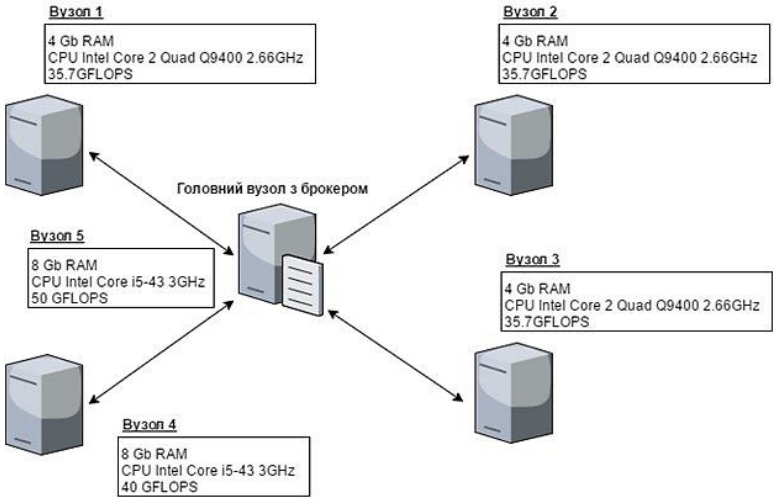


Рисунок 2.9 Топологія кластера використаного в експерименті

Для зняття характеристик $P_j = f_j(CPU_j)$ кожного вузла, було використано цифровий мультиметр Yokogawa WT210, що здатний вимірювати струм, напругу та миттєву потужність безпосередньо між джерелом живлення (розеткою) та навантаженням.

Для моделювання реальної роботи серверного кластера було обрано різні за характеристиками завдання, при цьому завдання були об'єднані у пакет завдань.

Для отримання характеристик $P_j = f_j(CPU_j)$ було використано навантажувальний тест (stress-test) [46]. Визначення характеристик проводилось шляхом послідовного підключення мультиметра у коло живлення кожного сервера. Вимірювання потужності споживання проводилось для 0, 25, 50, 75 та 100% завантаженості *CPU*.

При перевірці роботи алгоритму RR і запропонованого алгоритму були використані ідентичні пакети завдань. При цьому, в рамках пакету завдання були різними та являли собою підрахунок числа π з точністю до 1000, 5000, 10000, 25000, 50000, 75000, 100000 знаків після коми. Підрахунок виконувався 5 різними методами. Кількість знаків після коми, що виступала

параметром завдання, була випадковою величиною, розподіленою за нормальним законом.

Для порівняння алгоритмів було використано параметри продуктивності та енергоефективності. Для оцінки продуктивності було проаналізовано середній час обробки одного завдання. Ця величина є обернено пропорційною продуктивності кластера (формула 2.2):

$$P = m/t_{\sum}\ , \tag{2.2}$$

де m – кількість завдань, що підлягають обробці;

$t_{\sum}$ – час роботи кластера над обробкою набору з m завдань.

Тоді, середній час виконання одного завдання з набору:

$$t_{task\_i} = t_{\sum}/m \tag{2.3}$$

Виходячи з цього, можливо оперувати параметром «середній час обробки завдання» як обернено пропорційним до параметру продуктивності. Мінімізація параметру часу приведе до максимізації параметра продуктивності.

Для оцінки енергоефективності обробки завдання було обрано параметр загального сумарного енергоспоживання серверного кластера за час проведення експерименту у Вт.

2.4.2. Хід експерименту та обробка результатів

На початковому етапі експерименту, згідно запропонованого підходу, було проведено попередню атестацію серверів кластера. Експериментально

виміряні залежності $P_j = f_j(CPU_j)$ були представлені у табличному та графічному вигляді (рис. 2.10).

Для формування опису вузлів та розміщення описів на головному вузлі із планувальником необхідним є представлення функцій $P_j = f_j(CPU_j)$ в аналітичному вигляді. Для цього було обрано метод інтерполяції функцій поліномом ступеня n.

Інтерполяція має як практичне, так і теоретичне значення. На практиці часто виникає завдання про відновлення неперервної функції по її табличних значеннях, отриманих, наприклад, в ході будь–якого експерименту. Для обчислення багатьох функцій виявляється ефективним наближення їх поліномами або дрібно-раціональними функціями [47].

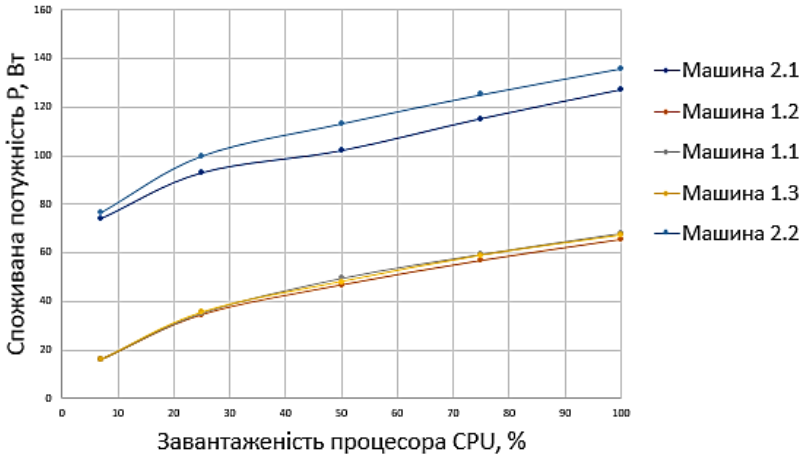


Рисунок 2.10 Експериментально отримані залежності $P_j = f_j(CPU_j)$ для всіх серверів кластера

Процес інтерполяції було автоматизовано завдяки використанню інструментів середовища MATLAB. Ступінь інтерполяції $n = 4$ було обрано емпіричним шляхом – використання нижчого ступеня не дає потрібної точності (графіки функцій отриманих експериментальним шляхом та інтерполяцією та не співпадають), а використання вищих ступенів є надлишковим і не дає помітного наближення до початкового вигляду функцій.

В результаті проведення етапу попередньої атестації для серверів кластера було отримано їх описи (рис. 2.11).

Наступним етапом проведення експерименту стала перевірка роботи алгоритму RR. Для цього код планувальника RR було завантажено на центральний вузол серверного кластера. Отримані результати представлено у табл. 2.8.

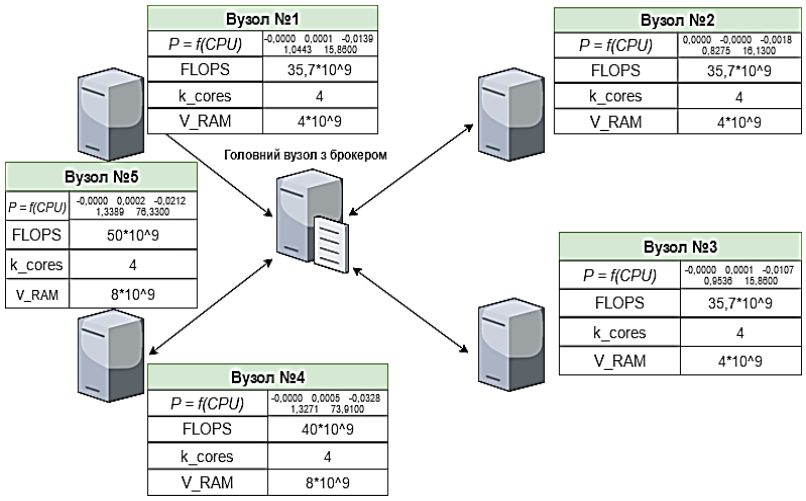


Рисунок 2.11 Описи вузлів отримані в ході проведення етапу попередньої атестації

Після завершення перевірки роботи алгоритму RR на центральний вузол кластера було завантажено код планувальника, реалізований згідно запропонованого алгоритму. Алгоритм було реалізовано на мові програмування Python, згідно блок-схеми. Результати експериментальної перевірки запропонованого підходу та алгоритму RR наведені у табл. 2.8.

Таблиця 2.8 Результати експериментального порівняння запропонованого підходу та алгоритму Round Robin для кластеру із 5 вузлів

Алгоритм	**Середній час обробки завдання,** *с*	**Сумарна спожита потужність,** *Вт*	**Виграш у порівнянні із Round Robin,** *%*	
			За часом	**За спожитою енергією**
Round Robin	14.835	376.88	-	-
Запропонований підхід	13.324	365.8	10,2%	3%

2.5. Аналіз отриманих результатів

Згідно результатів, наведених у табл. 2.8, запропонований алгоритм дозволяє отримати виграш у порівнянні із алгоритмом RR за параметром продуктивності – 10,2%, за параметром енергоефективності – 3%.

Отриманий виграш є незначним. Виграш за параметром енергоефективності майже підпадає під класифікацію статистичної помилки (при визначеній похибці у 2%).

Такий незначний виграш пояснюється тим, що вузли серверного кластера, використаного в експерименті, за своїми характеристиками належали до 2–х груп, причому в рамках кожної групи характеристики були однаковими. Крім того, залежності $P = f(CPU)$ для досліджуваного кластера також були близькими. Отже, можна припустити, що гетерогенність такого кластера була надто низькою, що й призвело до низької ефективності запропонованого підходу. Для перевірки даної гіпотези необхідно здійснити перевірку роботи запропонованого підходу для серверних кластерів, що містять більше вузлів, які є водночас більш різноманітними за своїми характеристиками.

2.6. Оцінка ефективності алгоритму шляхом моделювання

Для перевірки ефективності підходу у масштабному гетерогенному кластері було використано підхід імітаційного моделювання. Першим кроком було доведено адекватність моделі до об’єкту дослідження шляхом моделювання кластера з 5 вузлів. Після чого було створено модель з 20 вузлів, що має топологію, зображену на рис. 2.12.

При цьому, ефективність запропонованого підходу було порівняно із широким колом широко використовуваних алгоритмів розподілу задач.

Алгоритми, використані в ході моделювання, та їх короткі характеристики наведені у табл. 2.9.

Моделювання імітує роботу серверного кластера протягом доби (із середньостатистичним розподілом навантаження щогодини). Було використано набір завдань, аналогічний до випадку моделювання кластера з 5 вузлів. Інтервали часу між надходженням завдань – випадкова величина розподілена за нормальним законом. Результати моделювання зображено у табл.2.10.

