

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК 004.94; 004.4; 004.62

«До захисту допущено»
Завідувач кафедри

О.А.Павлов
(підпис) (ініціали, прізвище)
“ ” 20 18 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки та інформаційні технології
(код та назва спеціальності)

спеціалізацією Інформаційні управляючі системи та технології
(код та назва спеціалізації)

на тему: Управління віртуалізованими ресурсами кластеру хмарного
центру обробки даних

Виконав: студент VI курсу групи ІС-61м
(шифр групи)

Коваль Андрій Анатолійович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доц., к.т.н., доц. Жаріков Е.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант к.т.н., доц. Жданова О.Г.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

РЕФЕРАТ

Магістерська дисертація: 116 с., 17 рис., 7 табл., 1 додаток, 53 джерела.

Актуальність. Концепція центрів обробки даних або дата-центрів (ЦОД) втілена багатьма великими корпораціями для забезпечення доступу великої кількості користувачів до певних ресурсів. Ефективне управління ЦОД пов'язане з необхідністю розв'язання низки проблем, насамперед створення умов для функціонування інформаційно-обчислювальних потужностей ЦОД, управління віртуалізованими ресурсами, забезпечення надійності та безпеки. Вкладаючи кошти, хостингові компанії сподіваються на прибуток та очікують зменшення витрат на експлуатацію ЦОД, зниження вартості обслуговування користувачів, що дозволить, зрештою, закласти основу для ефективної діяльності, як самої компанії, так і клієнтів.

Забезпечення рівня вимог користувачів з мінімізацією витрат становить сутність проблеми управління функціонуванням ЦОД. Зазвичай цю комплексну проблему розбивають на ряд задач менших розмірів, але від того не набагато простіших. Однією з них є задача управління ресурсами і навантаженням ЦОД.

У зв'язку з цим актуальною є розробка алгоритму навчання з підкріпленням (НП, англ. reinforcement learning, RL) [1] для управління віртуалізованими ресурсами, який допоможе зменшити споживання електроенергії та час порушення вимог угоди про рівень послуг (англ. Service-level agreement, SLA).

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Розробка та впровадження системи управління IT-інфраструктурою з консолідованими інформаційно-обчислювальними ресурсами» (№ 0115U000322).

Метою дослідження є поліпшення якості управління віртуалізованими обчислювальними ресурсами кластеру хмарного ЦОД шляхом розробки алгоритму управління, що дозволяє зменшити споживання електроенергії та час порушення вимог SLA.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- проаналізувати предметне середовище управління віртуалізованими ресурсами ЦОД;
- провести огляд методів управління обчислювальними ресурсами;
- обрати середовище моделювання ЦОД;
- розробити модель ЦОД в обраному середовищі моделювання;
- розробити моделі споживання електроенергії фізичними серверами;
- підготувати дані для моделювання динамічного навантаження віртуальних машин в ЦОД;
- розробити алгоритм НП для управління віртуалізованими обчислювальними ресурсами ЦОД;
- виконати програмну реалізацію алгоритму НП;
- провести дослідження ефективності розробленого алгоритму.

Об'єктом дослідження є процес управління віртуалізованими обчислювальними ресурсами в центрі обробки даних.

Предметом дослідження є методи і алгоритми управління віртуалізованими обчислювальними ресурсами в центрі обробки даних.

Методами дослідження є методи машинного навчання, які базуються на НП.

Наукова новизна отриманих результатів. Проаналізовано можливість застосування НП для управління віртуалізованими ресурсами хмарних ЦОД. Розроблено метод динамічного розміщення віртуальних машин на основі НП, який при виборі управляючих впливів враховує витрати електроенергії та час порушення вимог угоди про рівень послуг. Розроблений алгоритм агента, який враховує зміни робочого навантаження на ресурси для прийняття рішення щодо включення або переключення в сплячий режим незавантажених фізичних серверів з метою зменшення витрат електроенергії. Запропонований агент навчання з підкріпленням базується на методі *Q*-навчання (англ. *Q-learning*) [2], який дозволяє визначати наближену до оптимальної політику управління режимами роботи фізичного сервера без попередньої інформації про навантаження.

Публікації. Матеріали роботи опубліковані у тезах 10-ї Всеукраїнської науково-практичної конференції «Комп'ютерні інтелектуальні системи та мережі» [3]; опубліковані у тезах 18-ї Всеукраїнської студентської науково-практичної конференції «Наука та техніка ХХІ століття» [4]; опубліковані у тезах науково-практичної конференції «Інформатика та обчислювальна техніка-ІОТ-2018» [5]; опубліковані в журналі «Наукові вісті Далівського університету» [6]; представлені на 14-ій міжнародній конференції Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering-TCSET-2018, Львів-Славське, Україна та опубліковані в електронній бібліотеці IEEE Xplore Digital Library [7].

МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ЦЕНТР ОБРОБКИ ДАНИХ, ДАТА-ЦЕНТР, ВІРТУАЛЬНА МАШИНА, ВІРТУАЛІЗАЦІЯ, ОБЧИСЛЮВАЛЬНІ РЕСУРСИ, ЕНЕРГОЕФЕКТИВНІСТЬ, SLA

ABSTRACT

Master dissertation: 116 pp., 17 fig., 7 tab., 1 app., 53 sources.

Topicality. The concept of data centers is embodied by many large corporations to provide access to a large number of users to certain resources. Effective management of the data center is connected with the need to solve a number of problems, first of all, creation of conditions for functioning of information and computing facilities of data centers, management of virtualized resources, maintenance of reliability and safety. By investing, hosting companies are hoping for profit and expecting a reduction in the cost of operating the data center, reducing the cost of customer service, which will eventually lay the foundation for effective business, both for the company itself and for customers.

Ensuring the level of user requirements by minimizing costs is the essence of the problem of managing the functioning of the data center. Typically, this complex problem is divided into a number of smaller tasks, but not so much simpler. One of them is the task of resources and load management in datacenter.

In this regard, it is important to develop a reinforcement learning algorithm [1] for managing virtualized resources that will help reduce power consumption and SLA violation time.

Relationship of work with scientific programs, plans, themes. The research was carried out at the Department of Computer-Aided Management And Data Processing Systems of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» within the theme «Development and implementation of an IT infrastructure management system with consolidated information and computing resources» (№ 0115U000322).

The aim of the research is to improve the quality of virtualized cloud computing resources management by developing a management algorithm that reduces power consumption and SLA violation time.

To achieve this goal, the following tasks must be performed:

- analyze the object environment of the virtualized data center resources management;

- review the methods of computing resources management;
- choose the simulation environment of data center;
- develop a data center model in a selected simulation environment;
- develop models of power consumption by physical servers;
- prepare data for simulating the dynamic load of virtual machines in data center;
- develop a modified reinforcement learning algorithm for managing the virtualized computing resources of data center;
- develop the software implementation of the reinforcement learning algorithm;
- make a research of developed algorithm effectiveness.

The object of research is a process of virtualized computing resources management in data center.

The subject of research is a methods and algorithms for virtualized computing resources management in data center.

Research methods are methods of machine learning, which based on reinforcement learning.

Scientific novelty of the obtained results. The possibility of reinforcement learning usage for of virtualized resources management of cloud data centers is analyzed. The method of dynamic placement of virtual machines on the basis of reinforcement learning is developed, which, when choosing controlling influences, takes into account power consumption and SLA violation time. An agent algorithm is developed which takes into account the changes in the workload on resources for deciding whether to turn on or switch to sleep mode of underutilized physical servers in order to reduce the cost of electricity. The proposed reinforcement learning agent is based on the *Q*-learning method [2], which allows determining approximation to optimal policy of controlling the modes of the physical server operation without prior load information.

Publications. The materials of research are published in theses of the 10th All-Ukrainian Scientific and Practical Conference «Computer Intelligent Systems and Networks» [3]; published in theses of the 18th All-Ukrainian Students' Scientific and Practical Conference «Science and Technology of the XXI Century» [4]; published in theses of the scientific and practical conference «Informatics and Computer Science-ICS-2018»

[5]; published in journal «Scientific News of Dahl University» [6]; presented at the 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering – TCSET-2018, Lviv-Slavske, Ukraine and published in IEEE Xplore Digital Library [7].

MACHINE LEARNING, REINFORCEMENT LEARNING, DATA CENTER, VIRTUAL MACHINE, VIRTUALIZATION, COMPUTING RESOURCES, ENERGY EFFICIENCY, SLA

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ	12
ВСТУП.....	14
1 ОГЛЯД РІШЕНЬ З УПРАВЛІННЯ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ В ЦЕНТРИ ОБРОБКИ ДАНИХ	17
2 ІСНУЮЧІ МЕТОДИ УПРАВЛІННЯ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ	28
2.1 Адаптивний евристичний метод консолідації віртуальних машин в центрі обробки даних.....	28
2.1.1 Формулювання проблеми міграції віртуальної машини.....	28
2.1.2 Формулювання проблеми динамічної консолідації віртуальних машин ..	31
2.1.3 Адаптивні евристики для динамічної консолідації віртуальних машин ...	34
2.1.3.1 Виявлення перевантажених фізичних серверів	35
2.1.3.2 Вибір віртуальних машин для міграції.....	37
2.1.3.3 Розміщення віртуальних машин.....	37
2.2 Метод управління розподілом ресурсів центру обробки даних при віртуальному хостингу	39
2.2.1 Формулювання проблеми розподілу й управління ресурсами.....	39
2.2.2 Математична модель розподілу ресурсів при віртуальному хостингу.....	39
2.2.3 Задачі планування розподілу ресурсів при надлишку ресурсів	41
2.2.4 Генетичний алгоритм розв’язання задачі	45
2.2.5 Евристичний алгоритм розв’язання задачі	46
2.3 Метод автоконфігурування віртуальних машин на основі навчання з підкріпленням	47
2.3.1 Формулювання проблеми автоконфігурування віртуальних машин.....	47
2.3.2 Навчання з підкріпленням для автоконфігурування віртуальних машин .	48
2.3.3 Онлайн алгоритм навчання з підкріпленням.....	50
2.4 Висновок до розділу	51
3 РОЗРОБКА МЕТОДУ ДИНАМІЧНОГО РОЗМІЩЕННЯ ВІРТУАЛЬНИХ МАШИН НА ОСНОВІ НАВЧАННЯ З ПІДКРІПЛЕННЯМ.....	52

3.1	Задача зменшення споживання електроенергії та часу порушення вимог SLA в центрі обробки даних.....	52
3.2	Математична модель центру обробки даних	54
3.3	Метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням	59
3.4	Модель агента навчання з підкріпленням	61
3.5	Висновок до розділу	64
4	ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	65
4.1	Інформаційне забезпечення	65
4.1.1	Вхідні дані	65
4.1.2	Вихідні дані	68
4.2	Програмне та технічне забезпечення	70
4.2.1	Засоби розробки.....	70
4.2.2	Вимоги до технічного забезпечення.....	71
4.2.2.1	Загальні вимоги	71
4.2.3	Архітектура програмного забезпечення	71
4.2.3.1	Діаграма класів.....	71
4.2.3.2	Діаграма послідовності	74
4.2.3.3	Специфікація функцій	75
4.2.4	Керівництво користувача	84
4.3	Висновок до розділу	90
5	РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ	91
5.1	Порядок проведення досліджень.....	91
5.2	Дослідження впливу коефіцієнтів на ефективність роботи розробленого методу.....	93
5.3	Порівняння розробленого методу з адаптивним евристичним методом консолідації віртуальних машин	97
5.4	Висновки до розділу	99
	ЗАГАЛЬНІ ВИСНОВКИ	100
	ПЕРЕЛІК ПОСИЛАНЬ	102

ДОДАТОК А Графічний матеріал.....	108
ПЛАКАТ 1 Блок-схема алгоритму динамічного розміщення віртуальних машин на основі навчання з підкріпленням.....	109
ПЛАКАТ 2 Блок-схема алгоритму розподілу віртуальних машин.....	110
ПЛАКАТ 3 Математична модель	111
ПЛАКАТ 4 Структура вхідних і вихідних даних	112
ПЛАКАТ 5 Схема структурна класів програмного забезпечення	113
ПЛАКАТ 6 Схема структурна послідовності.....	114
ПЛАКАТ 7 Копії екранних форм	115
ПЛАКАТ 8 Результати досліджень	116

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

RL	– англ. reinforcement learning – навчання з підкріпленням.
SLA	– англ. Service-level agreement – угода про рівень послуг.
ЦОД	– центр обробки даних.
НП	– навчання з підкріпленням.
COA	– сервіс-орієнтована архітектура.
VM	– віртуальна машина.
ФС	– фізичний сервер.
СУІ	– система управління ІТ-інфраструктурою.
QoS	– англ. Quality of Service – якість обслуговування.
ITSM	– англ. IT Service Management – керування ІТ-послугами.
АСУ	– автоматизована система управління.
УІТП	– управління інформаційно-технологічними послугами.
Live migration	– жива міграція.
MAD	– англ. Median Absolute Deviation – середнє абсолютне відхилення.
IQR	– англ. Interquartile Range – міжквартильний діапазон.
LR	– англ. Local Regression – локальна регресія.
LRR	– англ. Local Regression Robust – надійна локальна регресія.
MMT	– англ. The Minimum Migration Time – мінімальний час міграції.
RC	– англ. The Random Choice – випадковий вибір.
MC	– англ. The Maximum Correlation – максимальна кореляція.
MDP	– англ. Markov decision process – Марківський процес прийняття рішень.
MIPS	– англ. Million instructions per second – мільйон інструкцій на секунду.
IaaS	– англ. Infrastructure as a service – інфраструктура як послуга.
PaaS	– англ. Platform as a service – платформа як послуга.
SaaS	– англ. Software as a service – програмне забезпечення як послуга.

- МФМ – менеджер фізичних машин.
- МВМ – монітор віртуальних машин.

ВСТУП

Сьогодні розвиток інформаційних технологій пов'язаний із реалізацією концепції ЦОД або дата-центрів – комплексних організаційно-технічних рішень для створення високопродуктивної, відмовостійкої ІТ-інфраструктури. До головних завдань ЦОД належать консолідоване зберігання і опрацювання даних користувачів, надання їм прикладних сервісів, підтримка функціонування застосунків.

Концепція ЦОД втілена багатьма великими корпораціями переважно для забезпечення доступу великої кількості користувачів до певних ресурсів (сервісів, застосунків, обчислювальних потужностей, даних тощо). Виявилось, що ефективна організація ЦОД пов'язана з необхідністю розв'язання низки проблем, насамперед створення умов для функціонування інформаційно-обчислювальних потужностей ЦОД, управління віртуалізованими ресурсами, забезпечення надійності та безпеки. Схожі проблеми постають перед корпораціями з розвиненою розподіленою ІТ-інфраструктурою, насамперед хостинговими компаніями. Вкладаючи кошти, компанії сподіваються на прибуток. У будь-якому випадку вони очікують зменшення витрат на експлуатацію ЦОД, зниження вартості обслуговування користувачів, що дозволить, зрештою, закласти основу для ефективної діяльності, як самої компанії, так і клієнтів.

Клієнти своє бачення роботи ІТ-інфраструктури погоджують із хостинговою компанією на рівні вимог, до яких звичайно належать: вартість послуг; доступність і керованість ІТ-інфраструктури; цілісність даних; безпека; надійність; масштабованість.

Досягнення рівня вимог користувачів найменшими коштами становить сутність проблеми створення і забезпечення функціонування ЦОД. Зазвичай цю комплексну проблему розбивають на ряд проблем менших розмірів, але від того не набагато простіших. Однією з них є проблема управління віртуалізованими ресурсами і навантаженням ЦОД.

Необхідні гнучкі рішення, які ґрунтуються на оцінюванні стану ресурсів і обсягів навантаження та полягають у правильному балансуванні навантаження і

ефективному управлінні ресурсами. Для систематичного прийняття правильних рішень необхідні інструментарій та комплекс методик і алгоритмів для вирішення задачі підтримки ІТ-інфраструктури. Їх створення становить важливу науково-практичну проблему, розв'язання якої вимагає розуміння процесів, які відбуваються в хостингових компаніях, функціонування ІТ-інфраструктури, чіткої постановки конкретних завдань дослідження, розробки математичних моделей, моделей ЦОД і відповідних методів вирішення задачі та реалізації згаданих вище методик і алгоритмів.

Комплекс задач дослідження та їх постановки залежать від багатьох чинників, які тією чи іншою мірою впливають на згадані вище процеси. Першим таким чинником є модель хостингу. Другим важливим чинником є прийнята архітектура побудови програмних систем. Це традиційна або сервіс-орієнтована архітектура (СОА). Перша з них ґрунтується на абстракціях застосунків та елементів їх побудови. Переважно використовується трирівнева архітектура із сервером бази даних, сервером застосунків і клієнтом. У СОА ідеологія побудови виходить із абстракції сервісів. Мета розроблення системи полягає у створенні комплексу прийняттого рівня абстракцій сервісів, які використовуються багатьма застосунками, а не жорстко прив'язуються до одного з них.

Для масштабування розподілених систем, управління їх ресурсами і балансування навантаження, хостингові організації використовують клієнт-серверні і Web-технології, технології віртуалізації і кластеризації.

У ході дослідження будуть розглянуті технології віртуалізації, що надають користувачу можливість абстрагуватися від особливостей окремих груп ресурсів, об'єднати їх у апаратно-програмні комплекси потрібної конфігурації і спростити управління ними. Віртуалізація платформ полягає у створенні віртуальних машин (ВМ) – програмних абстракцій, що запускаються на платформі фізичних апаратно-програмних (хостових) систем. Віртуалізація ресурсів узагальнює підходи до створення ВМ і переносить їх на усі види ресурсів – обладнання ЦОД, простори імен, мереж і т.п.

Досягнення переваг клієнт-серверних і Web-технологій, кластеризації і віртуалізації вимагає додаткових зусиль, оскільки процеси прийняття рішень стають складнішими і вимагають розвиненого інструментарію їх підтримки. Необхідно визначити склад кластерів, параметри ВМ і ефективно управляти їх ресурсами. З'являється необхідність розроблення моделей, алгоритмів управління інформаційно-обчислювальними процесами ЦОД з урахуванням специфіки архітектурних рішень, особливостей кластеризації і віртуалізації та реалізації їх у системі управління ІТ-інфраструктурою (СУІ).

Метою дослідження є поліпшення якості управління віртуалізованими обчислювальними ресурсами кластеру хмарного ЦОД шляхом розробки алгоритму управління, що дозволяє зменшити споживання електроенергії та час порушення вимог SLA.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- проаналізувати предметне середовище управління віртуалізованими ресурсами центру обробки даних та провести огляд методів управління обчислювальними ресурсами;
- обрати середовище моделювання центрів обробки даних;
- розробити модель центру обробки даних в обраному середовищі моделювання;
- розробити моделі споживання електроенергії фізичними серверами (ФС);
- підготувати дані для моделювання динамічного навантаження ВМ в центрі обробки даних;
- розробити алгоритм НП для управління віртуалізованими обчислювальними ресурсами центру обробки даних;
- виконати програмну реалізацію алгоритму НП;
- провести дослідження ефективності розробленого алгоритму.

Об'єктом дослідження є процес управління віртуалізованими обчислювальними ресурсами в центрі обробки даних.

1 ОГЛЯД РІШЕНЬ З УПРАВЛІННЯ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ В ЦЕНТРІ ОБРОБКИ ДАНИХ

Підходи машинного навчання були досліджені для управління ресурсами та витрат електроенергії у великомасштабних розподілених системах, таких як Grid та хмара. Політика консолідації завдань у [8] виконує всі завдання з мінімальною кількістю ресурсів і виконує планування як основну роль у зниженні витрат електроенергії. У роботі використовується підхід машинного навчання, який вивчає поточну інформацію про систему, таку як рівень споживання електроенергії, завантаження процесора, що сприяє поліпшенню якості планування рішень. Метою цієї політики є максимізація задоволеності користувачів без збільшення витрат електроенергії. В [9] запропоновано алгоритм онлайн навчання, який динамічно вибирає різних експертів для прийняття рішень щодо управління електроенергією, де кожен експерт є попередньо розробленою політикою управління електроенергією. Різні політики перевершують одна одну при різних робочих навантаженнях та апаратних характеристиках.

Дослідження показали доцільність використання підходів НП у розподілі ресурсів [10] [11], керування живленням [8] [12] та саморегулювання контролера пам'яті [13].

Щоб відобразити ресурси віртуальних машин до вимог продуктивності, автори в [10] запропонували підхід VCONF для навчання автоматизації процесів конфігурації VM у відповідь на зміну вимог застосунку. Запропонований метод використовує модельні алгоритми НП для вирішення проблем масштабованості та адаптації при застосуванні НП в реальному управлінні системою. VCONF може знаходити близькі до оптимальних конфігурації віртуальних машин в малих системах і показує хорошу адаптивність та масштабованість. Але запропонований спосіб не враховує необхідності міграції віртуальних машин, при нестачі ресурсів фізичного сервера.

У [11] гібридний онлайн алгоритм НП виділяє сервери між множиною застосунків, щоб максимально збільшити очікувану суму винагород SLA у кожному

застосунку. Цей підхід дозволяє контролеру НП завантажуватися з існуючої політики управління, істотно знижуючи рівень навчання. Ефективність підходу перевіряється за допомогою простої моделі ЦОД. Але запропонований підхід обмежений орієнтуванням на веб-застосунки і не застосовується до віртуальних середовищ.

Крім того, у [12] політика управління витратами електроенергії на рівні системи, яка базується на НП, забезпечила зниження витрат електроенергії на 24%. Вона вивчає оптимальну політику без попередньої інформації про навантаження. Автори встановлюють затримку у виконанні дії як обмеження продуктивності, мінімізуючи споживання електроенергії. З огляду на існуючу технологію управління витратами електроенергії на базі машинного навчання, НП може досягти компромісу між потужністю та кращою політикою управління енергією.

В роботі [14] представлена децентралізована архітектура енергозберігаючої системи управління ресурсами для ЦОД. Визначені проблеми мінімізації споживання енергії при виконання вимог QoS (англ. Quality of service – якість обслуговування) та сформульовані вимоги для політики розподілу VM. Запропоновано три стадії безперервної оптимізації розміщення VM: перерозподіл відповідно до поточного використання декількох системних ресурсів, оптимізації віртуальних мережеских топологій, встановлених між VM та перерозподіл VM з урахуванням теплового стану ресурсів. Для першої стадії застосовані евристичні алгоритми, які були оцінені за допомогою розширення для інструмента CloudSim. Один з алгоритмів призвів до зменшення споживання енергії в ЦОД.

Окрім того, політика мінімізації міграцій надає гнучке регулювання SLA встановлюючи відповідні значення порогів утилізації: SLA може бути послаблена, що призводить до збільшення споживання енергії. Політика забезпечує гетерогенність апаратних засобів і VM; не потребує інформації про конкретні програми що працюють на VM; не залежить від типу навантаження. Отримані результати дослідження показали, що динамічна консолідація VM призводить до значної економії енергії забезпечуючи вимоги якості обслуговування та скорочення витрат на експлуатацію.

В [15], управління віртуалізованими ресурсами у хмарному середовищі розглядається як проблема автоматичного управління, використовуючи НП. Автори використовували метод Q -навчання для управління кількістю віртуальних машин, які надають хмарні сервіси. Запропонований алгоритм зберігає значення пар дія-винагорода і використовує їх для визначення необхідності додавання чи деактивування віртуальних машин під час роботи хмарного сервісу. Слід зазначити, що запропонований підхід зосереджений на довгострокових операціях з хмарними сервісами і не розглядає розміщення віртуальних машин на фізичних серверах.

У статті [16] представлено метод уніфікованого навчання з підкріпленням (англ. a unified reinforcement learning, URL), що відкриває новий напрям в автоконфігуруванні VM та апаратних засобів в умовах хмарних обчислень. В цьому методі агент навчання з підкріпленням регулює конфігурацію VM щоб максимізувати свою винагороду в довгостроковій перспективі. Для прискорення процесу навчання в великомасштабних системах розроблено різні моделі апроксимації винагород від дій по конфігурації VM. Результати дослідження показали виграш в продуктивності і пропускній здатності системи.

Недоліком є те, що URL-фреймворк не виконує керуючі впливи на L2-кеш, який знаходиться на першому місці в області віртуалізації. Підхід навчання з підкріпленням має можливість змінювати будь-яку початкову конфігурацію до оптимальної, але сам процес навчання займає багато часу. Точність моделі мало впливає на якість кінцевої конфігурації, проте це впливає на якість конфігурації в процесі навчання. Агент завжди може закінчити процес конфігурації, до того як зміниться навантаження на застосунок. Здатність агента навчання з підкріпленням реагувати на зміни навантаження потребують подальшого дослідження.

У статті [17] запропоновано метод динамічної консолідації на основі НП (англ. Reinforcement Learning-based Dynamic Consolidation method, RL-DC), що забезпечує зменшення кількості активних фізичних серверів відповідно поточним вимогам до ресурсів. RL-DC використовує агент навчання з підкріпленням для вивчення оптимальної політики щоб визначити режим роботи фізичного сервера. На основі попередніх знань агент визначає коли фізичний сервер потрібно перевести в сплячий

або в активний режим роботи та покращує себе коли змінюється робоче навантаження. Тому RL-DC не потребує жодної інформації про робоче навантаження і динамічно адаптується до середовища для зменшення витрат електроенергії та покращення продуктивності. Проте в статті не наведено спосіб зменшення кількості станів середовища. Також недоліком є те, що ресурсні можливості фізичних серверів та використання ресурсів ВМ характеризуються єдиним параметром – продуктивністю процесора.

Міграції ВМ широко використовуються для динамічного керування ресурсами [18] в багатьох фреймворках хмарного керування ресурсами, рішеннях та системах. Sandpiper [19] – це система, яка автоматизує завдання моніторингу та виявлення вузьких місць, визначає нове розташування віртуальних машин на фізичних серверах і, як результат, ініціює необхідні міграції віртуальних машин. Система базується на підходах "чорного" та "сірого" ящиків для виявлення вузьких місць та резервування ресурсів. Але фіксовані пороги використання ресурсів не є ефективними для застосування в системах управління IaaS зі змішаним навантаженнями та нестационарними моделями використання ресурсів.

Автори [20] запропонували алгоритм управління Measure-Forecast-Remap (MFR) для міграцій віртуальних машин, щоб мінімізувати кількість фізичних серверів, необхідних для підтримки робочого навантаження за певним показником порушення вимог SLA. Вони використовують оціночну модель для прогнозування майбутнього використання ресурсів стандартним способом, а MFR – для розміщення набору віртуальних машин на фізичному сервері, не враховуючи кількість міграцій. Автори [21] запропонували автоматизовану систему керування ємністю та робочим навантаженням, яка об'єднує декілька контролерів ресурсів за трьома різними областями та часовими масштабами.

В роботі [22] представлена система CloudScale, яка здатна адаптуватися до розподілу ресурсів для різних віртуальних машин у хмарних середовищах для зменшення витрат на ресурси та електроенергію. CloudScale використовує модель онлайн прогнозування короткострокових ресурсних потреб. Але такий підхід

використовує невеликий демон моніторингу пам'яті в межах кожної віртуальної машини, який не підходить для багатьох хмарних середовищ.