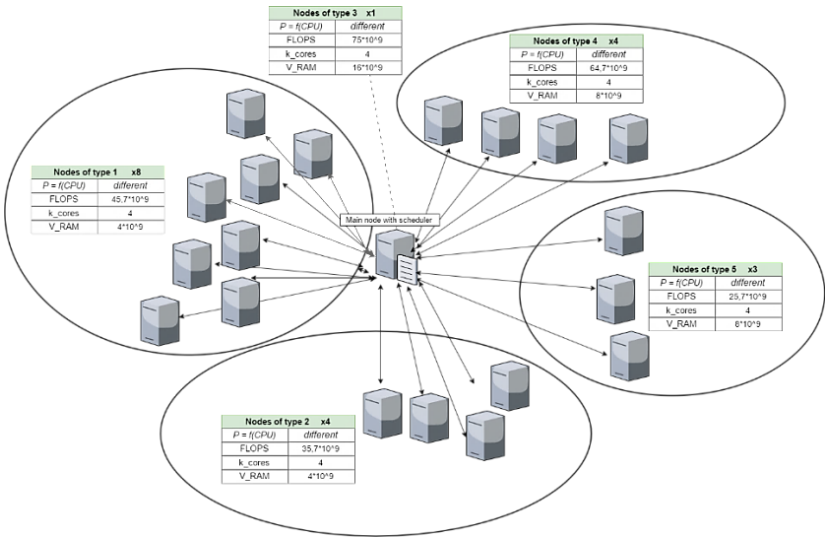


Рисунок 2.12 Топологія кластера з 20 вузлів, використана у процесі моделювання

Таблиця 2.9 Алгоритми, що були використані в ході моделювання для порівняння із запропонованим

Алгоритм	**Характеристика**	**Головна ідея**	**Посилання**
RoundRobin (RR)	Простий, широко застосовний	Розміщує завдання на вузли по черзі	https://en.wikipedia.org/wiki/Round-robin_scheduling
FIFO(First Available)	Простий	Розміщує завдання на перший доступний вузол	https://en.wikipedia.org/wiki/First-come,_first-served
Least Connections	Широко застосовний	Розміщує завдання на вузол, що завантажений найменше	https://www.citrix.com/blogs/2010/09/02/load-balancing-least-connections/
Weighted RR	Покращений RR, широко застосовний	Деякі вузли більш пріоритетні для завантаження, тому вони мають більший ваговий коефіцієнт. Завдання розміщуються пропорційно коефіцієнтам, але почергово, як і для RR	http://www.brocade.com/content/html/en/configuration-guide/nos-700-l2guide/GUID-B75BD370-B251-45A3-B3FD-87F54E35DC6E.html
«Алгоритм Р»	Енерго-ефективний	Розміщує завдання на вузол з найменшою чергою. Якщо на цьому вузлі недостатньо ресурсів – на найбільш енергофективний вузол	http://cyberleninka.ru/article/n/energoeffektivnye-vychisleniya-dlya-gruppy-klasterov.pdf
The Most-Efficient-Server first (MES1)	Енерго-ефективний	Розміщує завдання на найбільш енергоефективні вузли. Якщо ці вузли зайняті – використовується RR для розміщення на інші вузли почергово	http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6679892&tag=1

Таблиця 2.10 Результати моделювання роботи кластера з 20 вузлів протягом доби

Алгоритм	**Середній час обробки завдання,** *с*	**Сумарна спожита потужність за годину,** *Вт*	**Програш у порівнянні із запропонованим алгоритмом,** *%*	
			За часом	**За спожитою енергією**
Запропонований підхід	10,880	1281,167	-	-
FIFO	10,996	1324,836	1,06	3,41
RR	16,221	1396,948	49,09	9,04
RR Weighted	14,019	1359,887	28,85	6,14
Least Connections	13,474	1357,448	23,84	5,95
MES1	13,857	1309,055	27,36	2,18
«Алгоритм Р»	15,470	1381,166	42,19	7,81

Таким чином, при дослідженні роботи кластера протягом доби із реальним розподілом навантаження було виявлено, що запропонований підхід дає виграш за обома параметрами у порівнянні із усіма алгоритмами. Найбільший виграш у 49,03% дає підхід за параметром продуктивності у порівнянні із алгоритмом RR. Виграш запропонованого експерименту за енергоефективністю близький до існуючих енергоефективних алгоритмів MES1 та «Алгоритм Р», проте, для випадку добової роботи, виграш за параметром продуктивності був значно більшим, ніж у зазначених алгоритмів.

Отримані результати дають право вважати, що ефективність запропонованого підходу зростає зі збільшенням кількості вузлів у кластері, їх гетерогенності та часу моделювання. Це дає підстави стверджувати, що запропонований підхід доцільно застосовувати у середніх та великих гетерогенних кластерах при тривалому часі їх роботи [48].

2.7. Визначення оптимальних значень вагових коефіцієнтів алгоритму в залежності від ступеню гетерогенності серверного кластеру.

Запропонований алгоритм балансування навантаження базується на визначені відмінностей у продуктивності та енергоефективності між окремими вузлами кластеру. Алгоритм складає таблицю коефіцієнтів, яка містить в собі оцінки ефективності, складені відносно найкращих показників в обох категоріях. Характерною відмінністю запропонованого алгоритму є використання вагових коефіцієнтів $w_{productivity}$ та $w_{energy\_efficiency}$, при чому

$$w_{productivity} + w_{energy\_efficiency} = 1$$

Ці коефіцієнти позначають вагу критеріїв продуктивності та енергоефективності відповідно. У проведених в пунктах 2.6 та 2.7 дослідженнях коефіцієнти були обрані наступним чином:

$$w_{productivity} = w_{energy\_efficiency} = 0{,}5$$

З цього випливає, що вага критеріїв продуктивності та енергоефективності рівна і запропонований алгоритм обиратиме для обробки вхідних даних вузол, найбільш збалансований по відношенню швидкість обробки/витрачена електроенернія. Варто зазначити, що при проведенні оцінки енергоефективності використовується визначена раніше оцінка продуктивності. Наприклад, є два вузли $node_1$ та $node_2$, продуктивність яких складає 20 та 40 GFLOPS, а електроспоживання при навантаженні рівне 150 і 250 Вт/Год відповідно. Нехай на вхід системи поступає дві абсолютно однакові задачі, які розподілені між двома вузлами системи. При цьому кожна задача має такий обчислювальний об’єм, що вузлом $node_1$ вона буде виконана за 2 години, а вузлом $node_2$ за 1 годину. Неважко визначити, що вузлом $node_1$ за цей період часу буде спожито $150 * 2 = 300$ Вт/Год, а вузлом $node_2$ -

$250 * 1 = 250$ Вт/Год. Таким чином, вузол $node_2$ буде кращим за вузол $node_1$ як за показником продуктивності, так і за показником енергоефективності.

Для дослідження ефективності запропонованого алгоритму балансування навантаження раніше створена в середовищі Matlab імітаційна модель була доповнена функціоналом визначення ступеня гетерогенності кластера шляхом оцінки середньоквадратичного відхилення показників продуктивності вузлів кластера. Всі можливі конфігурації центрального процесору вузла (понад 2700 значень) були винесені в окрему таблицю, в якій зазначаються показники:

1. TDP (thermal design power) - величина, яка показує, на відведення якої теплової енергії може бути розрахована система охолодження комп'ютера, процесора або іншого пристрою [48]. Наприклад, якщо система охолодження процесора розрахована на TDP 30 Вт, вона повинна бути в змозі відвести 30 Вт тепла при деяких заданих «нормальних умовах». Максимально можливе енергоспоживання процесора в нормальних умовах складає приблизно 75% від паспортного значення TDP.
2. Індекс продуктивності – паспортна величина, яка визначає максимальну продуктивність процесора в кількості операцій з плаваючою точкою за секунду (FLOPS)
3. Кількість ядер(потоків)

При цьому конфігурації кожного вузла в даному випадку генеруються автоматично за законом нормального розподілення з різними показниками σ. Кількість вузлів в кластері було обрано рівним 100 – розмір середнього за об’ємом ЦОД, який теоретично дозволить оброблювати той об’єм трафіка, який генерується абонентами великого міста. Навантаження, яке буде використовуватися, як вхідні дані моделі було обчислено в пункті 2.2 цього дослідження.

В зв’язку із значним обчислювальним ускладненням моделі, було зроблено декілька допущень:

1. Для порівняння використовується лише 2 алгоритми балансування навантаження: Round Robin та запропонований PCPB.

2. Визначення приросту споживання електроенергії при навантажені замість повного споживання електроенергії всім вузлом.

3. Величина енергоспоживання ЦП на відміну від величини продуктивності не підпорядковується використаному в моделі нормальному розподілу. Таким чином, може бути дві конфігурації, однакові за продуктивністю, але різні за енергоспоживанням.

4. В моделі імітується навантаження за годину найбільшого навантаження замість доби. Набор задач при цьому однаковий для обох алгоритмів балансування навантаження.

5. Всі задачі, що надходять до системи є однопоточними (з послідовними операціями) та можуть виконуватися лише на одному ядрі (потоці)

Моделювання проводиться для 10 конфігурацій ЦОД, сортованих за ступіню гетерогенності – від найбільш різноманітних до практично однакових конфігурацій. При цьому вимірювання проводиться 20 разів з різними параметрами $w_{productivity}$ та $w_{energy\_efficiency}$ (з кроком 0.05). Функція ймовірності та функція розподілу ймовірності продуктивності процесора зазначено на рис. 2.13 та 2.14:

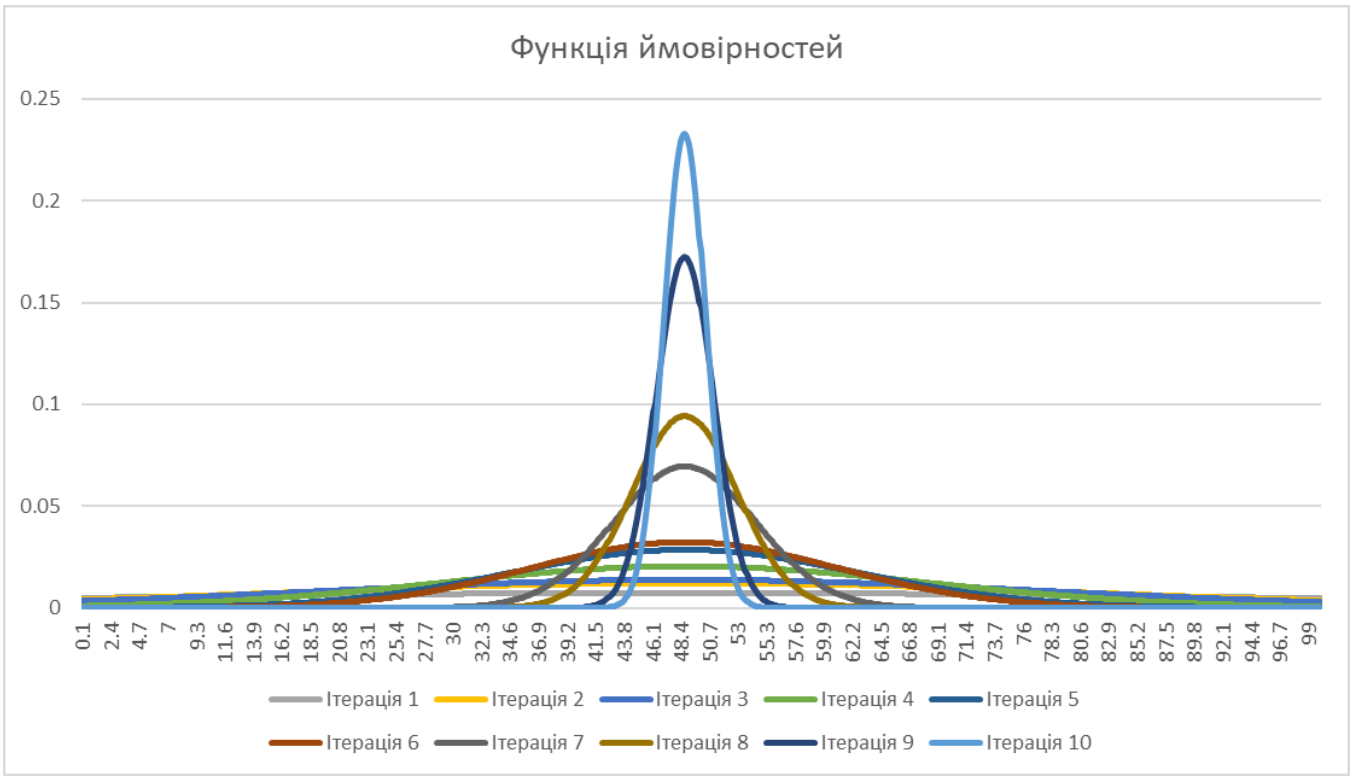


Рисунок 2.13 Функція ймовірностей серед продуктивності всіх ЦП кластеру.