Детальний аналіз проблеми енергоефективності та ефективної динамічної консолідації віртуальних машин виконали Beloglazov та Vuууа [23]. Автори аналізують проблему консолідації онлайн, офлайн, детермінованих та динамічних віртуальних машин та пропонують адаптивну евристику для динамічного процесу консолідації віртуальних машин. Автори показують, що адаптивна евристика для проблеми енергоефективності та ефективної динамічної консолідації віртуальних машин краще оптимального онлайн детерміністичного алгоритму. Запропонована модель системи являє собою багаторівневий набір, що складається з локальних та глобальних менеджерів. Результатом запропонованого алгоритму є комбінований план міграції нових віртуальних машин та вибраних віртуальних машин з перевантажених і недовантажених фізичних серверів. Недоліком методу є те, що процес міграції груп віртуальних машин виконується у дискретному режимі при створенні додаткового навантаження на фізичний сервер.

У дослідженні [24] представлена система динамічного ресурсного забезпечення та моніторингу (Dynamic Resources Provisioning and Monitoring, DRPM), яка є багатоагентною системою для управління хмарним постачальником ресурсів, беручи до уваги вимоги SLA клієнтів. DRPM містить новий алгоритм Host Fault Detection (HFD) для вибору віртуальної машини та використовує глобальний допоміжний агент та набір локальних допоміжних агентів. Пропонована система DRPM оцінюється за допомогою інструменту моделювання CloudSim, і результати показують що DRPM дозволяє хмарному постачальнику ресурсів збільшити використання ресурсів, зменшити енергоспоживання та уникати порушень SLA. Однак DRPM здійснює контроль у дискретному режимі за допомогою трьох основних етапів: моніторингу, аналізу та виконання. Крім того, використання локальних допоміжних агентів може бути неможливим для багатьох хмарних постачальників ресурсів.

У [25] описана динамічна міграція віртуальних машин для покращення обсягу необхідної потужності та показника порушення SLA. Алгоритм прогнозує змінні навантаження за інтервалами, меншими за тимчасову шкалу зміни попиту. Ця робота

зосереджена на використанні динамічної консолідації, але в ній не розглядається споживання електроенергії фізичними серверами.

В деяких підходах консолідації віртуальних машин сформульована як проблема оптимізації [26] [27] [28]. Хоча проблема оптимізації пов'язана з обмеженнями, такими як потужність ЦОД та SLA. Тому в цих роботах використовують евристичний метод для багатовимірної задачі упаковки в контейнери як алгоритм консолідації навантаження. ЦОДи є контейнерами, а віртуальні машини – об'єктами, при цьому ЦОДи мають однаковий розмір. Алгоритми вирішують цю проблему, щоб звести до мінімуму кількість контейнерів при упакуванні всіх об'єктів. Архітектура VirtualPower [29] використовує систему керування живленням на основі локальної та глобальної політики. На локальному рівні система застосовує стратегії керування живленням гостьової операційної системи. Глобальна політика застосовує міграцію віртуальних машин для перерозподілу віртуальних машин. PADD [30] використовує адаптивну схему буферизації для визначення того, скільки потрібно резервної ємності ресурсів. Експерименти у цій роботі показують зменшення витрат електроенергії, коли кількість віртуальних машин збільшується.

У публікаціях [31] розглянуто підхід до управління ресурсами ЦОД, призначений забезпечити ефективне їх використання у процесі функціонування ЦОД за схемою виділених серверів. Наводяться моделі і алгоритми розподілу ресурсів, структура відповідних інструментальних засобів. Описано три варіанти системи, яка надає хостингові послуги на основі моделі виділених серверів: виділені сервери і СОА, виділені сервери і традиційна трирівнева архітектура, гібридна архітектура.

Для моделі планування при надлишку ресурсів наведені чотири задачі мінімізації витрат на підтримку частини серверів, яка забезпечуватиме підтримку запитів користувачів клієнта. Для вирішення наведених вище задач оптимального розміщення екземплярів застосунків пропонуються варіанти генетичного алгоритму (ГА) і евристичного алгоритму, який враховує специфіку критеріїв і обмежень. Крім того, пропонується комбінований алгоритм, побудований на використанні методів вирішення задач лінійного програмування і неявного перебору.

До переваг можна віднести те, що запропоновані алгоритми розміщення були досліджені стосовно одержання розв'язків близьких до оптимального. Розрахунки виконувалися з різними варіантами кількостей фізичних серверів і клієнтів. Результати впровадження показали працездатність евристичних методів та ГА для всіх чотирьох задач. При цьому евристичні алгоритми переважно одержували розв'язок швидше, але з меншим наближенням до оптимуму.

У роботі [32] автори на основі узагальнення накопиченого досвіду розв'язання проблеми розподілу і управління навантаженням і ресурсами ЦОД пропонують модифікації раніше розроблених моделей, які більш вдало враховують реалії цієї цікавої галузі досліджень. З метою розроблення універсального алгоритму для цього класу задач авторами пропонується варіант керованого ГА, що полягає у механізмі отримання нової популяції особин на базі представників минулої епохи. Шляхом налаштування зазначених параметрів керований ГА можна досить швидко навчити розв'язувати різноманітні проблеми зазначеного класу, особливістю яких є складний характер взаємодії критеріїв і обмежень, що раніше часто призводило до передчасного виродження популяції.

Експериментальне дослідження показало, що керований ГА дозволяє отримати покращені результати відносно базового ГА. У більшості випадків керований варіант отримував розв'язки швидше за кількістю епох за базовий. Налаштовуючи систему правил керованого ГА, можна для кожної задачі підібрати схему значень, за якою за орієнтовну кількість кроків (епох) можна досягти розв'язку достатнього рівня оптимальності. Однак, недоліком цього алгоритму є те що пошук таких схем параметрів вимагає витрат часу та експериментальної роботи.

Робота [33] присвячена розробці моделей управління доступом до обмежених інформаційно-обчислювальних ресурсів підприємства чи організації. Проведено аналіз недоліків моделей та обґрунтування ресурсного підходу до управління ІТ-інфраструктурою в умовах недостатності ресурсів. Робота СУІ повинна базуватися на сукупності математичних моделей, які враховують важливість бізнес-процесів, що претендують на використання спільних обмежених ІТ-ресурсів, схемі вибору і

взаємодії моделей з урахуванням поточної ситуації в ІТ-системі, методах управління доступом до ресурсів в умовах їх обмеженості.

Задачі управління, які вирішуються в СУІ при наданні доступу до обмежених ресурсів, було розподілено за шістьма ієрархічними рівнями, безпосередньо пов'язаними з життєвим циклом управління ІТ-інфраструктурою. Для кожного рівня розглянуто математичні моделі і методи розв'язання.

Закладені в основу технології управління ІТ-інфраструктурою методи і моделі управління доступом до обмежених ресурсів, що запропоновані в роботі, а також їх реалізація, дозволяють збільшити ефективність використання обмежених ресурсів ІТ-системи. Адміністратори системи управління ІТ-інфраструктурою можуть отримувати інформацію про стан обладнання, програмного забезпечення елементів підсистем і системи в цілому, попередження про можливі несправності та рекомендації щодо швидкого відновлення працездатності компонентів ІТ-системи. Перевагою є те, що розроблені моделі можуть бути застосовані не тільки при управлінні використанням ІТ-ресурсів, але й на етапах проектування та модернізації ІТ-системи для визначення мінімальної кількості ресурсів, необхідних для підтримки найбільш важливих бізнес-процесів.

У статті [34] запропоновано математичну модель управління перерозподілом ресурсів ІТС, яка побудована з урахуванням того, що значимість бізнес-процесів може змінюватися, а ресурси ІТС обмежені. Передбачається, що компоненти ІТС розподілені по значній території, а система управління функціонуванням ІТС організаційно є територіально-розподіленою ієрархічною багаторівневою системою з центральним вузлом (рівнем), що координує роботу регіональних вузлів і рівнів.

На першому етапі на підставі значущості бізнес-процесів визначаються пріоритети застосунків і виконуваних задач. На підставі цієї інформації формується нова політика управління, яка зберігається в центральному вузлі і реплікується в регіональні вузли управління.

При зміні політики перевизначаються пріоритети виконуваних завдань і встановлюються пріоритети нових завдань. Зміна пріоритетів виконуваних в даний момент задач може позначитися на їх успішному завершенні – виконання завдань

може бути негайно припинено або відкладено. Що стосується нових завдань, то в залежності від їх пріоритету їм взагалі може бути відмовлено в наданні ресурсу. Все це визначається політикою управління, а також реальною ситуацією з ресурсами ІТС.

Далі визначаються ресурси, задіяні виконуваними завданнями, і встановлюються обсяги ресурсів, в яких потребують нові завдання. Після чого відбувається перерозподіл ресурсів з урахуванням пріоритетів завдань і обсягом задіяних і необхідних ресурсів за умови їх обмеженості. Перевагою запропонованого алгоритму виділення ресурсів є те, що він може бути застосований в ієрархічних системах керування.

У публікації [35] запропоновано інформаційну модель системи управління інформаційними технологіями підприємства з точки зору процесу управління проблемами. Запропоновано підхід до організації ефективного управління в рамках методології ITSM (IT Service Management).

Проте в публікації не наводиться рішення проблеми правильної класифікації інцидентів, що виникає на етапі організації ефективного управління ІТ-послугами. Необхідно провести аналіз інформаційних потоків, що надходять від користувачів, провести їх класифікацію з метою вибору типових технологій вирішення проблем, що виникають в роботі систем.

В статті [36] розглянуто проблеми розробки за застосування нового класу автоматизованих систем управління (АСУ), а саме, систем управління інформаційно-технологічними послугами (УІТП) організацій, наведено порівняльний огляд деяких існуючих УІТП-систем і побудовано схему їх типової функціональності.

Для підвищення ефективності застосування таких автоматизованих систем запропоновано розробити модуль інтелектуальної підтримки при вирішенні проблемних ситуацій, що виникають в їх роботі з ІТ-сервісами, який використовує для цього методи логічного виводу на основі аналізу прецедентів. В подальшому планується розробити прототип відповідної УІТП-системи та дослідити ефективність її застосування на прикладі управління ІТ-інфраструктурою ВНЗ.

В публікації [37] розглянута можливість впровадження технології віртуалізації, що дозволяє вирішити багато завдань по вдосконаленню ІТ-інфраструктури ВНЗ.

Рішення, що існують на даний момент, для віртуалізації серверних ресурсів, систем зберігання даних, робочих місць клієнта дають можливість істотно поліпшити ефективність використання устаткування і понизити витрати на його обслуговування. Розглянуто можливі варіанти реалізації, проекти і результати.

Разом з наявністю безперечних переваг технологія віртуалізації, має ряд недоліків, зв'язаних, наприклад, з неможливістю забезпечити максимальну надійність при консолідації серверів, оскільки між віртуальними машинами немає електричної ізоляції. Так збій в роботі операційної системи хоста приводить до необхідності перезавантажувати всі ВМ і їх застосунки. Крім того, ПО віртуалізації є достатньо «важким», таким, що вимагає для свого функціонування наявності відповідних серверних конфігурацій. Недоліком контейнерної віртуалізації є те, що всі ВМ вимушені працювати під управлінням однієї і тієї ж версії ОС. Це не дозволяє, наприклад, запустити Linux і Windows на одній машині.

В статті [38] визначено основні тенденції розвитку технологій віртуалізації, зміни в підходах до надання ІТ-послуг на їх базі, для найбільш ефективного використання в ІТ-середовищах. При використанні технологій віртуалізації забезпечується динамічне і безперервне надання ІТ-послуг відповідно до потреб бізнесу, підтримується висока доступність і безпека даних, а також надається доступ до корпоративних ресурсів мобільних користувачів і співробітників філій. При цьому найповніше використовуються ресурси, а самі ІТ-середовища стають гнучкими, централізованими, масштабованими і незалежними від обладнання.

Серед проблем, які ще потребують вирішення, залишаються важливі проблеми управління ІТ-інфраструктурою в цілому і, зокрема, базова її складова – розподіл і управління ресурсами ЦОД. Тому необхідне проведення досліджень в даній галузі.

З метою розробки алгоритму для розв'язання проблеми розподілу і управління ресурсами пропонується застосувати машинне навчання, а саме метод навчання з підкріпленням – Q -навчання. Цей метод полягає у тому, що агент-менеджер взаємодіє із середовищем у дискретні моменти часу та виконує керуючі впливи для управління віртуалізованими ресурсами. В кожен момент часу агент отримує винагороду. Потім він обирає дію з множини доступних дій, яка відправляється до середовища.

Середовище переходить до нового стану, який знову спостерігає агент щоб визначити наступні керуючі впливи. Метою агента є збирання якомога більшої винагороди.

При проведенні аналізу літературних джерел були оцінені основні методи управління обчислювальними ресурсами в ЦОД і виявлені їх недоліки, основними з яких є:

- тривалий процес пошуку параметрів алгоритму, за яких можна досягти розв'язку близького до оптимального;
- велика розмірність простору станів середовища та керуючих впливів на середовище;
- під час навчання в моделях враховуються тільки показники використання процесора;
- фіксовані пороги використання ресурсів.

Метою дослідження є поліпшення якості управління віртуалізованими обчислювальними ресурсами кластеру хмарного ЦОД шляхом розробки алгоритму управління, що дозволяє зменшити споживання електроенергії та час порушення вимог SLA.

Для досягнення поставленої мети мають бути виконані наступні завдання:

- проаналізувати предметне середовище управління віртуалізованими ресурсами центру обробки даних та провести огляд методів управління обчислювальними ресурсами;
- обрати середовище моделювання центрів обробки даних;
- розробити модель центру обробки даних в обраному середовищі моделювання;
- розробити моделі споживання електроенергії фізичних серверів;
- підготувати дані для моделювання динамічного навантаження віртуальних машин в центрі обробки даних;
- розробити алгоритм НП для управління віртуалізованими обчислювальними ресурсами центру обробки даних;
- виконати програмну реалізацію алгоритму НП та провести дослідження ефективності розробленого алгоритму.