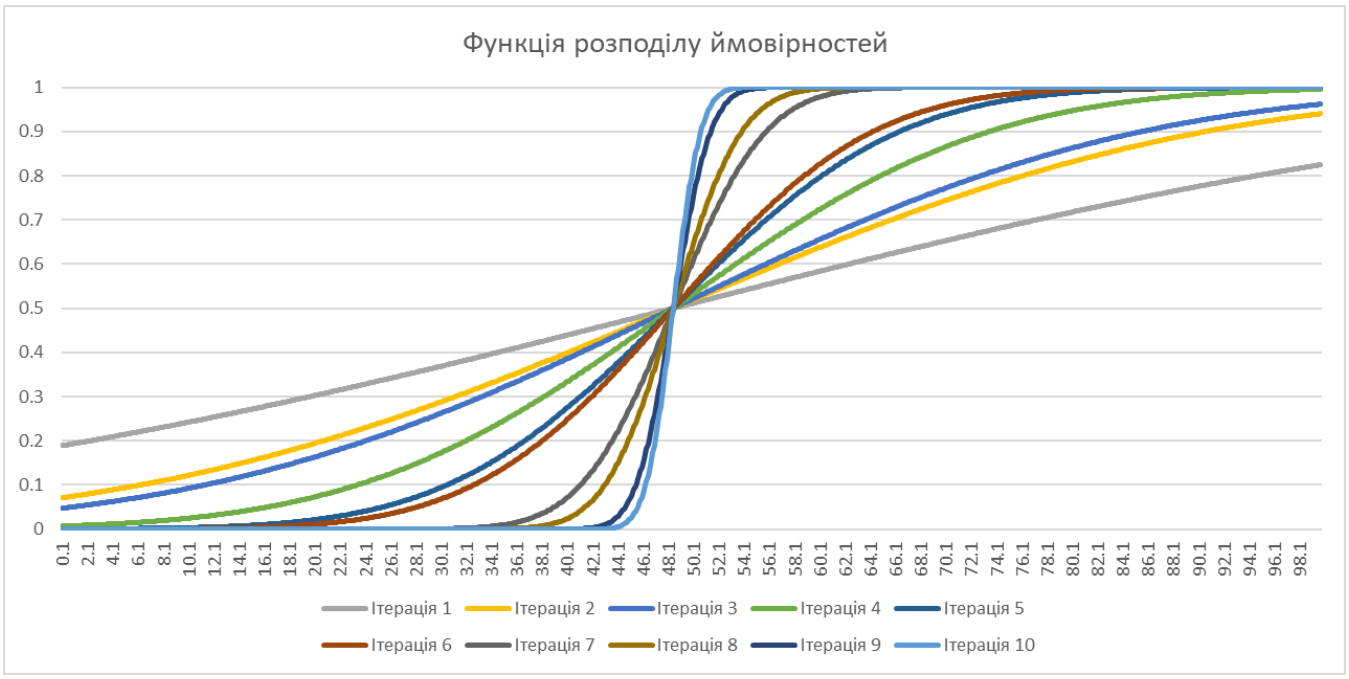


Рисунок 2.14 Функція розподілу ймовірностей серед продуктивності всіх ЦП кластеру.

На рис. 2.13 та 2.14 можна побачити, що 10 обраних значень середньоквадратичного відхилення дозволяють згенерувати 10 серверних кластерів, які значно відрізняються між собою за ступеню гетерогенності – від практично хаотичних кластерів, кожен вузол яких є унікальним до детермінованих кластерів, які складаються з вузлів 2-3 типів.

Результати моделювання наведені на рис. 2.15 – 2.29.

Рисунок 2.15 Результати ітерації №1 ($\sigma = 1.12\ \mu$)

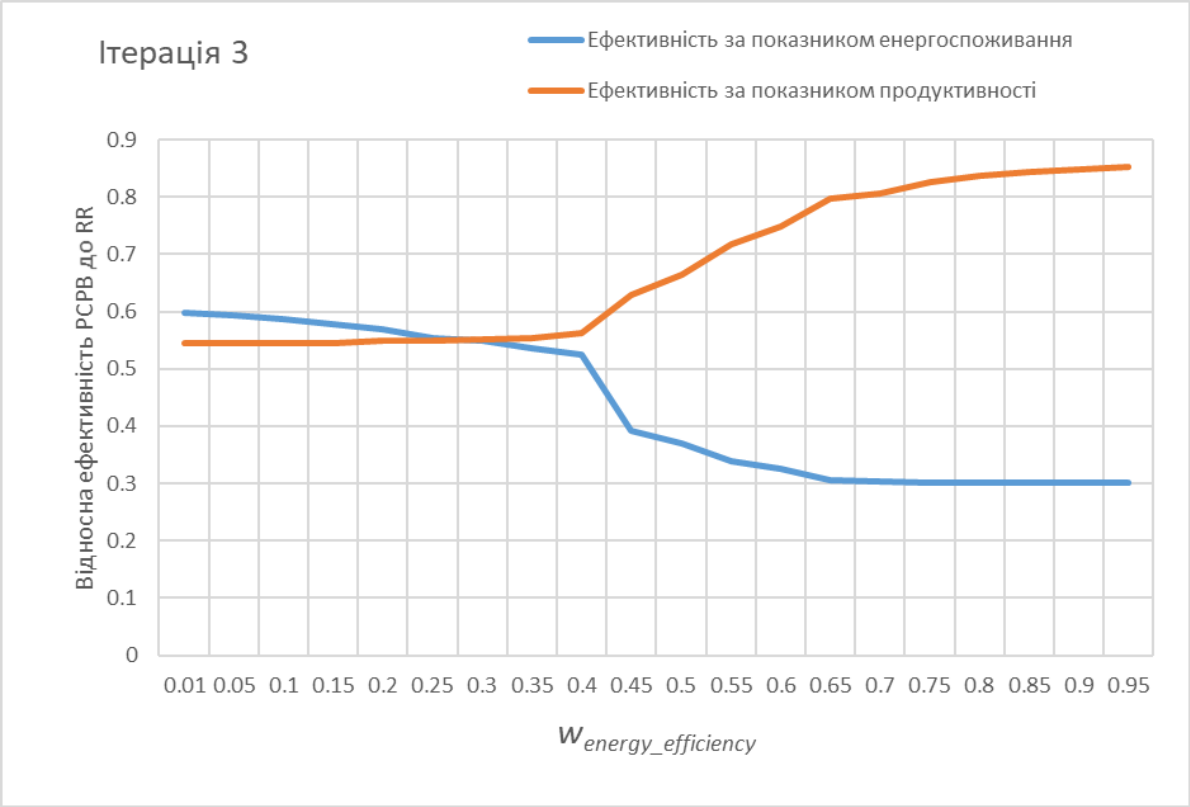


Рисунок 2.16 Результати ітерації №3 ($\sigma = 0.59\ \mu$)

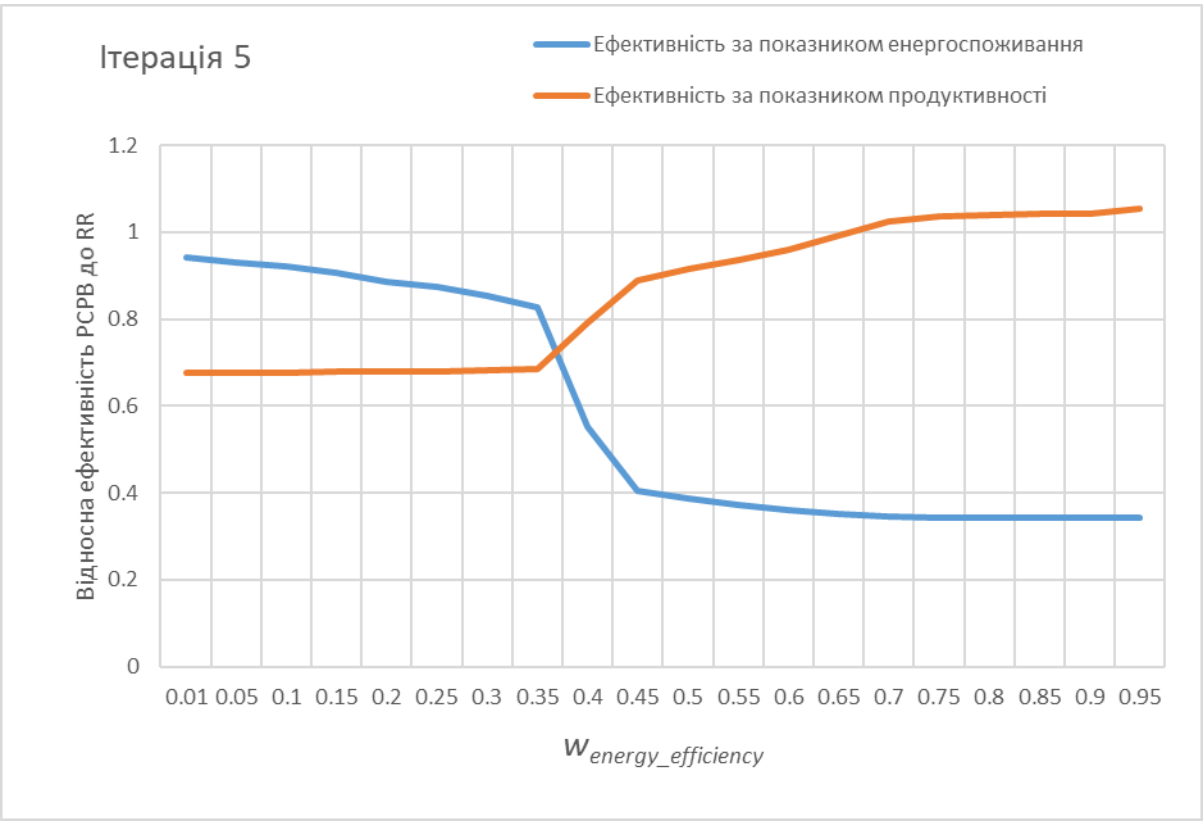


Рисунок. 2.17 Результати ітерації №5 ($\sigma = 0.28\ \mu$)