2 ІСНУЮЧІ МЕТОДИ УПРАВЛІННЯ ОБЧИСЛЮВАЛЬНИМИ РЕСУРСАМИ

2.1 Адаптивний евристичний метод консолідації віртуальних машин в центрі обробки даних

2.1.1 Формулювання проблеми міграції віртуальної машини

Нехай маємо один ФС та M ВМ розміщених на цьому ФС. У цій задачі час дискретний і ділиться на N інтервалів, де кожен інтервал складає одну секунду. Постачальник ресурсів оплачує вартість електроенергії, яку споживає ФС. Вартість електроенергії розраховується як $C_p t_p$, де C_p – вартість електроенергії за одиницю часу, а t_p – період часу. Ресурсний потенціал ФС та використання ресурсу ВМ характеризується єдиним параметром – продуктивністю процесора. Використання ресурсу процесора ВМ довільно змінюється з часом. ФС перевантажений, якщо всі ВМ потребують їхню максимально допустиму продуктивність процесора, сумарні вимоги до процесора перевищують продуктивність процесора. Коли вимоги до продуктивності процесора перевищують доступну потужність, то відбувається порушення вимог SLA між постачальником ресурсів та користувачами. Порушення вимог SLA призводить до штрафування постачальника ресурсів. Штраф розраховується як $C_v t_v$, де C_v – вартість порушення вимог SLA за одиницю часу, t_v – період часу порушення вимог SLA. Без втрати загальності можна визначити $C_p = 1$ та $C_v = s$, де $s \in R^+$. Це еквівалентно позначенню $C_p = \frac{1}{s}$ та $C_v = 1$.

В певний момент часу v відбувається порушення вимог SLA та триває до N . Іншими словами, завдяки перевантаженню та зміні робочого навантаження ВМ у момент часу v в цілому вимоги до продуктивності процесора перевищують доступну потужність процесорі і не зменшується до N . Передбачається, що згідно з визначенням проблеми, одна ВМ може мігрувати з ФС. Ця міграція призводить до

зниження вимог до продуктивності процесора. Позначимо n як час закінчення міграції VM або час початку порушення вимог SLA. Міграція VM займає час T . Під час міграції додатковий ФС використовується для розміщення VM що мігрує, і тому сумарне значення споживання електроенергії протягом міграції VM рівне $2C_p T$. Проблема полягає у визначенні часу m , коли необхідно почати міграцію VM, щоб мінімізувати загальну вартість, яка складається з вартості електроенергії і витрат, пов'язаних з порушенням вимог SLA. Нехай r це час що залишився з моменту початку порушення вимог SLA, тобто $r = n - v$.

Функція вартості. Загальна вартість включає в себе витрати за порушення вимог SLA та витрати на додаткове споживання електроенергії. Додаткове споживання електроенергії це електроенергія, яка споживається додатковим ФС, до якого мігрувала VM, і споживання електроенергії основним ФС після початку порушення вимог SLA. Іншими словами, враховується все споживання електроенергії, окрім електроенергії, яка споживається головним ФС від моменту часу t_0 (час початку) до v . Справа в тому, що ця частка електроенергії не може бути усунута жодним алгоритмом відповідно до визначення проблеми. Ще одне обмеження полягає в тому, що порушення вимог SLA не може відбутися до моменту початку міграції t_0 , тобто $v > T$. Відповідно до визначення проблеми, функція вартості $C(v, m)$, матиме вигляд:

$$C(v, m) = \begin{cases} (v - m)C_p, & \text{якщо } m < v, v - m \geq T, \\ (v - m)C_p + 2(m - v + T)C_p + (m - v + T)C_v, & \text{якщо } m \leq v, v - m < T, \\ rC_p + (r - m + v)C_p + rC_v, & \text{якщо } m > v. \end{cases} \quad (2.1)$$

Функція вартості C визначає три випадки, які охоплюють всі можливі зв'язки між v та m . Позначимо випадки (2.1) як C_1 , C_2 , і C_3 відповідно. C_1 описує випадок, коли міграція відбувається до настання порушення вимог SLA ($m < v$), але міграція починається не пізніше T до початку порушення вимог SLA ($v - m \geq T$). У цьому випадку вартість складає лише $(v - m)C_p$, тобто витрати електроенергії, споживаної додатковим ФС, починаючи з початку міграції VM до початку порушення вимог SLA. Вартість порушення вимог SLA відсутня, оскільки, відповідно до визначення

проблеми, час зупинки є початком порушення вимог SLA, так що тривалість порушення вимог SLA становить 0 [23].

C_2 описує той випадок, коли міграція відбувається до початку порушення вимог SLA ($m \leq v$), але міграція починається пізніше ніж T до початку порушення вимог SLA ($v-m < T$). C_2 складається з трьох частин: $(v-m)C_p$ – вартість електроенергії, споживаної додатковим ФС, від початку міграції до початку порушення вимог SLA; $2(m-v+T)C_p$ – вартість електроенергії, споживаної як основним ФС, так і додатковим ФС від початку порушення вимог SLA до n ; $(m-v+T)C_v$ – вартість порушення вимог SLA від початку порушення вимог SLA до кінця міграції ВМ. C_3 описує випадок, коли міграція починається після початку порушення вимог SLA. У цьому випадку вартість складається з трьох частин: rC_p – вартість електроенергії, спожита головним ФС від початку порушення вимог SLA до n ; $(r-m+v)C_p$ – вартість електроенергії, споживаної додатковим ФС, з початку міграції ВМ до n ; rC_v – вартість порушення вимог SLA від початку порушення вимог SLA до n .

Вартість оптимального офлайн алгоритму. Теорема 2.1. Оптимальний офлайн алгоритм для проблеми міграції ВМ покриває вартість $\frac{T}{s}$ і виконується коли $\frac{v-m}{T} = 1$ (доведення теореми дивитися в [23]).

Оптимальний онлайн детермінований алгоритм. У реальному світі алгоритм управління не має інформації про майбутні події, і тому має справу з онлайн проблемою. Проблеми оптимізації, в яких вхідні дані отримуються в режимі онлайн, і в яких вихідні дані повинні отримуватись онлайн, називаються онлайн проблемами. Алгоритми, розроблені для онлайн проблем, називаються онлайн алгоритмами. Один з способів характеризувати продуктивність та ефективність онлайн алгоритмів полягає у застосуванні конкурентного аналізу. В рамках конкурентного аналізу якість онлайн алгоритмів вимірюється відносно найкращої ефективності алгоритмів, які мають повне уявлення про майбутні події. Онлайн-алгоритм ALG є c -конкурентоспроможним, якщо існує константа a , така, що для всіх кінцевих послідовностей I :

$$ALG(I) \leq c \cdot OPT(I) + a, \quad (2.2)$$

де $ALG(I)$ – це вартість отримана ALG з вхідними даними I ;

$OPT(I)$ – вартість оптимального поточного алгоритму з вхідними даними I ;

a – константа.

Це означає, що для всіх можливих вхідних даних ALG отримує вартість в межах постійного фактора c оптимальної офлайн вартості плюс константа a . c може бути функцією параметрів задачі, але вона повинна бути незалежною від вхідних даних I . Якщо ALG є c -конкурентоспроможним, то ALG досягає конкурентного співвідношення c . У конкурентному аналізі онлайн детермінований алгоритм аналізується на вхідних даних, які генерує деякий супротивник. На основі знань онлайн алгоритму супротивник генерує найгірші вхідні дані для онлайн алгоритму, тобто дані, які максимізують конкурентний коефіцієнт. Конфігурація алгоритму – це стан алгоритму відносно зовнішнього середовища, який не слід плутати із внутрішнім станом алгоритму, який складається з його керування та внутрішньої пам'яті.

Теорема 2.2. Конкурентність оптимального онлайн детермінованого алгоритму для проблеми міграції ВМ становить $2+s$, і алгоритм виконується, коли $m = v$ (доведення теореми дивитися в [23]).

2.1.2 Формулювання проблеми динамічної консолідації віртуальних машин

Нехай маємо n однорідних ФС, потужність кожного ФС – A_h . Максимальна потужність процесора, яку можна виділити ВМ становить A_v . Максимальна кількість ВМ на ФС, коли вони вимагають максимальної потужності процесора становить $m = \frac{A_h}{A_v}$. Загальна кількість віртуальних машин становить nm . ВМ можуть мігрувати між ФС, використовуючи живу міграцію з тривалістю міграції t_m . Порушення вимог SLA відбувається, коли сумарні вимоги до продуктивності процесорів перевищує доступну потужність процесора A_h . Вартість електроенергії становить C_p , а вартість

порушення вимог SLA за одиницю часу складає C_v . Без втрати загальності можна визначити $C_p = I$ та $C_v = s$, де $s \in R^+$. Це еквівалентно позначенню $C_p = \frac{I}{s}$ та $C_v = 1$. Коли ФС не працює, тобто не існує розміщених VM, він вимикається і не споживає електроенергії або переходить до сплячого режиму роботи з незначним споживанням електроенергії. Загальна вартість C визначається наступним чином:

$$C = \sum_{t=t_0}^T \left(C_p \sum_{i=0}^n a_{ti} + C_v \sum_{j=0}^n v_{tj} \right), \quad (2.3)$$

де t_0 – початковий час;

T – сумарний час;

$a_{ti} \in \{0,1\}$ – режим роботи i -го ФС у момент часу t ;

$v_{tj} \in \{0,1\}$ – порушення вимог SLA на j -му ФС у момент часу t .

Проблема полягає в тому, щоб визначити в який момент часу яку VM і на який ФС слід перенести, щоб мінімізувати загальну вартість C .

Оптимальний онлайн детермінований алгоритм. Теорема 2.3. Верхня межа конкурентного співвідношення оптимального онлайн детермінованого алгоритму для проблеми динамічної консолідації VM становить $\frac{ALG(I)}{OPT(I)} \leq 1 + \frac{ms}{2(m+1)}$ (доведення теореми дивитися в [23]).

Недетерміновані онлайн алгоритми. Недетерміновані або випадкові онлайн алгоритми, як правило, покращують якість їх детермінованих аналогів. Тому можна очікувати, що коефіцієнт конкурентності онлайн випадкових алгоритмів для проблеми міграції VM, який зводиться до оптимального онлайн детермінованого алгоритму, коли $i \geq v$, лежить між $\frac{T}{s}$ і $2+s$. Аналогічно можна очікувати, що конкурентне співвідношення онлайн випадкових алгоритмів для проблеми динамічної консолідації VM повинно бути покращено порівняно з верхньою оцінкою, визначеною в теоремі 2.3. У конкурентному аналізі аналізуються випадкові алгоритми проти різних типів супротивників, ніж супротивник, який використовується для детермінованих алгоритмів. Наприклад, один з цих

супротивників – це супротивник, який генерує вхідні дані до початку виконання алгоритму. Він генерує дані на основі знань розподілів імовірності, що використовуються алгоритмом. Інший підхід до аналізу випадкових алгоритмів полягає в отриманні результатів алгоритму на основі моделей розподілу вхідних даних. Проте, в реальному середовищі, навантаження на VM, є більш складним і не може бути змодельованим за допомогою простих статистичних розподілів [23].

Вартість живої міграції VM. Жива міграція VM дозволяє переміщати VM між фізичними вузлами без призупинення чи короткого простою. Але жива міграція негативно впливає на ефективність застосунків, що працюють у VM під час міграції. Для застосунків зі змінними робочими навантаженнями, середнє зменшення продуктивності може бути оцінене приблизно на 10% від використання процесора. Крім того кожна міграція може призвести до порушення вимог SLA. Тому важливо мінімізувати кількість міграцій VM. Тривалість міграції в режимі реального часу залежить від загальної кількості пам'яті, що використовується VM та доступною пропускну здатністю мережі. Час міграції та зменшення продуктивності j -ї VM визначається наступним чином:

$$T_{mj} = \frac{M_j}{B_j}, U_{dj} = 0.1 \cdot \int_{t_0}^{t_0+T_{mj}} u_j(t) dt, \quad (2.4)$$

де U_{dj} – загальне зменшення продуктивності j -ї VM;

t_0 – час, коли починається міграція;

T_{mj} – час, необхідний для завершення міграції j -ї VM;

$u_j(t)$ – використання процесора j -ї VM;

M_j – об'єм пам'яті, що використовується j -ю VM;

B_j – доступна пропускну здатність мережі.

Показники порушення вимог SLA. Вимоги QoS зазвичай оформлюються у формі SLA, які можна визначити з точки зору таких характеристик, як мінімальна пропускну спроможність або максимальний час відгуку, наданий розгорнутою системою. Оскільки ці характеристики можуть відрізнятися для різних програм, необхідно визначити незалежний показник робочого навантаження, який може

використовуватися для оцінки SLA будь-якої віртуальної машини, розгорнутою в IaaS. Вимоги SLA виконуються, коли 100% продуктивності, запитуваної застосунками всередині ВМ, надаються у будь-який час і обмежені лише параметрами ВМ. Пропонуються два показники для вимірювання рівня порушень вимог SLA в середовищі IaaS: відсоток часу, протягом якого активні ФС мали 100% використання процесора (SLATAH); загальне зменшення продуктивності ВМ через міграції (PDM):

$$SLATAH = \frac{1}{N} \sum_{i=1}^N \frac{T_{si}}{T_{ai}}, \quad PDM = \frac{1}{M} \sum_{j=1}^M \frac{C_{dj}}{C_{rj}}, \quad (2.5)$$

де N – кількість ФС;

T_{si} – це загальний час, протягом якого ФС мав 100% використання процесора, що призвело до порушення вимог SLA;

T_{ai} – це загальна кількість ФС, що перебувають у активному режимі роботи (обслуговують ВМ);

M – кількість ВМ;

C_{dj} – оцінка зменшення продуктивності j -ї ВМ, обумовлена міграціями;

C_{rj} – загальна потужність процесора, що вимагається j -ю ВМ протягом її роботи.