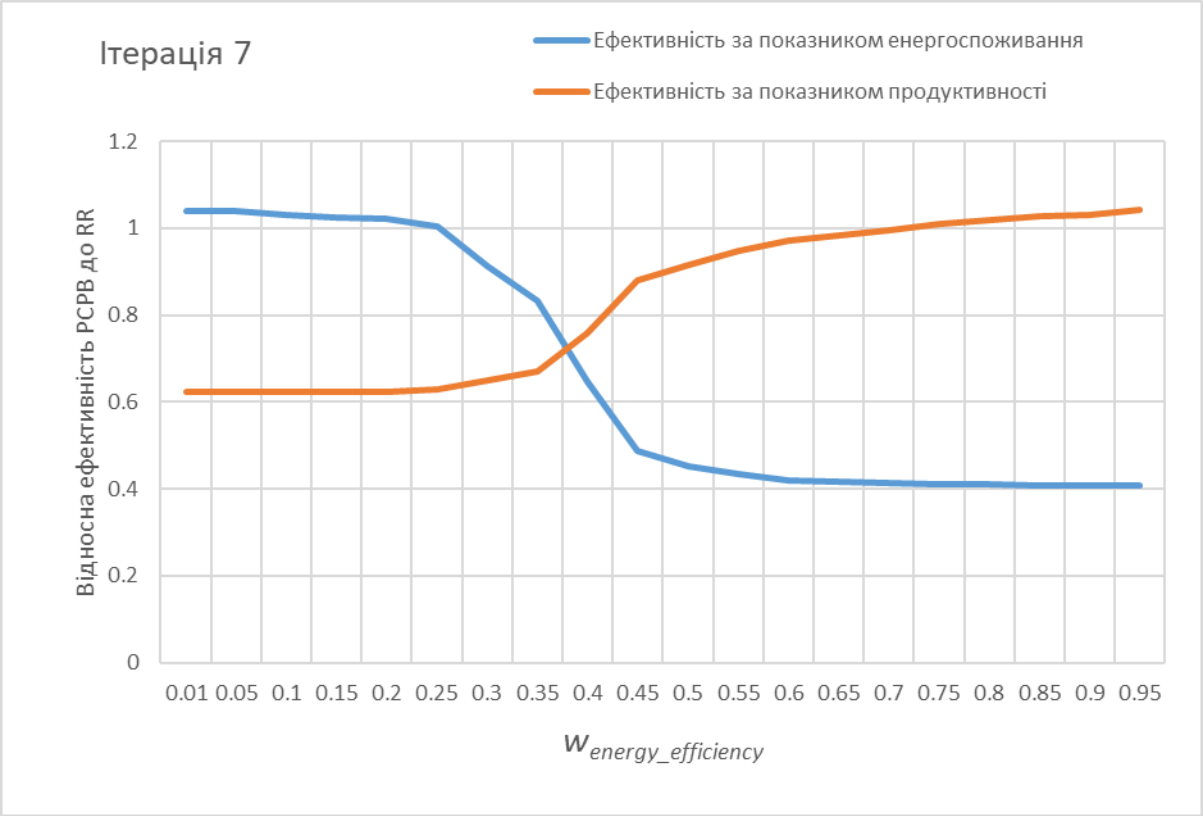


Рисунок 2.18 Результати ітерації №7 ($\sigma = 0.11\ \mu$)

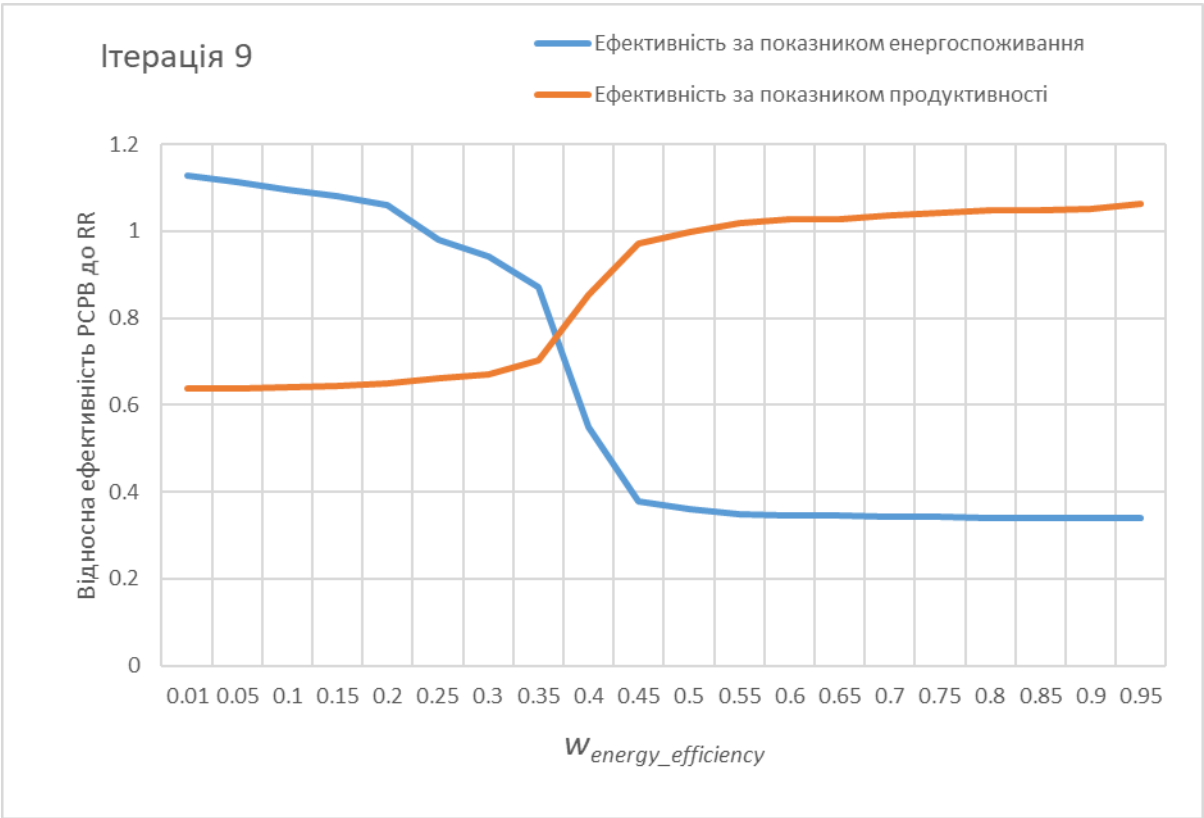


Рисунок 2.19 Результати ітерації №9 ($\sigma = 0.04\ \mu$)

З рис. 2.15-2.16 видно, що при значному ступені гетерогенності серверного кластера, запропонований підхід при будь-якому налаштуванні параметрів $w_{productivity}$ та $w_{energy\_efficiency}$ є більш ефективним за Round Robin як за показником продуктивності так і за показником енергоефективності.

З рис. 2.17-2.219 можна зробити висновок, що при зменшенні ступені гетерогенності кластеру зменшується вікно налаштувань параметрів $w_{productivity}$ та $w_{energy\_efficiency}$ при якому запропонований підхід має кращі результати в обох показниках. При необхідності подальшого покращення одного з параметрів необхідно жертвувати іншим.

Особливість запропонованого підходу – змінні вагові коефіцієнти $w_{productivity}$ та $w_{energy\_efficiency}$ роблять РСРВ універсальним алгоритмом балансування навантаження, який при точних налаштуваннях, в залежності від вимог, може виступати як алгоритм, який незначно покращує показники як продуктивності так і енергоефективності, так і як алгоритм, який дозволить домогтися значного покращення по будь-якому з двох показників за рахунок іншого.

Таким чином, РСРВ дозволяє значно покращити на 40-60% експлуатаційні характеристики великого гетерогенного кластеру при будь-яких налаштуваннях. У випадку незначного ступеню гетерогенності серверного кластеру можна досягти покращення в 25% з обох показників при $w_{productivity}$ *=60-65%* та $w_{energy\_efficiency}$*=35-40%.*

2.8. Огляд можливих модифікацій для підвищення ефективності запропонованого підходу та рекомендації щодо застосування підходу у обчислювальному середовищі

Для підвищення ефективності алгоритму запропоновано 2 основних напрями його модифікації. Перший полягає у використанні підходів масштабування додатково до застосування запропонованого алгоритму,

другий – у врахуванні типів та характеристик вхідних завдань при їх розподілі.

2.8.1. Модифікація з використанням підходів масштабування

Підхід масштабування передбачає тимчасове переведення частини вузлів кластера у режим сну у випадку, якщо кількість вхідних завдань незначна і може бути оброблена без участі даних вузлів [50]. Використання даного підходу дозволить отримати значний виграш за параметром енергоефективності, адже споживання вузлів серверного кластера у режимі сну мінімальне.

Для реалізації даної модифікації, постає завдання експериментального або аналітичного визначення порогових кількостей завдань у черзі на обробку, при яких повинно бути здійснено переведення одного вузла кластера у режим сну та навпаки. При цьому у режим сну повинні бути переведені машини із «найгіршими» характеристиками енергоспоживання.

Крім того, постає проблема визначення порогу перенавантаження активних вузлів, коли необхідним стає виведення нового вузла з режиму сну та переведення його в активний режим. Ці проблеми тісно пов’язані з проблемами консолідації віртуальних машин при використанні підходів масштабування.

2.8.2. Модифікація з використанням типізації задач

Для збільшення продуктивності серверного кластера доцільним є використання підходу типізації завдань додатково до запропонованого алгоритму. Цей підхід передбачає розбиття вхідного потоку завдань на умовно «складні» та «прості» в залежності від ресурсів, яких кожне з них потребує [51].

Після проведення типізації підхід передбачає розподіл «простих» завдань на вузли, що мають меншу продуктивність, а «складних» – відповідно, більшу.

Для реалізації цього підходу необхідно експериментальним або аналітичним шляхом визначити значення параметрів завдань, за якими їх можна віднести до одного із типів («складні» чи «прості»). Для кожного окремого випадку такий поділ має бути налаштований індивідуально в залежності від характеристик завдань.

Для отримання загального уявлення про ефективність запропонованих модифікацій, було проведено імітаційне моделювання у середовищі MATLAB. При цьому пороги для ввімкнення та вимкнення вузлів кластера та характеристики завдання для їх типізації було підібрано методом послідовного наближення.

Для модифікації із застосуванням підходів масштабування, шляхом зміни значення порогів вимкнення та ввімкнення серверів у діапазоні від 5 завдань у черзі до 30 завдань у черзі із кроком 5, було визначено, що для конкретних умов моделі (20 вузлів кластера із характеристиками, зазначеними на рис. 6.7), оптимальним є значення 10 і менше завдань у черзі для переведення у режим сну найменш енергоефективного сервера, та 25 і більше завдань у черзі для виведення одного сервера із режиму сну. Для модифікації із використанням типізації завдань, аналогічно, було визначено поріг у 350 GFLOPS. Завдання з об’ємом меншим цього порогу, вважалися «простими» в рамках даного випадку моделювання. Більш обчислювально–об’ємні задачі вважалися «складними».

Моделювання проводилось для кластера з 20 вузлів за умов, описаних у пункті 2.3.1. Результати проведеного моделювання представлені у табл. 2.11.

Як видно із табл. 2.11, для даних умов моделювання застосування модифікації із використанням підходу типізації завдань не дає відчутного результату. Дослідження умов, за яких дане доповнення до запропонованого підходу дозволить отримати виграш, буде досліджено у майбутньому. При

застосуванні підходу масштабування стає можливим отримання значного виграшу за параметром енергоефективності. Для умов моделювання відносний виграш склав 33,59% [52].

На основі аналізу отриманих результатів можна зробити висновок, що подальшого дослідження вимагають обидва підходи модифікації. Серед них, більш перспективним є підхід із застосуванням масштабування.

Таблиця 2.11 Результати моделювання роботи запропонованого підходу із можливими модифікаціями

Алгоритм	**Середній час обробки задачі,** *с*	**Сумарна спожита потужність,** *Вт*	**Виграш у порівнянні із запропонованим алгоритмом,** *%*	
			За часом	**За спожитою енергією**
Запропонований підхід	10,880	1281,167	0,00	0,00
Модифікація 1	10,885	1284,339	0,05	0,25
Модифікація 2	10,893	850,838	0,12	33,59

Висновки

1. Проведено аналіз існуючих підходів до підвищення енергоефективності обчислень у центрі обробки даних, який показав наявність низки методів, що мають на меті зниження енергоспоживання під час обробки даних, але не враховують одночасно параметр продуктивності обчислень.

2. Виділено клас методів підвищення енергоефективності обчислень, а саме енергоефективне планування завдань, у рамках якого доцільним є внесення змін з метою покращення показників енергоефективності та продуктивності обчислень.

3. Запропоновано підхід щодо підвищення енергоефективності обчислень для серверного кластера як інформаційно-телекомунікаційної

одиниці інфраструктури ЦОД, який відрізняється одночасним врахуванням параметрів енергоефективності та продуктивності при розподілі завдань.

4. Розроблено алгоритм підвищення енергоефективності обчислень на основі запропонованого підходу, який складається з етапу попередньої індивідуальної атестації серверного кластера та етапу енергоефективного розподілу завдань. Врахування індивідуальних математичних залежностей енергоспоживання серверів з урахуванням їх завантаженості є основною відмінною рисою запропонованого підходу.

5. Перевірено роботу запропонованого підходу в залежності від ступеню гетерогенності серверного кластеру та виставлених вагових коефіцієнтів показників продуктивності та енергоефективності. Отримані результати показують, що ефективність роботи алгоритму значно збільшується у випадку гетерогенного кластеру. Доведено, що оптимальними налаштуванням алгоритму, які дадуть найбільшу збалансовану ефективність є

$w_{productivity}$ *=60-65%* та $w_{energy\_efficiency}$*=35-40%.*

6. Перевірено роботу запропонованого підходу експериментально та шляхом імітаційного моделювання. Проведено аналіз отриманих результатів, та визначено, що підхід проявляє більшу ефективність – до 25% за показниками продуктивності та енергоефективності при оптимальних вагових коефіцієнтах – для великих гетерогенних кластерів.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЕНЕРГОЕФЕКТИВНОГО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В СЕРВЕРНОМУ КЛАСТЕРІ

Для перевірки адекватності розробленої моделі необхідно провести експеримент на реальному обладнанні, описаному в пункті 2.4. Для цього необхідно розробити програмне забезпечення, яке дозволить проводити балансування вхідного навантаження між вузлами, які підключені до вузла - брокера.

3.1. Архітектура програмного забезпечення

Створене рішення складається з таких компонентів:

1. *Main.py* – головний виконувальний файл, з ініціалізації якого починається виконання програми. Містить функції алгоритму RAFT, в залежності від результатів роботи якого може надалі виконувати функції *leader( )* , яка виконує прослухання зовнішнього інтерфейсу системи та додає/видаляє вузли за вимогою, викликає функцію брокера для отриманої задачі і перенаправляє її на визначений вузол. Окрім того, лідер надсилає всім вузлам повідомлення heartbeat (серцебиття) метою якого є визначення стану всіх вузлів кластеру. У випадку виходу з ладу будь-якого вузла лідер дізнається про це шляхом повідомлення heartbeat і виключає цей вузол з кандидатів на перенаправлення обчислювальних задач. В іншому випадку вузол виконує функцію *cell( )* , єдиною задачею якої є прослуховування інтерфейсу з лідером та виконання отриманої від нього задачі. Незалежно від виконуваної функції, кожен вузол має можливість провести атестацію, приблизно визначивши власну продуктивність та енергоспроживання.

2. *Attestation.py* – файл, основною задачею якого є проведення атестації вузла. Для цього він містить бібліотеки, які можуть отримувати системну інформацію про обладнання комп’ютера (модель ЦП, відеокарти,

жорсткого диску, тощо) та його власне системне навантаження. В процесі виконання атестації викликає файл *Collect_info.py*. Результат роботи цього файла зображений на рис. 3.1.

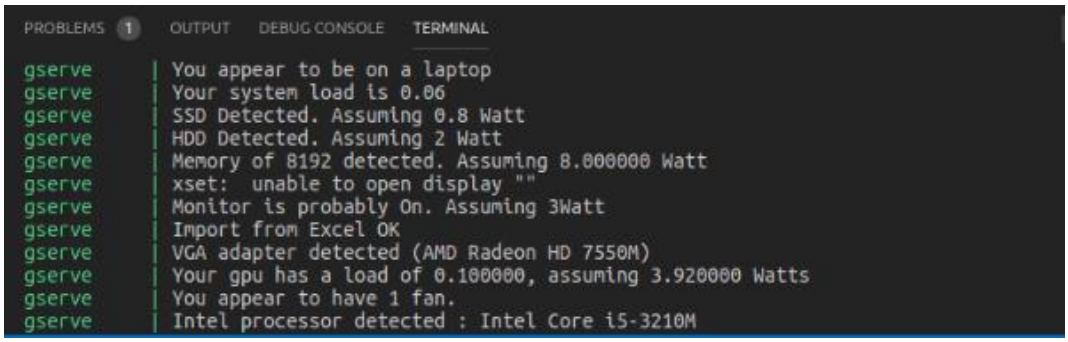


Рисунок 3.1 Результат роботи файла Attestation.py на ПК

Точність створеної програмної функції атестації складає приблизно 80%, оскільки вона, як і її аналоги, базується на паспортних технічних показниках обладнання. Точність була перевірена експериментальним шляхом в п.2.4 цього дослідження. Для більш точної роботи алгоритму можна використовувати дані з фізичного вимірювача потужності.

3. *Collect_info.py* – файл, який містить бібліотеки для роботи з мережею інтернет (отримання характеристик обладнання), та Excel (зберігання локальної копії отриманої інформації у вигляді таблиць).

4. *Broker.py* – файл, який містить програмний код балансування навантаження. Результатом його роботи є номер вузла, на який треба перенаправити задачу. Може бути викликаний лише лідером.

5. *Worker.py* – файл, який містить програмний код обробника задач. Є доступним до редагування – користувач може записати в ньому виконання будь-яких математичних, логічних та інших операцій, а також виклики інших програм, тощо. Файл може бути викликаний як лідером, так і послідовником.

Взаємодія компонентів при ініціалізації кластера показана на рис. 3.2

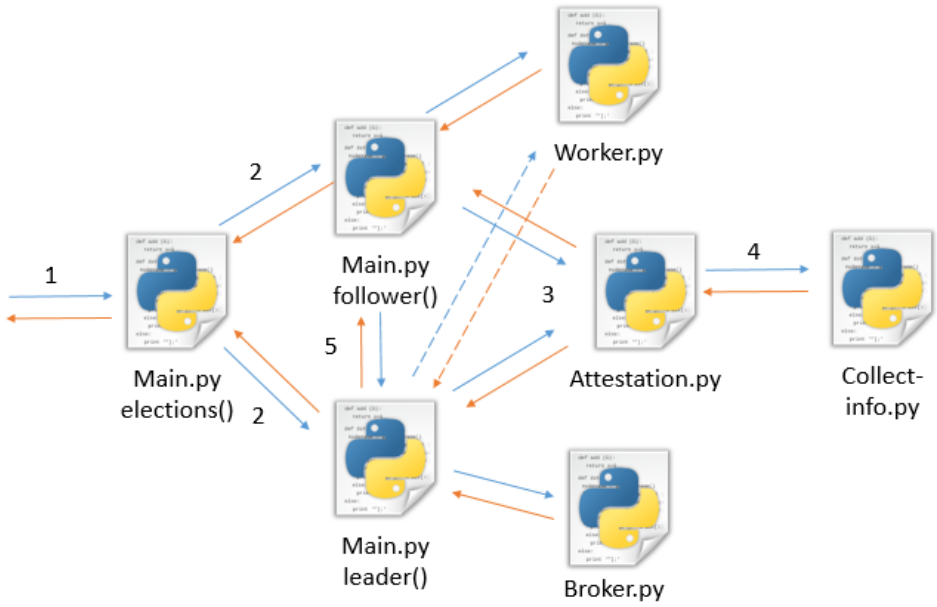


Рисунок 3.2 Процес ініціалізації кластеру в ПЗ

На рис. 3.2. відображений процес ініціалізації вузла в створюваному логічному серверному кластері. Цифрові позначки показують етапи ініціалізації в хронологічному порядку:

1. Точка входу в програму (початок роботи ПЗ). На цьому етапі передається назва кластеру (яка повинна бути унікальною) до якого вузол має приєднатися.
2. Після підключення до кластеру вузол має визначити свою роль: якщо в кластері немає лідера, то проводиться процедура голосування згідно алгоритму RAFT, в іншому випадку новий вузол залишається послідовником.
3. Кожен вузол проводить локальну атестацію обладнання.
4. В процесі атестації вузол використовує базу даних обладнання та створює локальну копію БД, в яку потім помістить дані атестації всіх інших вузлів кластеру. Це робиться з метою пришвидшення реконфігурації кластеру у випадку виходу з ладу лідера. В цьому випадку не буде необхідності проводити повторну атестацію кластеру.

5. Послідовник з лідером обмінюються інформацією, зазначеною в п.4

Процес взаємодії компонентів при надходженні задачі буде показаний в наступному підрозділі на прикладі простого серверного кластеру, який складається з двох вузлів – лідера та послідовника.

3.2. Проведення експериментального дослідження роботи запропонованого алгоритму

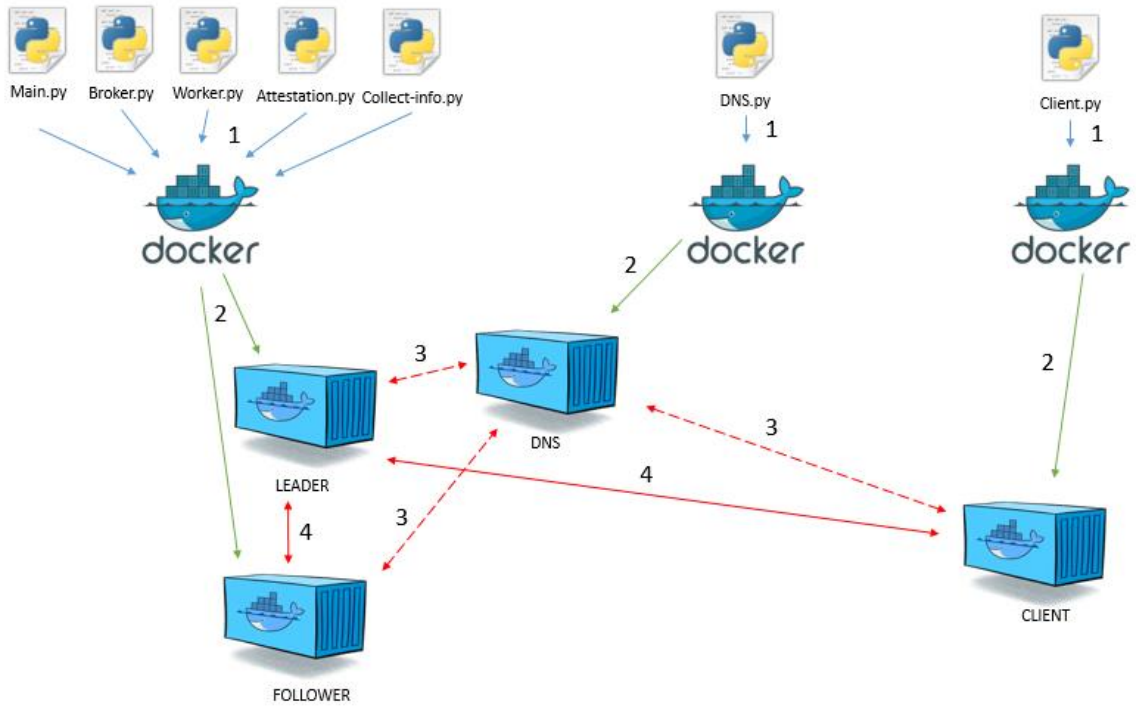
Першим етапом є перевірка коректності функціонування створеної програми на локальному комп’ютері. Для цього треба створити ще 2 файли:

1. *Client.py* – файл, який імітує роботу клієнта і генерує вхідне навантаження на серверний кластер. В якості навантаження обрано обчислення факторіалу певного числа – простий запис (необхідно передавати лише одне число) та змінна складність обчислення, яка при великому вхідному значенні є досить високою.