Комбінований показник порушення вимог SLA (SLAV) розраховується наступним чином [23]:

$$SLAV = SLATAH \cdot PDM. \quad (2.6)$$

2.1.3 Адаптивні евристички для динамічної консолідації віртуальних машин

Розбиваємо проблему динамічної консолідації ВМ на чотири частини: визначення того, коли ФС вважається перевантаженим, що вимагає переміщення однієї або декількох ВМ із цього ФС; визначення, коли ФС вважається недовантаженим, що призводить до перенесення всіх ВМ з цього ФС та його переведення в сплячий режим роботи; вибір ВМ, які повинні мігрувати з

перевантаженого ФС; виявлення нового розташування вибраних ВМ для переміщення з перевантажених та недовантажених ФС.

Загальний алгоритм оптимізації розміщення ВМ переглядає список ФС i , застосовуючи алгоритм виявлення перевантажених ФС, перевіряє, чи є ФС перевантаженим. Якщо ФС перевантажений, алгоритм застосовує політику вибору ВМ, які потрібно перенести з ФС. Після того, як буде побудовано список ВМ, які потрібно перенести з перевантажених ФС, алгоритм розміщення ВМ знаходить нові ФС для розміщення ВМ. Друга частина алгоритму полягає у виявленні недовантажених ФС і переміщенні ВМ з цих ФС. Алгоритм повертає план міграції ВМ, який містить інформацію про нове розташування ВМ вибраних з перевантажених і недовантажених ФС. Складність алгоритму становить $2N$, де N – це кількість ФС [23].

2.1.3.1 Виявлення перевантажених фізичних серверів

Середнє абсолютне відхилення. Фіксовані значення порогів використання процесора непридатні для середовища з динамічним навантаженням. Система повинна мати можливість автоматично регулювати свої дії в залежності від шаблонів навантаження застосунків. Основною ідеєю запропонованих політик є коригування значення верхнього порогу використання процесора в залежності від сили відхилення використання процесора. Чим вище відхилення, тим нижче значення верхнього порогу, чим вище відхилення, тим більша ймовірність того, що використання процесора досягне 100% і призведе до порушення вимог SLA.

Середнє абсолютне відхилення (MAD) є надійнішим, ніж стандартне відхилення, оскільки воно є більш стійким до відхилень у наборі даних, ніж стандартне відхилення.

Для набору даних X_1, X_2, \dots, X_n , MAD визначається наступним чином:

$$MAD = \text{median}_i(|X_i - \text{median}_j(X_j)|). \quad (2.7)$$

Верхній поріг використання процесора визначається наступним чином:

$$T_u = 1 - s \cdot MAD, \quad (2.8)$$

де $s \in R^+$ – параметр, який дозволяє регулювати безпечність політики, чим менше s , тим менше споживання електроенергії, але тим вище рівень порушення вимог SLA, спричинений консолідацією VM.

Міжквартильний діапазон. Міжквартильний діапазон (IQR) є мірою статистичної дисперсії і дорівнює різниці між третьою та першою квартилями: $IQR = Q_3 - Q_1$. Використовуючи IQR, верхній поріг використання процесора визначається наступним чином:

$$T_u = 1 - s \cdot IQR, \quad (2.9)$$

де $s \in R^+$ – параметр, який дозволяє регулювати безпечність політики, чим менше s , тим менше споживання електроенергії, але тим вище рівень порушення вимог SLA, спричинений консолідацією VM.

Локальна регресія. Основна ідея політики – визначити прості моделі локалізованих підмножин даних для побудови кривої, яка наближає вхідні дані. Використовуючи метод Loess [23] для кожного нового спостереження визначається нова лінія тренду $\hat{g}(x) = \hat{a} + \hat{b}x$. Ця лінія тренду використовується для оцінки наступного спостереження $\hat{g}(x_{k+1})$. Політика показує, що ФС перевантажений, а деякі VM повинні мігрувати з нього, якщо виконується наступні нерівності:

$$s \cdot \hat{g}(x_{k+1}) \geq I, \quad x_{k+1} - x_k \leq t_m, \quad (2.10)$$

де $s \in R^+$ – параметр, який дозволяє регулювати безпечність політики;

t_m – максимальний час, необхідний для міграції будь-якої VM.

Надійна локальна регресія. Використовуючи оцінювану лінію тренду та застосовуючи метод локальної регресії для оцінки наступного спостереження, можна визначити, що ФС перевантажений, якщо виконуються нерівності (2.10) [23].

2.1.3.2 Вибір віртуальних машин для міграції

Наступним кроком є вибір конкретних ВМ для переміщення з перевантажених ФС. Після вибору ВМ для переміщення, ФС знову перевіряється на перевантаження. Якщо він все ще вважається перевантаженим, політика вибору ВМ застосовується знову. Це повторюється, поки ФС не буде вважатись перевантаженим.

Політика мінімального часу міграції (ММТ) визначає ВМ v , що вимагає мінімального часу для завершення міграції відносно інших ВМ на ФС. Час міграції визначається шляхом ділення об'єму оперативної пам'яті, що використовується ВМ, на пропускну здатність мережі, доступної для j -го ФС. Нехай V_j це набір ВМ, які розміщені на j -му ФС. Політика ММТ знаходить ВМ, що задовольняє наступні умови:

$$v \in V_j \mid \forall a \in V_j, \frac{RAM_u(v)}{NET_j} \leq \frac{RAM_u(a)}{NET_j}, \quad (2.11)$$

де $RAM_u(a)$ – кількість оперативної пам'яті, яка в даний момент часу використовується ВМ; NET_j – пропускну здатність мережі, доступна для j -го ФС.

Політика випадкового вибору (РС) вибирає ВМ для перенесення відповідно до рівномірно розподіленої дискретної випадкової величини, значенням якої є індекс ВМ, розміщеної на j -му ФС.

Політика максимальної кореляції (МС) базується на ідеї, що чим вище кореляція між використанням ресурсів у застосунках, тим вище вірогідність перевантаження сервера. Згідно з цією ідеєю, для міграції обираються ВМ, які мають найбільшу кореляцію використання процесора порівняно з іншими ВМ.

Політика мінімального використання процесора (МУ) визначає ВМ, яка мінімально використовує процесор відносно інших ВМ на ФС [23].

2.1.3.3 Розміщення віртуальних машин

Розміщення ВМ можна розглядати як проблему упаковки об'єктів в контейнери з різними розмірами та цінами, де контейнери – це ФС; об'єкти – це ВМ, які повинні

бути розміщені; розміри контейнерів – це потужності процесорів ФС; ціни відповідають споживанню електроенергії ФС. Оскільки упаковки об'єктів в контейнери є NP-складною, для її вирішення застосовується модифікований алгоритм Best Fit Decreasing (BFD), який використовує не більше $\frac{11}{9}OPT+1$ контейнерів (OPT – оптимальне рішення). В модифікації алгоритму BFD, всі віртуальні машини сортуються у порядку зменшення їх поточного використання процесора, кожна VM розміщується на ФС, який забезпечує найменший приріст споживання електроенергії після розміщення VM. Складність алгоритму – nm , де n – кількість ФС, а m – кількість VM, які повинні бути розміщені. Алгоритм розміщення VM представлений в Алгоритмі 2.1 [23].

Алгоритм 2.1

Вхід: *pmList*, *vmList*

Вихід: розміщення VM на ФС

```

1.  foreach vm in vmList do
2.      minPower = MAX;
3.      allocatedPM = NULL;
4.      foreach pm in pmList do
5.          if pm має достатньо ресурсів для vm then
6.              power = estimatePower(pm, vm);
7.              if power < minPower then
8.                  allocatedPM = pm;
9.                  minPower = power;
10.             end if
11.          end if
12.      end foreach
13.      if allocatedPM != NULL then
14.          allocation.add(vm, allocatedHost);
15.      end if
16.  end foreach
17.  return allocation;

```

2.2 Метод управління розподілом ресурсів центру обробки даних при віртуальному хостингу

2.2.1 Формулювання проблеми розподілу й управління ресурсами

Нехай в ЦОД прийнята модель віртуального хостингу, яку підтримують технології сервіс-орієнтованої архітектури. Фізичні сервери об'єднані в кластер. Кластерів фізичних серверів може бути декілька. Серверний парк ЦОД обслуговує множину користувачів, надаючи їм доступ до екземплярів множини застосунків. Сервери характеризуються набором технічних параметрів, а застосунки – набором вимог множини своїх користувачів. Дуже потужні застосунки вимагають консолідації ресурсів, тобто декілька серверів забезпечують функціонування одного застосунку. Потрібно розробити математичні моделі та відповідні методи планування і диспетчерування навантаження і розподілу ресурсів фізичних серверів між застосунками за доцільними критеріями ефективності за умови виконання ресурсних, технологічних та інших актуальних обмежень [31].

2.2.2 Математична модель розподілу ресурсів при віртуальному хостингу

Щоб сформулювати задачу оптимального розподілу ресурсів, потрібно ввести такі позначення для найзагальніших понять, показників і параметрів, за допомогою яких можна описати систему, яка надає послуги користувачам, розподіляючи запити екземплярам застосунків на існуючих серверах.

$PM = \{PM_1, \dots, PM_m\}$ – множина серверів, кількість яких дорівнює m .

$A = \{A_1, \dots, A_n\}$ – множина застосунків, кількість яких дорівнює n .

Кожен сервер $PM_i, i = 1, \dots, m$, має чотири параметри, які характеризують його потужність:

- Ω_i – процесорна місткість сервера PM_i ;
- Γ_i – місткість оперативної пам'яті сервера PM_i ;

- Φ_i – місткість жорстких дисків сервера PM_i ;
- L_i – місткість каналів сервера PM_i .

Кожен застосунок $A_j, j = 1, \dots, n$, має вимоги до процесорної місткості ω_j серверів, на яких розташовані екземпляри застосунку A_j ; до оперативної пам'яті γ_j серверів, на яких розташовані екземпляри застосунку A_j ; до місткості жорстких дисків φ_j серверів на яких розташовані екземпляри застосунку A_j ; до місткості каналів λ_j серверів, на яких розташовані екземпляри застосунку A_j .

Характеристики серверів і вимог застосунків поділяються на залежні від навантаження – Ω_i, ω_j і незалежні від навантаження – $\Gamma_i, \gamma_j, \Phi_i, \varphi_j$.

Вектор $C = \{C_1, \dots, C_n\}$, де

$$C_j = \begin{cases} 1, & \text{якщо застосунок } A_j \text{ може розгортатись і згортатись} \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Матриця $R = |R_{ji}|_{n \times m}$, де

$$R_{ji} = \begin{cases} 1, & \text{якщо екземпляр застосунку } A_j \text{ можна розгорнути на сервері } PM_i, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

w_j – важливість застосунку j .

$$x_{ji} = \begin{cases} 1, & \text{якщо екземпляр застосунку } A_j \text{ запущений на сервері } PM_i, \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Величину ω_j можуть забезпечити декілька серверів. Величини γ_j визначають для екземпляра застосунку. Якщо оперативна пам'ять сервера менше γ_j , то екземпляр застосунку на сервері не може бути розміщений [31].

2.2.3 Задачі планування розподілу ресурсів при надлишку ресурсів

Задачу планування розподілу ресурсів доцільно розглядати як задачу визначення матриці $X = \|x_{ji}\|$, яка задає розміщення екземплярів застосунків на серверах.

Оптимальне розміщення повинно враховувати обмеження:

- сумарні вимоги, незалежні від навантаження, визначені для сервера екземплярів застосунків не перевищують незалежні від навантаження параметри сервера;
- вимоги застосунків, залежні від завантаження, визначають на підставі потенціалу усіх користувачів і повинні бути забезпечені сумарними можливостям залежних від завантаження параметрів тих серверів, на яких розміщуються екземпляри цього застосунку;
- обмеження, визначені матрицею R і вектором C .

Задача 1. Якщо для хостингової компанії важлива економія ресурсів, то виникає задача мінімізації витрат на підтримку клієнтських запитів. Тобто вибирається така мінімальна за витратами на їхню підтримку підмножина серверів, які зможуть підтримати потрібну для задоволення вимог існуючих користувачів множину екземплярів застосунків.

Позначимо через s_i експлуатаційні витрати на підтримку сервера PM_i і введемо змінну $y_i, i = 1, \dots, m$, яка може набувати таких значень:

$$x_{ji} = \begin{cases} 1, & \text{якщо сервер } PM_i \text{ задіяний для підтримки одного і більше застосунку,} \\ 0, & \text{якщо сервер } PM_i \text{ не підтримує жодного застосунку.} \end{cases}$$

З використанням введених позначень цільова функція набуде такого вигляду:

$$\sum_{i=1}^m s_i y_i \rightarrow \min. \quad (2.12)$$

Мінімальне значення потрібно обирати з рішень, які відповідають структурі булевих значень матриці $R = \|R_{ji}\|_{n \times m}$ та задовольняють обмеженням:

$$\sum_{j=1}^n x_{ji} \gamma_j \leq \Gamma_i, i = 1, \dots, m. \quad (2.13)$$

Обмеження (2.13) вимагає, щоб місткість оперативної пам'яті сервера PM_i була достатньою для розміщення запланованих на нього екземплярів застосунків.

$$\sum_{j=1}^n x_{ji} \lambda_j \leq \Lambda_i, i = 1, \dots, m. \quad (2.14)$$

Обмеження (2.14) вимагає, щоб місткість каналів зв'язку сервера PM_i була достатньою для розміщення та ефективної роботи усіх екземплярів застосунків, запланованих для нього.

$$\sum_{i=1}^m x_{ji} z_{ji} \Omega_i \leq \omega_j, j = 1, \dots, n, \quad (2.15)$$

$$\sum_{j=1}^n z_{ji} \geq 1, i = 1, \dots, m. \quad (2.16)$$

Обмеження (2.15) і (2.16) вимагають, щоб потреби усіх користувачів застосунку A_j у місткості процесорів задовольняли у сукупності вибрані фізичні сервери з екземплярами цього застосунку. Якщо один застосунок захоплює повністю один чи декілька серверів, то z_{ji} набуває значення 1. Якщо екземпляри декількох застосунків розташовуються на сервері, то z_{ji} визначає частку процесорної місткості сервера PM_i , яка виділяється застосунку A_j .

$$\sum_{i=1}^m x_{ji} \geq 1, j = 1, \dots, n. \quad (2.17)$$

Обмеження (2.17) вимагає, щоб для кожного застосунку був хоча б один екземпляр на якомусь сервері.

$$\sum_{j=1}^n x_{ji} \phi_j \leq \Phi_i, i = 1, \dots, m. \quad (2.18)$$

Обмеження (2.18) вимагає, щоб місткість жорсткого диску сервера PM_i була достатньою для роботи усіх екземплярів застосунків, запланованих для нього.