2. *DNS.py* – файл, який імітує роботу DNS (Domain Name Service) сервера і допомагає встановити зв’язок між послідовниками до обирання лідера.

При цьому буде створено 4 docker-контейнери: один гратиме роль клієнта-генератора навантаження, один гратиме роль DNS сервера, ще два будуть серверним кластером (1 лідер, 1 послідовник).

Архітектура тестового кластеру матиме вигляд, представлений на рис.



3.3:

Рисунок 3.3 Архітектура тестового кластеру

На рис.3.3. можна побачити як власне архітектуру кластеру, взаємозв’язок між компонентами (цифрами 3 позначені зв’язки для передачі службової інформації, цифрами 4 – інформації користувача: вхідні задачі від клієнта та результат обчислення), так і процес інсталяції кластеру: цифрами 1 позначено роботу технології Dockerfile, яка створює образ з інстальованими операційною системою UNIX, бібліотек Python, та завантажених виконавчих файлів створеного ПЗ. Цифрою 2 позначено роботу технологиї Docker-compose, яка дозволяє створити будь-яку кількість однотипних контейнерів з одного Dockerfile, та встановити взаємодію між створюваними контейнерами.

Після процесу ініціалізації кластеру з використанням DNS серверу до системи надходить клієнт з певним визначеним набором задач, які в

тестовому випадку є розрахунком факторіалу великого числа. Процес виконання задач показаний на рис. 3.4.

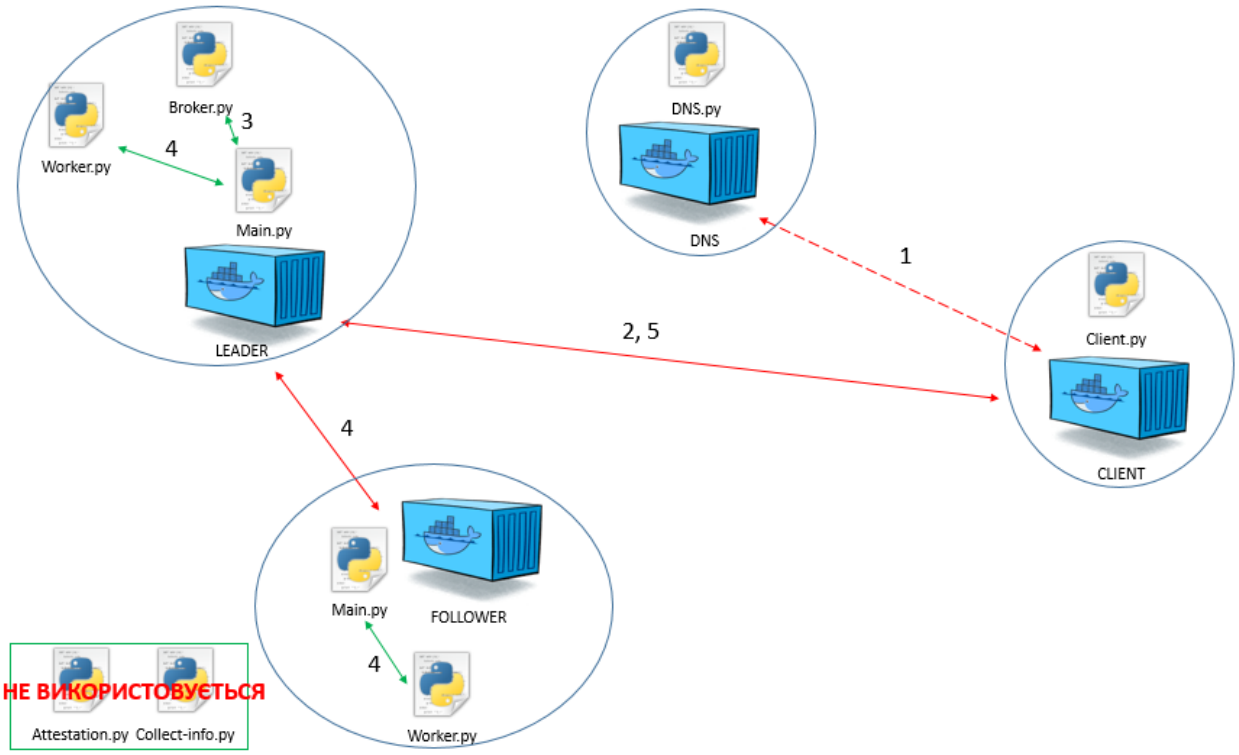


Рисунок 3.4 Процес виконання надійшовшої задачі від користувача

Цифрові позначки, зображені на рис. 3.4. показують наступні етапи виконання задачі в хронологічному порядку:

1. Встановлення з’єднання між клієнтом та DNS сервером. Клієнт посилає запит на отримання IP-адреси тестового кластера. DNS сервер у відповідь відправляє IP-адресу лідера кластера.
2. Клієнт встановлює з’єднання SSH з’єднання з лідером кластера та відправляє йому задачу.
3. Лідер викликає функцію балансування навантаження і визначає, який вузол повинен обробити отриману задачу.
4. Обраний вузол виконує задачу і відправляє результат лідеру.
5. Лідер відправляє результат клієнту.

В якості вузлів тестового кластера використовувалося 2 ПК з різними якісними характеристиками: один значно продуктивніший, але менш енергоефективний, інший – навпаки. Оскільки розмір тестового кластеру замалий для отримання збалансованого результату (покращення по обом показникам), то буде проведено 2 дослідження з різними показниками $w_{productivity}$ та $w_{energy\_efficiency}$, які приймуть граничні значення {0,1} та {1,0} відповідно. Очікуваним результатом є значна перевага по одному з показників при деградації іншого, що показано на рис. 3.5.

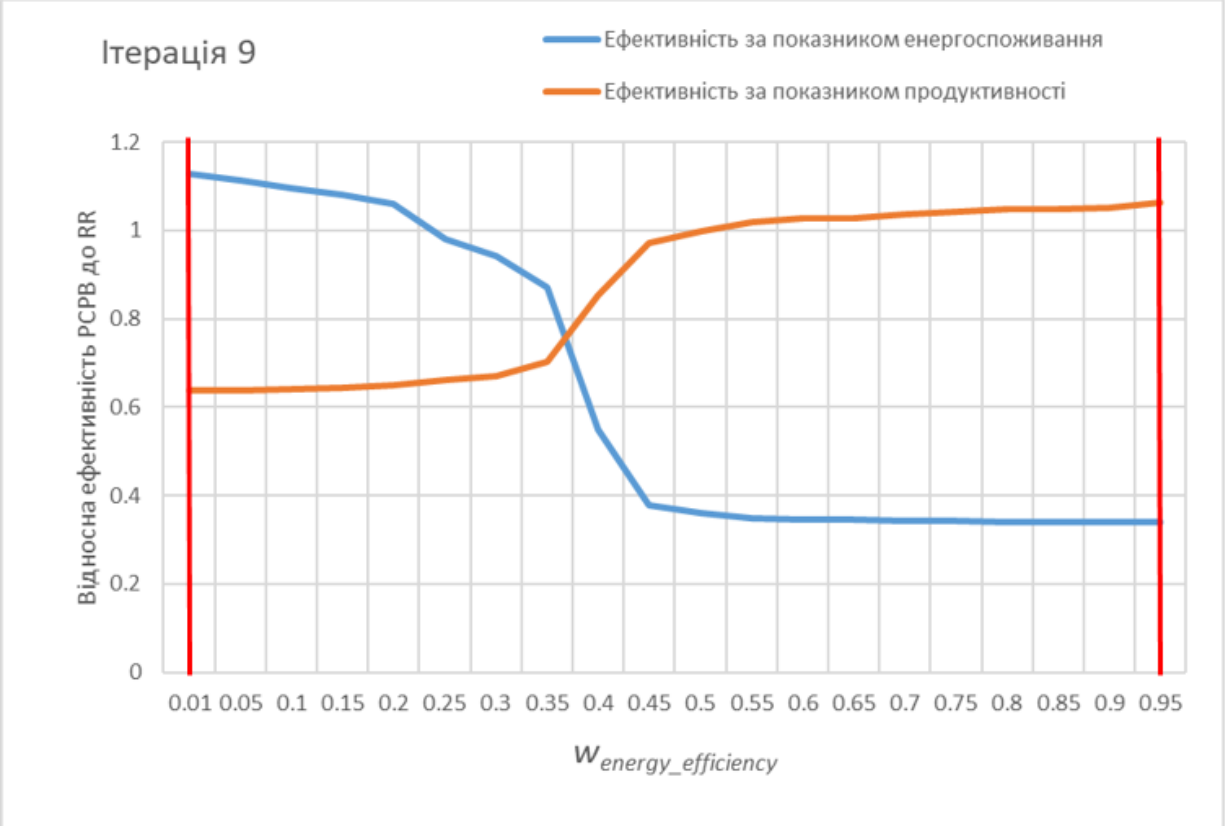


Рисунок 3.5 Очікувані результати тестування

На рис 3.5. можна побачити, що тестування, яке було проведено з граничними значеннями показників $w_{productivity}$ та $w_{energy\_efficiency}$ (відмічені червоними лініями) повинно показати значну ефективність для одного показника при незначному погіршенні іншого.

На рис.3.6. зображено результати тестування роботи ПЗ з використанням алгоритму Round Robin. Варто зазначити, що в цьому та подальших тестах використовувався один і той же набір задач з однаковими часовими моментами надходження заявок. Набір задач складав 10 одиниць, оскільки необхідно було проводити детальній аналіз логу, який виводило в командну строку ПЗ під час роботи.

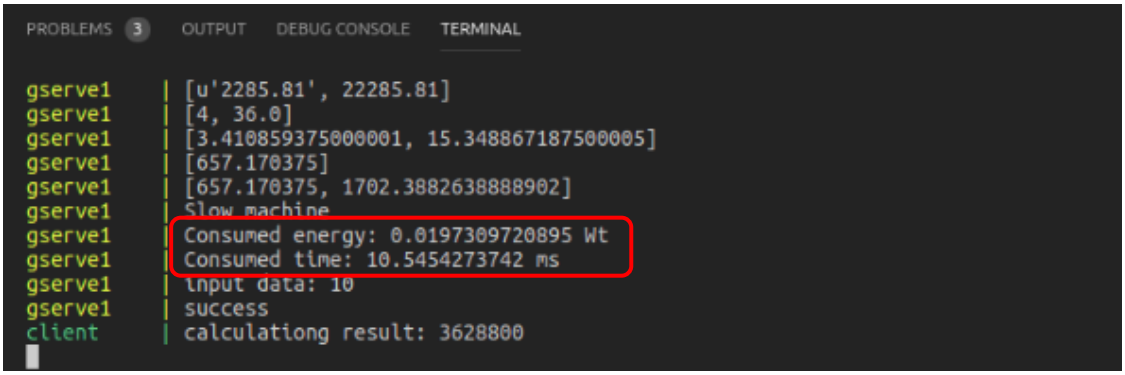


Рисунок 3.6 Результати тестування створеного ПЗ з використанням алгоритму Round Robin

Таким чином, 10 тестових задач було оброблено за 10.54 мс, при цьому було витрачено 0,0197 Вт електроенергії.

Результати роботи алгоритму PCPB з налаштуваннями $w_{productivity} = 1$ та $w_{energy\_efficiency} = 0$ зображено на рис. 3.7.

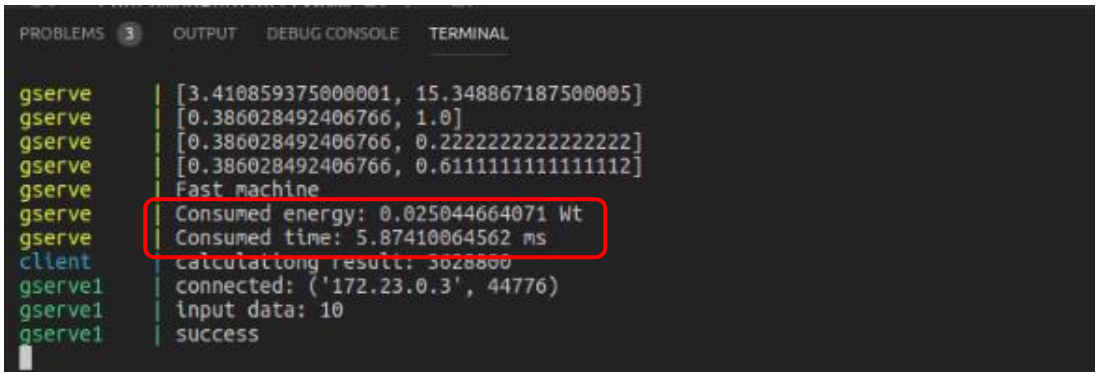
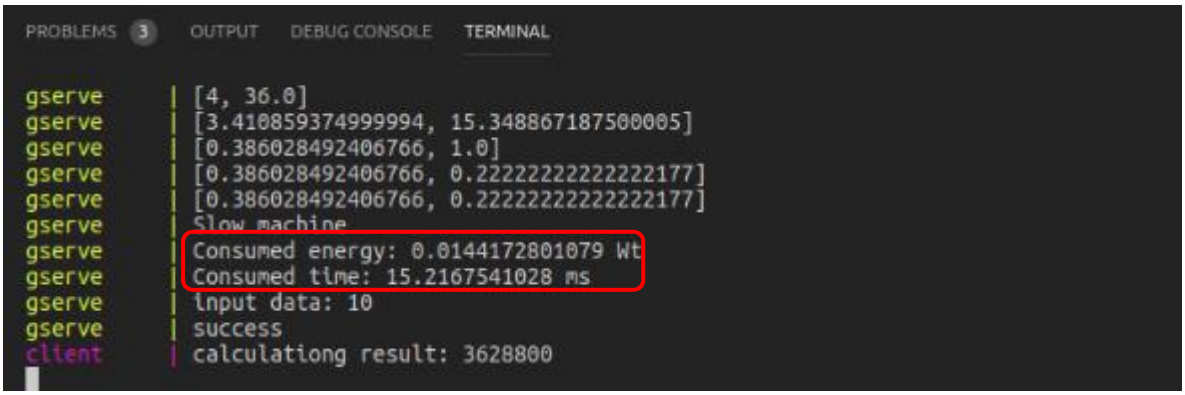


Рисунок 3.7 Результати тестування створеного ПЗ з використанням алгоритму PCPB ($w_{productivity} = 1$; $w_{energy\_efficiency} = 0$)

В цьому випадку задачі було виконано за 5.87 мс, за спожитою електроенергією в 0.0250 Вт.

Результати роботи алгоритму РСРВ з налаштуваннями $w_{productivity} = 0$ та



$w_{energy\_efficiency} = 1$ зображено на рис. 3.8.

Рисунок 3.8 Результати тестування створеного ПЗ з використанням алгоритму РСРВ ($w_{productivity} = 0$; $w_{energy\_efficiency} = 1$)

В цьому випадку задачі було виконано за 15.21 мс, за спожитою електроенергією в 0.0144 Вт.

Результати дослідження були занесені в таблицю 3.1.

Таблиця 3.1 Результати тестової перевірки роботи створеного ПЗ

Алгоритм	Час обробки, мс	Спожита електроенергія, Вт	Ефективність за показником продуктивності, %	Ефективність за показником енергозберігання, %
Round Robin	10,54	0,0197	100,00	100,00
РСРВ (Продуктивність)	5,87	0,025	44,31	-26,90
РСРВ (Енергозберігання)	15,21	0,0144	-44,31	26,90

З таблиці 3.1. видно, що закономірність, описана на рис. 3.5. підтверджується і що алгоритм в ПЗ налаштований вірно. Симетричність результатів пояснюється тим, що у випадку роботи алгоритму Round Robin

половина задач буде оброблена на одному вузлі, а друга половина – на іншому. У випадку роботі РСРВ всі задачі були направлені або на енергоефективний, або на продуктивний вузел, оскільки тестова вибірка не перевантажувала вузли, і необхідності міграції навантаження на інший вузол у випадку перевантаження не існувало.

Висновки

1. Розроблене ПЗ є гнучким інструментом балансування навантаження. Ефективність і коректність його роботи була доведена практичними тестами і моделюванням, оскільки єдиною різницею між Matlab моделлю та створеною програмою є вхідні дані. В той же час внутрішня структура та алгоритми є абсолютно однаковими зі структурної точки зору. Навколо алгоритму побудовано екосистему, яка забезпечує функціонування алгоритму точно таким же чином, як і в імітаційній моделі.

2. До переваг створеного ПЗ відносяться автоматизаваний процес інсталяції, налаштування, мінімальні вимоги (необхідно лише мати інстальований docker та docker-compose). Функціонування ПЗ дозволяє використовувати запропонований алгоритм балансування навантаження з більш ефективно у порівнянні з існуючими. Також система має підвищену надійність та відмовостійкість за рахунок уніфікації функцій лідера та послідовника в одному комплексі та адаптивної реконфігурації кластера у випадку виходу з ладу лідера. Використання алгоритму RAFT дозволяє не враховувати готовність будь-якого вузла до виконання певної задачі: алгоритм гарантує запис копій інформації на всіх вузлах системи.

3. До недоліків створеного ПЗ можна віднести необхідність ручного налаштування файлу worker.py, в якому записаний набір інструкцій, які мають виконуватися на кластері під кожний новий тип задач. Ця проблема може бути вирішена двома шляхами: налаштуванням єдиного GUI (Graphical User Interface – графічний інтерфейс користувача) для всього кластеру, консенсус виконуваних команд на всіх вузлах забезпечується алгоритмом

RAFT, або за рахунок використання сервісів перетворення моделей у програмний код.

4. Крім того, недоліком є вимірювання енергоспоживання вузлів на етапі атестації програмним методом, яке має меншу точність, ніж вимірювання фізичним обладнанням. Проте цей метод є безальтернативний при великому розмірі кластеру, оскільки така атестація, наприклад, десяти тисяч вузлів вимагає значних обсягів часових ресурсів.

5. У великому дата-центрі досить частою подією є заміна або додавання нового обладнання, що робить задачу ручного вимірювання енергоспоживання кожного вузла та внесення цих результатів до кожного вузла занадто важкою та часозатратною, особливо, враховуючи, що такі системи мають працювати повністю в автоматичному режими, тому незначна втрата точності вимірювання є більш допустимим недоліком, ніж необхідність постійного часозатратного втручання людини в автоматичну систему.

ЗАГАЛЬНІ ВИСНОВКИІ

У роботі на здобуття ступеня магістра вирішено важливе наукове завдання щодо підвищення енергоефективності та продуктивності обробки запитів користувача в гетерогенному ЦОД (центрі обробки даних) оператора зв’язку.

В роботі отримані такі теоретичні та експерементальні результати:

1. Досліджено можливості віртуалізації основних компонентів мережі LTE операторів мобільного зв’язку України з використанням технологій SDN та NFV. Кожен вузол серверного кластеру vEPC може виконувати одну або декілька віртуалізованих мережевих функцій одночасно. Допущено, що для спрощення моніторингу та керування серверним кластером кожен вузол може виконувати лише одну віртуалізовану мережеву функцію.
2. Проаналізовано розподіл навантаження серед компонентів мережі LTE та визначено, що доцільним є тестування алгоритмів балансування навантаження як на невеликих кластерах (5-20 машин), які обслуговують сигнальну інформацію, так і на великих (понад 100 машин), які, відповідно, обслуговують основний трафік.
3. Визначено підхід щодо розрахунку обсягу навантаження, який надходитиме на ЦОД мобільних операторів зв’язку України.
4. Проаналізовано проблематику енергоефективності обчислень в серверному кластері, виділено основну проблему – сучасні алгоритми балансування навантаження мають за мету досягти оптимального розподілу навантаження серед вузлів кластера з отриманням максимальної продуктивності, не враховуючи показник енергоефективності. Серед проаналізованих алгоритмів балансування навантаження, які намагаються зменшити енергоспоживання спостерігається інший важливий недолік: продуктивність обчислень знижується. Враховуючи ці фактори можна

представити проблему енергоефективних обчислень як необхідність вибору між продуктивністю та енергоефективністю обчислень.

5. Для вирішення цієї проблеми в дослідженні запропоновано новий алгоритм балансування навантаження, характерною відмінністю якого є одночасне врахування показників продуктивності та енергоефективності в процесі прийняття рішення щодо вибору вузла для обробки чергової задачі.

6. Для перевірки ефективності створена імітаційна модель в середовищі Matlab, яка є досить гнучкою і дозволяє імітувати роботу середнього за розміром серверного кластеру (100-500 вузлів).

7. З допомогою створеної моделі проведено декілька досліджень роботи запропонованого алгоритму PCPB на кластерах різного розміру (5, 20, 100) вузлів. Причому для останнього було проведено дослідження ефективності роботи алгоритму в залежності від ступеню гетерогенності кластеру та в залежності від коефіцієнтів $w_{productivity}$ та $w_{energy\_efficiency}$, які відповідають за міру важливості того чи іншого критерію при прийнятті рішення щодо вибору вузла.

8. Результати цих досліджень показали, що ефективність роботи алгоритму значно залежить від ступеню гетерогенності та обраних коефіцієнтів важливості параметрів, та в меншій мірі від розміру кластеру.

9. Визначено оптимальну конфігурацію параметрів $w_{productivity}$ =60-65% та $w_{energy\_efficiency}$=35-40%, яка дозволяє досягти невеликого зростання продуктивності та енергоефективності (до 25-30%) в порівнянні з класичним алгоритмом балансування навантаження Round Robin.

10. Розроблено програмний комплекс, який дозволяє інсталювати балансувальник навантаження з використанням запропонованого алгоритму. Характерною відмінністю цього комплексу є простота інсталяції та налаштування, підвищена відмовостійкість та доступність. ПЗ дозволяє швидко та просто розгорнути логічний кластер між будь-яким числом вузлів з різних фізичних мереж, з будь-якими сучасними операційними системами сімейства UNIX, Windows або MacOS.

11. Тестування комплексу показало коректність його функціонування та практичну застосовність. Єдиним значним недоліком є необхідність налаштування файла Worker.py кожний раз для обробки задач нового типу. Проте цей недолік можна вирішити, викликавши власний обробник задач із файлу Worker.py за допомогою bash команд (системні команди UNIX), для виконання яких в мові програмування Python є окрема бібліотека.

12. Проведені дослідження показали як наявність проблеми енергоефективності обчислень в серверному кластері, так і визначили шлях вирішення цієї проблеми з використанням нового алгоритму балансування навантаження. Ефективність його роботи досліджена шляхом імітаційного моделювання Matlab. Результатом даного дослідження є програмний продукт, який спрямований на вирішення проблеми енергоефективності, який окрім того ще має низку важливих переваг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sulima, Svitlana & Skulysh, Mariia. (2017). HYBRID RESOURCE PROVISIONING SYSTEM FOR VIRTUAL NETWORK FUNCTIONS. Radio Electronics, Computer Science, Control. 10.15588/1607-3274-2017-1-2.