Задача 2. Якщо для хостингової компанії важливе рівномірне завантаження серверів, то виникає задача мінімізації сумарного недовантаження фізичних серверів за умови виконання обмежень на ресурси і технологічних обмежень. З використанням введених вище позначень цільову функцію цієї задачі можна записати у наступному вигляді:

$$\sum_{i=1}^m \left(\Omega_i - \sum_{j=1}^n \omega_j x_{ji} \right) \rightarrow \min . \quad (2.19)$$

Мінімальне значення потрібно обирати з рішень, які відповідають структурі булевих значень матриці $R = |R_{ji}|_{n \times m}$ та задовольняють обмеження (2.13) - (2.18) та обмеження (2.20):

$$\sum_{j=1}^n x_{ji} \geq 1, i = 1, \dots, m . \quad (2.20)$$

Обмеження (2.20) вимагає, щоб кожний сервер містив хоча б один екземпляр якогось застосунку.

Задача 3. Якщо для хостингової компанії важлива економія ресурсів, а витрати на реалізацію переходу до нового оптимального плану розміщення застосунків порівнювані за витратами на підтримку клієнтських запитів, то виникає задача мінімізації сумарних витрат на підтримку клієнтських запитів і реалізацію переходу до нового оптимального плану розміщення застосунків. Тобто вибирається така мінімальна за сумою витрат на реалізацію переходу до нового оптимального плану розміщення застосунків і на їхню підтримку підмножина серверів, які зможуть підтримати необхідну для задоволення вимог існуючих користувачів множину екземплярів застосунків. З використанням введених вище позначень цільову функцію цієї задачі можна записати у наступному вигляді:

$$\sum_{i=1}^m \left(s_i y_i + \sum_{j=1}^n p_{ji} |x'_{ji} - x_{ji}| \right) \rightarrow \min . \quad (2.21)$$

У (2.21) p_{ji} позначає витрати пов'язані з переходом до нового оптимального плану розташування застосунків при його відмінності від старого плану для застосунку A_j на сервері PM_i , x'_{ji} – значення змінної x_{ji} у старому плані.

Мінімальне значення потрібно обирати з рішень, які відповідають структурі булевих значень матриці $R = |R_{ji}|_{n \times m}$ та задовольняють обмеження (2.13) - (2.18).

Задача 4. Якщо для хостингової компанії важливе рівномірне завантаження серверів, а витрати на реалізацію переходу до нового оптимального плану розміщення застосунків порівнювані за втратами від недовантаження серверів до повної потужності, то виникає задача мінімізації сумарних витрат від рівномірного недовантаження фізичних серверів та витрат на реалізацію переходу до нового оптимального плану розміщення застосунків, за умови виконання обмежень на ресурси і технологічних обмежень. З використанням введених вище позначень цільову функцію задачі можна записати у вигляді (2.22):

$$\sum_{i=1}^m \left(\delta_i \frac{\sum_{j=1}^n \omega_j x_{ji}}{\Omega_i} + \sum_{j=1}^n p_{ij} |x'_{ji} - x_{ji}| \right) \rightarrow \min. \quad (2.22)$$

У (2.22) δ_i позначає втрати, пов'язані з недовантаженням сервера PM_i .

Мінімальне значення потрібно обирати з рішень, які відповідають структурі булевих значень матриці $R = |R_{ji}|_{n \times m}$ та задовольняють обмеження (2.13) - (2.18), (2.20).

Наведені вище задачі оптимізації розміщень для одного кластера – це лінійні та нелінійні варіанти дискретної задачі пакування рюкзака. У випадку декількох кластерів вони становлять варіанти дискретної задачі обмеженого

кластерами пакування декількох рюкзаків. Ці задачі є NP-повними. Оскільки обійти перебірні методи їхнього вирішення не можна, то виправданим буде використання евристичних методів або методів “м’яких” обчислень насамперед генетичних алгоритмів, застосування яких дає контрольоване наближення до точного результату [31].

2.2.4 Генетичний алгоритм розв'язання задачі

Варіанти генетичного алгоритму. Розглянемо універсальний варіант генетичного алгоритму, який можна модифікувати для кожної з наведених вище задач:

Крок 0. Визначити $i=0$. Утворення випадкової початкової популяції $P_k(0) = \{p_1(0), \dots, p_k(0)\}$ з урахуванням можливого відсіювання під час перевірки обмежень;

Крок 1. Оцінити кожен ланцюжок популяції щодо виконання обмежень відповідної задачі, наприклад, для задачі 1 обмежень (2.13) - (2.17). Для тих ланцюжків $p_q(0)$, для яких обмеження виконуються, правильна оцінка $f(p_q(0))$ цільової функції відповідної задачі, наприклад, для задачі 1 це $\sum_{i=1}^m s_i y_i$, і вибір найкращого рішення;

Крок 2. Відбираємо представників для нової популяції (ланцюжків з найкращим значенням функції $f(p_q(0))$);

Крок 3. $l=l+1$. Утворюємо нову популяцію застосовуючи оператор рекомбінації;

Крок 4. Поліпшуємо популяцію застосовуючи оператор мутації;

Крок 5. Оцінюємо кожний ланцюжок популяції щодо виконання обмежень і функцію $f(p_q(l))$, якщо обмеження виконуються. Перевірка обмежень (2.15) і (2.16) виконується в межах одиничних значень одержаної матриці X методом північно-західного кута;

Крок 6. Якщо рішення краще попереднього, то завершуємо виконання алгоритму, в іншому випадку повертаємося на крок 3.

Вибираємо найліпше рішення в утвореній популяції $P_k(l)$.

Тут кожен ланцюжок будь-якої популяції становить двійковий вектор довжини, $t = n \times m$ що визначається кількістю фізичних серверів і застосунків. Елемент q вектора набуває значення 0 або 1 відповідних змінних x_{ji} . Довжина ланцюжків постійна.

Відбір особини для нової популяції виконується на підставі ранжування за значенням цільової функції особин попередньої популяції, причому ранжуються лише особини, які задовольняють ресурсні та інші обмеження.

Для рекомбінування застосовують одноточковий кросинговер з випадковим вибором точки кросинговеру. Реалізація мутацій відбувається на основі схеми інверсії ділянок ланцюжків [31].

2.2.5 Евристичний алгоритм розв'язання задачі

Евристичні алгоритми ґрунтуються на інтуїтивних уявленнях, які враховують роль залежних і незалежних характеристик, з яких найкритичніші – оперативна пам'ять, ресурсів у вигляді критерію. Оскільки процесорну місткість можна забезпечити сукупно декількома серверами, а оперативну пам'ять лише одним сервером, то буде логічним спершу розмістити застосунки, які мають вищі потреби в оперативній пам'яті, ніж у процесорному часі. Приймаючи це за правило, далі треба лише врахувати особливості критерію. Кожному фізичному серверу PM_i припишемо коефіцієнт P_i , значенням якого приймемо відношення процесорної місткості сервера PM_i до місткості його оперативної пам'яті, кожному застосунку A_j – коефіцієнт ρ_j , значенням якого приймемо відношення вимог застосунку A_j до процесорної місткості до вимог застосунку до місткості оперативної пам'яті сервера.

Попередній етап для будь-якого з наведених нижче варіантів евристичного алгоритму полягає у сортуванні фізичних серверів за значенням їхнього коефіцієнта P у порядку зростання і сортуванні застосунків за значенням їхнього коефіцієнта ρ у порядку спадання.

Евристичний алгоритм для задачі 1. Попередній етап виконано.

Крок 1. Пошук застосунку, який має найменший коефіцієнт ρ ;

Крок 2. Пошук сервера з найбільшим коефіцієнтом P ;

Крок 3. Якщо для вибраних сервера і застосунку не виконуються обмеження задачі 1, то повертаємося на крок 2 для пошуку наступного сервера;

Крок 4. Якщо для вибраних сервера і застосунку виконуються обмеження задачі 1, причому залежні від навантаження вимоги повністю задоволені, то переходимо на крок 5. Якщо виконуються обмеження задачі 1, але сервер не може повністю задовольнити залежні від навантаження вимоги застосунку, то переходимо на крок 6;

Крок 5. Застосунок вилучається зі списку застосунків, коригується коефіцієнт P сервера і він вбудовується в список серверів відповідно до нового значення коефіцієнта. Перехід до кроку 7;

Крок 6. Ресурси сервера, що залишилися, застосовують і сервер вилучається зі списку серверів. Переходимо до кроку 2;

Крок 7. Якщо список застосунків порожній, то кінець алгоритму, інакше переходимо до кроку 1.

Обчислювальна складність евристичного алгоритму становить $n \times m$. Для інших критеріїв використовують іншу логіку вибору серверів у їхньому впорядкованому списку [31].

2.3 Метод автоконфігурування віртуальних машин на основі навчання з підкріпленням

2.3.1 Формулювання проблеми автоконфігурування віртуальних машин

Якщо VM створюється з конфігурацією за замовчуванням (з шаблону) або мігрує з іншого ФС, то для VM може знадобитися переконфігурування для кращої продуктивності на новому ФС. Через різні вимоги застосунків до ресурсів, зазвичай, необхідно перерозподіляти ресурси для кожної розміщеної VM для збільшення загальної продуктивності. У консолідації служб із різнорідними застосунками нелегко визначити найкращі параметри для VM з різними потребами в ресурсах. Сучасні монітори VM забезпечують багатий набір параметрів ресурсів, які можна налаштувати, що значно ускладнює проблему [10].

2.3.2 Навчання з підкріпленням для автоконфігурування віртуальних машин

НП полягає в прийнятті рішень агентом у динамічному середовищі, щоб максимізувати довгострокову винагороду. НП має дві переваги: НП не потребує моделі будь-якої системи або динаміки навколишнього середовища; НП здатна відстежувати вплив наслідків від дій під час вирішення завдання.

Результат НП – це політика, яка відображає поточний стан, в якому агент знаходить найкращі дії. Оптимальність дії в стані вимірюється ціннісною функцією, яка оцінює суму майбутніх нагород після виконання цієї дії. Агент НП виконує взаємодію із середовищем шляхом проб та помилок, після кожної дії середовище повертає миттєву винагороду. Оптимальна політика – вибрати дію, яка максимізує ціннісну функцію у кожному стані. Дії поділяються на такі що дотримуються оптимальної політики, та такі що обираються випадковим чином.

Розглянемо агента НП як контролера ВМ. Стани середовища – це розподіл ресурсів ВМ, а можливі зміни в розподілі ресурсів – це дії. Середовище являє собою динамічну віртуалізовану платформу. Щоразу, коли контролер налаштовує конфігурацію ВМ, він отримує відгук про продуктивність від окремих ВМ. Після дій контролер отримує хороші оцінки рішень розподілу, що отримуються за допомогою поточних конфігурацій ВМ. Починаючи з довільних параметрів, контролер може керувати ВМ для оптимальної конфігурації, дотримуючись оптимальної політики. Завдяки дослідженням, контролер може змінити свою політику розподілу ресурсів відповідно динамічному навантаженню на ВМ.

Завдання НП зазвичай моделюється як Марківський процес прийняття рішень (англ. Markov decision process, MDP). Для набору станів середовища S та набору дій A , MDP визначається імовірністю переходу $P_a(s, s') = P_r(s_{t+1} = s' | s_t = s, a_t = a)$, дія a в стані s в момент часу t призведе до стану s_{t+1} в момент часу $t+1$ і негайної функції винагороди $R = E[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$. На кожному кроці t агент сприймає поточний стан $s_t \in S$ і доступний набір дій $A(s_t)$. Виконуючи дію на $a_t \in A(s_t)$, агент переходить

до наступного стану s_{t+1} і отримує негайну винагороду r_{t+1} від середовища. Значення ціннісної функції від прийняття дії a у стані s визначається наступним чином:

$$Q(s,a) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}, \quad (2.23)$$

де $0 \leq \gamma < 1$ – коефіцієнт знецінювання, який допомагає збіжності $Q(s,a)$.

Функція винагороди. Довгострокова сукупна винагорода – це мета НП. У завданні конфігурації ВМ оптимальні конфігурації – це ті, які оптимізують загальносистемну продуктивність. Винагороди є результатами отримання нової конфігурації ВМ. Продуктивність окремих віртуальних машин вимірюється балом, який є співвідношенням поточної пропускної здатності $thrpt$ та еталонної пропускної здатності плюс можливі штрафи коли час відгуку $resp$ порушує вимоги SLA:

$$\begin{aligned} score &= \frac{thrpt}{ref_thrpt} - penalty, \\ penalty &= \begin{cases} 0, & \text{якщо } resp \leq SLA, \\ \frac{resp}{SLA}, & \text{якщо } resp > SLA. \end{cases}, \end{aligned} \quad (2.24)$$

де ref_thrpt – максимальна доступна продуктивність застосунків в межах вимог SLA.

Тоді винагорода – це сума оцінок по усім ВМ:

$$reward = \begin{cases} \sqrt{\prod_{i=1}^n w_i \cdot score_i}, & \text{якщо } \forall score_i > 0, \\ -1, & \text{в іншому випадку.} \end{cases}, \quad (2.25)$$

де w_i – коефіцієнт важливості i -ї ВМ.

Простір станів. У завданні конфігурування ВМ простір станів відповідає конфігураціям ВМ. Стани детерміновані, тобто $P_a(s,s') = 1$, що спрощує завдання НП. Станом є загальний розподіл ресурсів:

$$(mem_1, time_1, vcpu_1, \dots, mem_n, time_n, vcpu_n),$$

де mem_i – об'єм оперативної пам'яті i -ї ВМ;

$time_i$ – процесорний час i -ї ВМ;

$vcpi_i$ – кількість процесорів i -ї ВМ.

Дії. Кожен з трьох зазначених вище параметрів можна збільшити, зменшити або не виконувати жодної дії. Таким чином, дії визначаються як комбінації операцій за кожним параметром. Дія недійсна, вона призводить до порушення обмежень стану. Дія виконується лише над одним параметром. Це дозволяє не пропускати жодного стану середовища [10].

2.3.3 Онлайн алгоритм навчання з підкріпленням

В якості моделі середовища виступає нейронна мережа з сигмоїдною активаційною функцією та лінійним виходом. Важливим є те, що легко контролювати структуру та складність мережі, змінюючи кількість прихованих шарів і кількість нейронів у кожному шарі. Це полегшує інтеграцію алгоритму НП для кращої збіжності. Апроксиматор Q -функції нейронної мережі на вхід отримує пару стан-дія та повертає апроксимоване Q -значення. У Алгоритмі 2.2 представлено онлайн алгоритм НП. На кожному інтервалі конфігурації, алгоритм фіксує попередній стан та отримує винагороду. Наступна дія обирається за допомогою ε -жадібною політики відповідно до вихідних даних апроксиматора Q -функції. Нові значення (s_t, a_t, r_{t+1}) оновлюють модель середовища. Апроксиматор Q -функції наведений в Алгоритмі 2.3 [10].

Алгоритм 2.2

1. Ініціалізувати Q_{appx} для навчання апроксиматора функції.
2. Ініціалізувати $t = 0$, $a_t = NULL$.
3. **repeat**
4. $s_t = get_current_state();$
5. $re_configure(a_t);$
6. $r_{t+1} = observe_reward();$
7. $a_{t+1} = get_next_action(s_t, Q_{appx});$
8. $workload = identify_workload();$
9. $Rmodel \leftarrow select_model(workload);$

10. $update\ Rmodel(s_t, a_t, r_{t+1}, Rmodel);$
11. $update\ Qappx(Rmodel, Qappx);$
12. $t = t + 1;$
13. **until** алгоритм не зупинено;

Алгоритм 2.3

1. Ініціалізувати $Qappx$ для навчання поточного апроксиматора функції.
2. **repeat**
3. $sse = 0;$
4. **for** n iterations **do**
5. $(s_t, a_t, r_t) = generate_sample(Rmodel);$
6. $target = r_t + \gamma * Qappx(s_{t+1}, a_{t+1});$
7. $error = target - Qappx(s_t, a_t);$
8. $sse = 0.9 * sse + 0.1 * error * error;$
9. $train\ Qappx(s_t, a_t)\ towards\ target;$
10. **end for**
11. **until** $converge(sse);$

2.4 Висновок до розділу

У даному розділі були розглянуті існуючі методи управління обчислювальними ресурсами. До таких методів було віднесено:

- адаптивний евристичний метод консолідації віртуальних машин в центрі обробки даних;
- метод управління розподілом ресурсів центру обробки даних при віртуальному хостингу;
- метод автоконфігурування віртуальних машин на основі навчання з підкріпленням.

Для кожного методу розглянуто постановку задачі та алгоритм розв'язання поставленої задачі. Розроблюваний метод розміщення ВМ на основі НП буде порівняно адаптивними евристичними методами консолідації ВМ в ЦОД.

3 РОЗРОБКА МЕТОДУ ДИНАМІЧНОГО РОЗМІЩЕННЯ ВІРТУАЛЬНИХ МАШИН НА ОСНОВІ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

3.1 Задача зменшення споживання електроенергії та часу порушення вимог SLA в центрі обробки даних

Розглянемо центр обробки даних як постачальника віртуальних ресурсів, який складається з m ФС з різною конфігурацією. Кожен ФС має процесор, який може бути багатоядерним, при цьому продуктивність ядер вимірюється в мільйонах інструкцій на секунду (англ. Million instructions per second, MIPS). Крім того, ФС характеризується обчислювальною потужністю процесора, об'ємом оперативної пам'яті, пропускнуою здатністю мережі та ємністю жорсткого диску. Користувачі відсилають завдання або запити для створення n ВМ. Спочатку ВМ розміщуються на ФС відповідно до запитуваних характеристик. Відсоток використання ресурсів ВМ з часом буде змінюватись. Необхідно розмістити ВМ на мінімальній кількості ФС для зменшення витрат електроенергії та часу порушення вимог SLA.

Для управління віртуалізованими ресурсами ЦОД пропонується безмодельний варіант агента, що заснований на методі НП Q -навчання [2]. Агент НП може отримати розв'язок близький до оптимального шляхом взаємодії з динамічним середовищем без будь-якої інформації про це середовище. Структура методу НП складається з:

- простору станів S – сукупність станів середовища, які агент може сприймати;
- простору дій A – сукупність дій, які агент може виконати;
- винагороди r , яка є або позитивною реакцією середовища, або штрафом (негативна реакція середовища) за виконану агентом дію. Таким чином, метою агента є зведення до мінімуму середніх довгострокових штрафів впродовж процесу навчання.

Q -навчання – це один з найпопулярніших методів НП, який застосовується у багатьох галузях досліджень. На кожній ітерації алгоритму Q -навчання агент отримує дані про поточний стан середовища $s_t \in S$ та вибирає дію $a_t \in A$, що впливає на

середовище. Після виконання дії середовище переходить до наступного стану $s' \in S$, і агент отримує винагороду p_t . На початку наступної ітерації агент оновлює Q -значення на основі наступного рівняння:

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot [p_t + \gamma \cdot \min_{a' \in A} Q(s', a')], \quad (3.1)$$

де $Q(s_t, a_t)$ – очікуваний довгостроковий штраф за виконання дії $a_t \in A$ в стані $s_t \in S$;

α – швидкість (темп) навчання (англ. learning rate), коефіцієнт, що показує наскільки швидко нові дані про стани будуть враховуватись на наступних кроках, $\alpha \in (0, 1]$;

γ – коефіцієнт знецінювання (англ. discount factor), коефіцієнт, що визначає важливість майбутніх винагород, $\gamma \in [0, 1]$;

$\min_{a' \in A} Q(s', a')$ – оцінка оптимального майбутнього Q -значення.

Якщо $\alpha = 0$, то агент не навчається, якщо $\alpha = 1$, то агент використовує дані про найновіші управляючі впливи. Якщо $\gamma = 0$, то агент розглядає лише поточні винагороди, якщо $\gamma = 1$, то агент прагне до довгострокового мінімального штрафу.

Коли агент знову отримає стан $s_t \in S$, він вибере дію з мінімальним Q -значенням. Політика π вибору найкращої дії у стані $s_t \in S$ вираховується наступним рівнянням:

$$\pi(s_t) = \min_{a \in A} Q(s_t, a). \quad (3.2)$$

Таким чином, метою агента НП є знаходження оптимальної політики відображення $S \rightarrow A$, яка мінімізує очікуваний довгостроковий штраф за виконані дії. Щоб вибрати керуючу дію, агент може використати один з двох методів: вибір випадкової дії на початку процесу навчання або під час отримання нового стану; вибір дій, визначених політикою π .

Алгоритм Q -навчання має наступні етапи:

Крок 1. Для кожної пари $s \in S$ та $a \in A$, ініціалізувати Q -значення, що дорівнює нескінченності;

Крок 2. Отримати поточний стан $s_t \in S$;

Крок 3. Обрати дію $a_t \in A$ випадковим чином або за допомогою (3.2);

Крок 4. Виконати дію $a_t \in A$;

Крок 5. Отримати винагороду p_t ;

Крок 6. Отримати новий стан $s' \in S$ та оновити $Q(s_t, a_t)$ за допомогою (3.1);

Крок 7. Зберегти новий стан як поточний, $s \leftarrow s'$;

Крок 8. Повернутися до кроку 3.

Використання НП має переваги та недоліки при вирішенні проблеми управління ресурсами ЦОД. Основна перевага НП – це адаптація до інтенсивних змін робочих навантажень та різні умови використання ресурсів в ЦОД. У той же час, недоліки навчання підкріплення – це низька швидкість конвергенції та обмеження розміру простору станів.

Завдяки процесам вивчення та експлуатації управління ресурсами, агент НП зближується з найкращою політикою контролю, переглянувши існуючу політику контролю у відповідь на зміну навколишнього середовища. Але швидкість конвергенції може не відповідати вимогам SLA, тому що агенту навчання необхідно виконати певну кількість дослідницьких дій для збору інформації про середовище. Часто агент витрачає багато часу, щоб зібрати найкращі Q -значення, адаптуючись для виконання керуючих впливів в нових станах системи. Це неприйнятно у промисловому ЦОД через ймовірність збільшення кількості порушень SLA. З іншого боку, розмір простору станів істотно впливає на кількість Q -значень і розмір Q -таблиці. Це призводить до (i) зростання кількості взаємодій, необхідних для збору штрафів за всі пари стан-дія, і (ii) збільшення часу, необхідного для збору штрафів, для заповнення Q -таблиці. Таким чином, у промисловому ЦОД необхідно вдосконалити агент НП, за допомогою додаткових евристик.

3.2 Математична модель центру обробки даних

ЦОД орієнтований на обробку пакетних та транзакційних завдань. Вони вирішуються в рамках трьох основних хмарних моделей IaaS, PaaS та SaaS. Модель

IaaS є основною для обслуговування сучасних ЦОД. Зберігання даних забезпечується централізованим сховищем даних.

Забезпечення ресурсами здійснюється на основі їх наявності для уникнення перерозподілу ресурсів ЦОД. При цьому зміни клієнтських вимог до ресурсів можуть збільшуватися до певного рівня не впливаючи на інших клієнтів та у відповідності до SLA. Сучасна практика перерозподілу ресурсів з метою попередження небажаних наслідків, спричинених браком ресурсів із збільшенням навантаження та потреб клієнтів, призводить до збільшення споживання енергії та експлуатаційних витрат.

Модель ЦОД показана на рисунку 3.1. Структура складається з набору ФС, кожен з яких характеризується апаратною та операційною платформами і має фіксовану кількість ресурсів. [6]

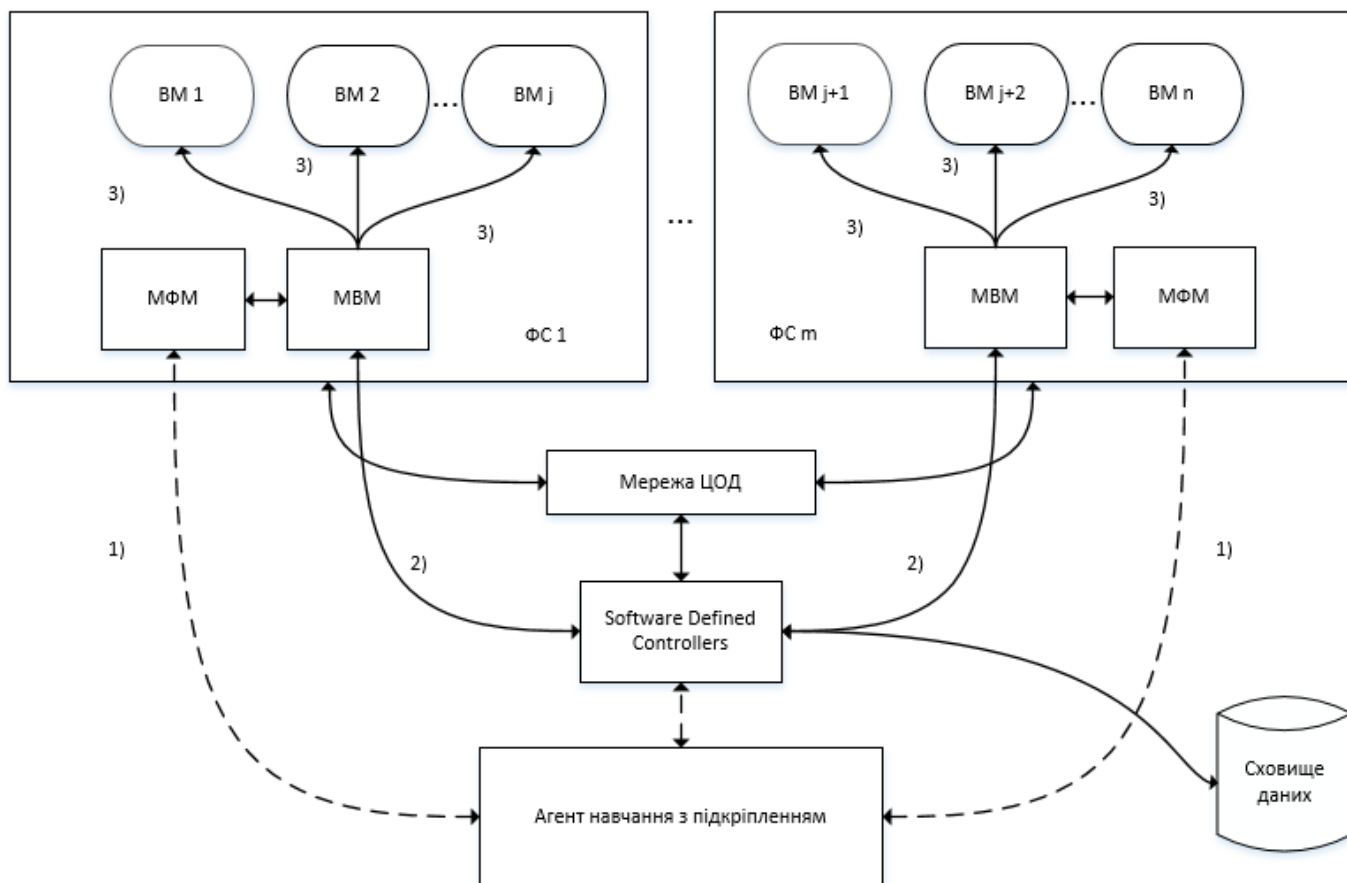


Рисунок 3.1 – Модель ЦОД

Кожен ФС може містити певну кількість VM. Агент НІ розглядається як частина Adaptive Software Defined Manager, представлена в [39]. Послідовність операцій алгоритму динамічного розміщення VM:

1) Спостереження агентом навчання за поточним станом ФС. Агент НП отримує інформацію про поточні стани ФС від менеджерів фізичних машин (МФМ), після надходження завдань або запитів на розміщення VM від користувачів. Ця інформація являє собою поточне використання ресурсів ФС. Після цього агент обирає режим роботи для ФС на основі Q -навчання.