2. Таненбаум Э., Ван Стеен М. Распределенные системы. Принципы и парадигмы СПб.: Питер, 2003. — 877 с.: ил. — (Серия «Классика Computer Science») — ISBN 5–272–00053–6.

3. https://www.slideshare.net/Yole_Developpement/data-center-market-and-technology-trends-power-electronics-presentation-held-at-apec-2016-from-yole-dveloppement Yole Développement “Data Center Market and Technology Trends Power Electronics presentation”, APEC, 2016

4. “Google Transparency Report,” [Електронний ресурс] Режим доступу до ресурсу: http://www.google.com/transparencyreport/traffic/

5. А. Л. Литвинов ТЕОРІЯ СИСТЕМ МАСОВОГО ОБСЛУГОВУВАННЯ НАВЧАЛЬНИЙ ПОСІБНИК Харків ХНУМГ ім. О. М. Бекетова 2018

6. А.Г. Ложковський ТЕОРІЯ МАСОВОГО ОБСЛУГОВУВАННЯ В ТЕЛЕКОМУНІКАЦІЯХ Затверджено Міністерством транспорту та зв’язку України як підручник для студентів вищих навчальних закладів, які навчаються за напрямом „Телекомунікації” Одеса 2010

7. Urgaonkar B. Agile dynamic provisioning of multi-tier Internet applications / B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, T. Wood // ACM Transactions on Autonomous and Adaptive Systems (TAAS). – 2008. – Vol. 3, № 1. – P. 1–39.

8. Scholler M. Resilient deployment of virtual network functions / M. Scholler, M. Stiemerling, A. Ripke, R. Bless // Proc. 5th Int. Congr. Ultra Mod. Telecommun. Control Syst. Workshops (ICUMT). – St. Petersburg, Russia, 2013. – P. 208-214.

9. Mu'alem, Ahuva, G. Feitelson, Dror. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling,

Parallel and Distributed Systems, IEEE Transactions on 2001/07/01, v. 12, pp. 529-543, DOI - 10.1109/71.932708

10. J. C. Boemer, M. Gibescu and W. L. Kling, "Dynamic models for transient stability analysis of transmission and distribution systems with distributed generation: An overview," 2009 IEEE Bucharest PowerTech, Bucharest, 2009, pp. 1-8. doi: 10.1109/PTC.2009.5282177

11. G. S. Blair, "Building heterogeneous distributed multimedia systems," IEE Colloquium on Building Distributed Systems, London, UK, 1990, pp. 611-613.

12. Liu N. Task Scheduling and Server Provisioning for Energy-Efficient Cloud-Computing Data Centers / N. Liu, Z. Dong, R. Rojas-Cessa. // IEEE 33rd International Conference on Distributed Computing Systems Workshops. – 2013. – №33. – С. 226–231.

13. Novikov A. B. “Algorithms of scalable tasks scheduling in cluster computer system” // Young scientists. — 2011. — №11. T.1. — С. 7479. (Новиков А. Б. Алгоритмы планирования масштабируемых заданий кластерной вычислительной системы // Молодой ученый. — 2011. — №11. Т.1. — С. 74-79.)

14. Choi D. An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems / D. Choi, K.S. Chung, J. Shon // Springer, vol.21, Berlin, Heidelberg

15. Hosseinimotlagh S. A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds / S. Hosseinimotlagh, F. Khunjush, S. Hosseinimotlagh. // Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. – 2014. – №22. – С. 178–182.

16. An Energy and Deadline Aware Resource Provisioning, Scheduling and Optimization Framework for Cloud Systems / Y.Gao, Y. Wang, S. K. Gupta, M. Pedram. // IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. – 2013. – №9. – С. 31:1–31:10

17. Армента-Кано Ф. Min_c: стратегія неоднорідної концентрації задач для енерго-заощаджуючих комп'ютерних планувальників / Ф. Армента-Кано, А. Черних, Х. М. Кортес-Мендоза // Труды ИСП РАН. – 2015. – №6. – С. 355.

18. Jyoti V. Comparative Study of Load Balancing Algorithms / V. Jyoti, K. J. Anant. // IOSR Journal of Engineering (IOSRJEN). – 2013. – С. 45–50.

19. Грушин Д. А. Энергоэффективные вычисления для группы кластеров / Д. А. Грушин, Н. Н. Кузюрин. – Москва, 2013. – 433 с.

20. Xiong G. A virtual service placement approach based on improved quantum genetic algorithm / G. Xiong, Y.-X. Hu, L. Tian, J.-L. Lan, J.-F. Li, And Q. Zhou // Frontiers of Information Technology & Electronic Engineering. – 2016. – Vol. 17, No. 7. – P. 661–671.

21. Gember-Jacobson A. OpenNF: Enabling Innovation in Network Function Control / A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella // Proceedings of the 2014 ACM Conference on SIGCOMM, ser. SIGCOMM '14. – New York, NY, USA, 2014. – P. 163-174.

22. Rajagopalan S. Split/Merge: System Support for Elastic Execution in Virtual Middleboxes / S. Rajagopalan, D. Williams, H. Jamjoom, and A. Wareld // 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). – Lombard, IL, USA, 2013. – P. 227-240.

23. Fajjari I. VNR Algorithm: A Greedy Approach For Virtual Networks Reconfigurations / I. Fajjari, N. Aitsaadi, G. Pujolle and H. Zimmermann. // IEEE Globel Communications Conference, Exhibition and Industry Forum. – Houston, United States, 2011. – P. 1-7.

24. Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing Anton Beloglazov, Department of Computing and Information Systems THE UNIVERSITY OF MELBOURNE, 2013

25. L. A. Barroso, J. Clidaras, and U. Hoezle "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines", 2nd ed. San Rafael, CA, USA: Morgan & Claypool, 2013.

26. CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. (2014) Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3314 OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds Anton Beloglazov and Rajkumar Buyya

27. M. van Steen et al., "A brief introduction to distributed systems", Computing, vol. 98, pp. 967-1009, 2016.

28. M. A. Austin, "Parallel distributed real-time systems in manufacturing (an aerospace view)," Proceedings of 5th International Workshop on Parallel and Distributed Real-Time Systems and 3rd Workshop on Object-Oriented Real-Time Systems, Geneva, Switzerland, 1997, pp. 284-288.

29. “Energy-efficient Backfill-based Scheduling Approach for SLURM Resource Manager” Larysa Globa, Nataliia Gvozdetska and Volodymyr Prokopets// CADSM, February 26 – March 2, 2019, Polyana-Svalyava (Zakarpattya), Ukraine

30. Наукоємні технології оптимізації та керування в інфокомунікаційних мережах : монографія / Під загальною редакцією В.М. Безрука, Л.С. Глоби, О.Є Стрижака. – К.: Інститут обдарованої дитини НАПН України, 2019. – 194 с. ISBN 978-617-7734-02-3

31. Держстат України. Статево-віковий склад населення. [Електронний ресурс] Режим доступу до ресурсу: http://database.ukrcensus.gov.ua/MULT/Dialog/statfile_c_files/stat_vik.html

32. "Статистика споживання трафіку в мережах мобільного зв'язку серед студентів м. Новосибірська" [Електронний ресурс] Режим доступу до ресурсу: http://tayga.info/132324

33. "Статистика споживання трафіку оператора мобільного зв'язку Мегафон" [Електронний ресурс] Режим доступу до ресурсу: https://corp.megafon.ru/press/news/federalnye_novosti/20170424-1510.html

34. "Вищі навчальні заклади в регіонах України на початок 2014 року" [Електронний ресурс] Режим доступу до ресурсу: http://uafrontier.com/vnz-v-regionah-ukrainy-na-pochatok-2014-roku/

35. "Середнє споживання трафіку мобільних мереж України" [Електронний ресурс] Режим доступу до ресурсу: https://itc.ua/news/globaldata-srednee-potreblenie-trafika-na-odnu-sim-kartu-v-2017-godu-v-ukraine-sostavit-menee-1-gb-v-mesyats/

36. “QoS and Energy Efficiency Improving in Virtualized Mobile Network EPC based on Load Balancing”, Larysa Globa, Nataliia Gvozdetska, Volodymyr Prokopets, Oleksandr Stryzhak //ACS 2018, Międzyzdroje, Poland, September 24-26, 2018

37. "Управление энергопотреблением коммерческих серверов" [Електронний ресурс] Режим доступу до ресурсу: http://www.osp.ru/os/2004/02/183912/

38. Струтинський В.Б. Математичне моделювання процесів та систем механіки, Год: 2001, Издательство: Житомир, ЖИТИ,Количество страниц: 613

39. Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. Численные методы. Изд-во "Лаборатория базовых знаний". 2003.

40. Wikipedia. FLOPS [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/FLOPS

41. “Експериментальне дослідження енергоефективності обробки даних в розподіленому дата центрі”, Прокопець В.А. // Перспективи розвитку інформаційно-телекомунікаційних технологій та систем 2018, Київ, Україна.

42. “Експериментальне дослідження енергоефективності обробки даних в розподіленому дата центрі”, Прокопець В.А., Глоба Л.С. // Проблеми телекомунікацій 2018, Київ, Україна

43. Christoph Möbius, Waltenegus Dargie, Senior Member, IEEE, and Alexander Schill “Power Consumption Estimation Models for Processors, Virtual

Machines, and Servers”, IEEE Transactions on Parallel & Distributed Systems,2014

44. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers Christoph Möbius, Waltenegus Dargie, Senior Member, IEEE, and Alexander Schill

45. “Power Consumption and Performance Balance (PCPB) scheduling algorithm for computer cluster”, Alexander Schill, Larysa Globa, Oleksandr Stepurin, Nataliia Gvozdetska, Volodymyr Prokopets// UkrMiCo’2017, 11-15 September 2017, Odesa, Ukraine

46. "CPU Stress Testing" [Електронний ресурс] Режим доступу до ресурсу: https://github.com/ethdragon/CPU-stress-testing

47. X. Fan, W. D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007, pp. 13–23.

48. “Prognostic-Reactive NFV Resource Allocation Method for Implementation in Virtualized Mobile Network EPC of Ukraine’’, Larysa S. Globa, Volodymyr A. Prokopets, Nataliia A. Gvozdetska//Black Sea Com 2018, Batumi, Georgia

49. Jeabin Lee, Byeong-Gyu Nam and Hoi-Jun Yoo, "Dynamic Voltage and Frequency Scaling (DVFS) scheme for multi-domains power management," 2007 IEEE Asian Solid-State Circuits Conference, Jeju, 2007, pp. 360-363. doi: 10.1109/ASSCC.2007.4425705

50. Aldossary M, Djemame K. Performance and Energy-Based Cost Prediction of Virtual Machines Auto-Scaling in Clouds. In: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018). 29-31 Aug 2018, Prague, Czech Republic. 2018. pp. 502-509. DOI: 10.1109/SEAA.2018.00086

51. Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing Anton Beloglazov, Department of Computing and Information Systems THE UNIVERSITY OF MELBOURNE, 2013

52. Наукоємні технології оптимізації та керування в інфокомунікаційних мережах : монографія / Під загальною редакцією В.М. Безрука, Л.С. Глоби, О.Є Стрижака. – К.: Інститут обдарованої дитини НАПН України, 2019. – 194 с. ISBN 978-617-7734-02-3