2) Надсилання керуючих впливів до моніторів віртуальних машин (МВМ). Оптимізація розміщення VM відбувається в залежності від рішення агента щодо режиму роботи ФС. Якщо агент обрав сплячий режим, то всі VM на ФС повинні мігрувати до інших ФС. Вибір VM для міграції з перевантаженого ФС відбувається відповідно до політики вибору. Політика полягає у виборі VM, яка вимагає мінімального часу міграції (VM має найменший об'єм оперативної пам'яті), ніж інші VM на ФС. Час міграції розраховується шляхом ділення пам'яті, призначеної для VM, на доступну пропускну здатність мережі між поточною і цільовою ФС. Після вибору VM відбувається пошук цільової ФС.

3) Команди міграції VM. МВМ надсилають команди міграції до VM, які повинні мігрувати на інші ФС.

Наступні змінні визначені для динамічної моделі ЦОД між моментами часу t та $t+1$ [40].

Режим роботи i -го ФС позначається як $z_i(t) \in \{0,1\}$, $i = \overline{1,m}$, якщо $z_i(t) = 0$, то i -ий ФС знаходиться в сплячому режимі, якщо $z_i(t) = 1$, то i -ий ФС знаходиться в активному режимі роботи.

Використання k -го ресурсу j -ою VM на i -му ФС позначається як $res_{ij}^k(t) \in [0,1]$ $i = \overline{1,m}$, $j = \overline{1,n}$, $k \in K$, де K це кількість типів ресурсів, таких як обчислювальна потужність процесора, об'єм оперативної пам'яті, пропускну здатність мережі. Значення $res_{ij}^k(t)$ нормалізується відносно найбільшого об'єму k -го ресурсу ФС. Використання k -го ресурсу на i -му ФС позначається як $Res_i^k(t) \in [0,1]$, $i = \overline{1,m}$, $k \in K$.

Змінна $w_{ij}^h(t) \in \{0,1\}$, $h = \overline{1, H}$ показує, чи працює j -а ВМ типу h на i -му ФС ($w_{ij}^h(t) = 1$) чи не працює ($w_{ij}^h(t) = 0$). H це кількість типів ВМ. Тип j -ї ВМ, що працює на i -му ФС позначається як $h_{ij}(t)$.

Об'єм k -го ресурсу, необхідний для j -ої ВМ, позначається як $c_j^k(t) \in [0,1]$ та нормалізується відносно найбільшого об'єму k -го ресурсу ФС. Використання k -го ресурсу j -ою ВМ не повинно перевищувати необхідну ємність $res_j^k(t) \leq c_j^k(t)$.

Об'єм k -го ресурсу i -го ФС позначається як $C_i^k(t) \in [0,1]$ та нормалізується відносно найбільшого об'єму k -го ресурсу ФС. Значення нормалізується таким чином, що найбільший об'єм k -го ресурсу ФС рівний 1.

Позначимо через $u_{ij}(t) \in \{0,1\}$ міграцію j -ої ВМ з i -го ФС. Міграція відбулася тоді, коли $u_{ij}(t) = 1$.

Позначимо через $x_{ij}(t) \in \{0, \dots, m\}$ індекс ФС до якої мігрує j -а ВМ з i -го ФС. Якщо j -а ВМ не мігрує, тоді $x_{ij}(t) = i$.

Позначимо через $y_i(t) \in \{0, \dots, n\}$ індекс ВМ, яка мігрує з i -го ФС до ФС з індексом $x_{ij}(t)$. Якщо j -а ВМ не мігрує, тоді $y_i(t) = j$.

Позначимо через $a_i(t) \in \{-1, 0, 1\}$ зміну режиму роботи i -го ФС. Якщо $a_i(t) = -1$, тоді i -ий ФС потрібно перевести до сплячого режиму, якщо $a_i(t) = 1$, тоді i -ий ФС потрібно перевести до активного режиму, якщо $a_i(t) = 0$, тоді режим роботи i -го ФС не потрібно змінювати.

Динаміка i -го ФС може бути виражена наступним чином:

$$z_i(t+1) = z_i(t) + a_i(t). \quad (3.3)$$

Динаміка j -ої ВМ в i -му ФС може бути виражена наступним чином:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t)(1 - u_{ij}(t)) + w_{x_{ij}(t)y_{ij}(t)}(t)u_{ij}(t), \\ u_{ij}(t) &= 0 \mid w_{ij}(t) = 0. \end{aligned} \quad (3.4)$$

Динаміка використання k -го ресурсу j -ої ВМ в i -му ФС може бути виражена наступним чином:

$$res_{ij}^k(t+1) = w_{ij}(t)(res_{new}^k(t) + res_{ij}^k(t))(1 - u_{ij}(t)) + res_{x_{ij}(t)y_{ij}(t)}^k(t)u_{ij}(t). \quad (3.5)$$

Використання ресурсів i -го ФС для кожного ресурсу k повинно бути обмеженим. Таким чином, повинна виконуватись умова (3.6):

$$\sum_{j=1}^N res_{ij}^k(t) \leq C_i^k. \quad (3.6)$$

Кількість ФС, що працюють в поточний момент часу t (3.7):

$$M_{PM}(t) = \sum_{i=1}^m z_i(t) \rightarrow \min. \quad (3.7)$$

$M_{PM}(t) \leq m$. Мінімальне значення $M_{PM}(t)$ свідчить про кращу якість розташування ВМ, мінімальне споживання електроенергії та час порушення вимог SLA.

Кількість ВМ, що розгорнуто в поточний момент часу t (3.8):

$$N_{VM}(t) = \sum_{i=1}^m \sum_{j=1}^n w_{ij}(t) z_i(t). \quad (3.8)$$

$N_{VM}(t) < n$. Кількість ВМ, що знаходяться в стані міграції в поточний момент часу t (3.9):

$$N_{mig}(t) = \sum_{i=1}^m \sum_{j=1}^n u_{ij}(t) z_i(t). \quad (3.9)$$

$N_{mig}(t) < n$. Кількість ВМ, що розгорнуто в поточний момент часу t в i -му ФС позначимо як $N_i(t) = \sum_{j=1}^n w_{ij}(t)$.

Метою управління віртуалізованими ресурсами ЦОД є керування режимами роботи ФС через змінну $a_i(t)$, визначення значень змінних розташування ВМ $x_{ij}(t)$, $y_{ij}(t)$ та значення змінної міграцій $u_{ij}(t)$ для мінімізації загального штрафу з точки зору споживання електроенергії та кількості порушень SLA.

3.3 Метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням

Метою запропонованого методу є зменшення витрат електроенергії в ЦОД та зменшення часу порушення вимог SLA. SLA включає в себе вимоги та обмеження щодо якості послуг, що надаються провайдером: часу відгуку, доступності, пропускну здатності, безпеки та ін. SLA виконується для кожного клієнта, коли вся продуктивність, яку потребують застосунки всередині VM, забезпечується в будь-який час.

VM розміщуються на ФС відповідно до поточних затребуваних ресурсів. Коли використання ресурсів на ФС є низьким, всі VM перерозподіляються на інші ФС, а недостатньо завантажений ФС переходить до сплячого режиму. Крім того, коли ФС перевантажений, деякі VM повинні бути переміщені, щоб зменшити час порушення вимог SLA. Якість алгоритму динамічного розміщення VM може поліпшуватися в процесі роботи агента НП та алгоритму Q -навчання для визначення режиму роботи кожного ФС (сплячий або активний).

Алгоритм може динамічно адаптувати ФС до зміни робочого навантаження. Важливою частиною алгоритму є вирішення питання про те, чи додатковий ФС повинен забезпечити ефективне використання ресурсів зі збільшенням робочого навантаження, або резервні ФС можуть бути переведені в сплячий режим або чи достатня поточна кількість ФС. Агент НП вважається важливою частиною алгоритму для прийняття такого рішення.

ФС, які задіяні в процесі оптимізації, позначаються наступними мітками:

HIBERNATE – позначаються ФС, які потрібно перевести до сплячого режиму;

PLACEMENT – позначаються ФС, які задіяні в процесі розміщення VM;

AVAILABLE – позначаються ФС, які доступні для процесу розміщення VM.

Алгоритм динамічного розміщення VM на основі НП наведено у додатку А, плакат 1.

Алгоритм розподілу ВМ наведено у додатку А, плакат 2. На вхід алгоритм отримує список L^{VM} з ВМ, які треба розподілити по ФС. Результатом роботи алгоритму є карта міграцій ВМ до ФС в ЦОД.

Вибирається список L^{PM} , який містить ФС з міткою AVAILABLE. До кожного ФС із списку L^{PM} надсилаються вимоги до обчислювальної потужності процесора, об'єму оперативної пам'яті, пропускну здатності мережі вибраної ВМ зі списку L^{VM} . Якщо ФС має ресурси для ВМ і за результатами прогнозування ФС не буде перевантажений після розміщення ВМ, то такий ФС позначається міткою PLACEMENT. ФС та ВМ додаються до карти міграцій і ВМ видаляється зі списку L^{VM} .

Виконується декілька перевірок перед тим, як ФС підтвердить можливість розміщення ВМ. Повинна виконуватись вимога (3.10) для кожного ресурсу:

$$Res_{PM}^{AVAIL} > Res_{VM}, \quad (3.10)$$

де Res_{PM}^{AVAIL} – доступна кількість ресурсів ФС;

Res_{VM} – вимоги до кількості ресурсів з боку ВМ.

ФС починає процес розміщення ВМ, якщо виконується вимога (3.11) для кожного ресурсу:

$$Res_{PM}^{MAX} > Res_{PM}^P, \quad (3.11)$$

де Res_{PM}^{MAX} – максимальна кількість ресурсів ФС;

Res_{PM}^P – прогнозоване навантаження.

Рівняння (3.12) визначає поточну ємність ЦОД, яка використовується для визначення динаміки використання ресурсів:

$$CAP_{Res}^{DC} = \frac{\sum_{j=1}^n Res_{VM}^j}{\sum_{i=1}^m Res_{PM}^i}, \quad (3.12)$$

де n – кількість ВМ в певний проміжок часу;

m – кількість ФС;

Res_{VM}^j – вимоги до кількості ресурсів j -ї ВМ;

Res_{PM}^i – поточне використання ресурсів i -го ФС.

3.4 Модель агента навчання з підкріпленням

Ефективний метод розміщення повинен зменшити кількість активних ФС відповідно до поточного навантаження на ФС ЦОД. Потрібно приймати рішення про те, коли перевести ФС в сплячий або активний режим роботи. З цієї причини пропонується агент НП як важлива частина алгоритму динамічного розміщення ВМ. Агент обирає режим ФС на основі попередніх даних і отримує винагороду при зміні стану середовища. З цією метою агент вивчає політику виявлення режиму роботи ФС на вхідних запитах та налаштовує відповідну політику за допомогою Q -навчання.

В момент часу t агент сприймає поточний стан ФС, виконує пошук дії за допомогою алгоритму Q -навчання та відправляє знайдену дію на виконання до МФМ (рисунок 3.1). Кожен елемент з простору станів S представляє середнє значення поточної завантаженості процесорів, об'єму оперативної пам'яті, пропускної здатності мережі усіх ФС:

$$s_t = \{CPU_{PM}, RAM_{PM}, BW_{PM}\},$$

де $CPU_{PM} \in [0,1]$ – середнє значення завантаженості процесора усіх ФС;

$RAM_{PM} \in [0,1]$ – середнє значення задіяного об'єму оперативної пам'яті;

$BW \in [0,1]$ – середнє значення показника завантаженості мережі.

Для зменшення кількості станів показник використання кожного ресурсу округляється до двох десяткових знаків.

Агент виконує дію на основі спостережуваного стану середовища. Простір дій визначається як набір $A = \{a_1(t), a_2(t), \dots, a_i(t), a_m(t)\}$, де $a_i(t) \in \{-1, 0, 1\}$, $i = \overline{1, m}$. Кожна дія з A переводить ФС у сплячий або активний режим роботи або залишає режим роботи ФС без змін до наступного інтервалу часу $t+1$. Алгоритм динамічного розміщення ВМ переводить кожен ФС у режим роботи на основі рішення агента. Потім агент отримує штраф та вираховує Q -значення після зміни режиму роботи ФС до початку

інтервалу часу $t+1$, але після завершення виконання обраної дії. Основною метою алгоритму динамічного розміщення ВМ є зменшення споживання електроенергії та часу порушення вимог SLA, обчислюючи значення штрафу p_t (3.13), яке складається з двох значень: штраф за порушення SLA та штраф за споживання електроенергії.

$$p_t = \beta \cdot p_t^{SLA} + \delta \cdot p_t^{power}, \quad (3.13)$$

де β та δ – ваги, що визначають відносну важливість p_t^{SLA} та p_t^{power} відповідно ($\beta + \delta = 1$).

Досягнення бажаних вимог QoS є надзвичайно важливим для середовища хмарних обчислень. Вимоги QoS зазвичай визначаються в термінах SLA, які описують такі характеристики, як пропускна спроможність або час відгуку. Оскільки ці характеристики можуть змінюватися для різних застосунків, необхідно визначити метрику, яка не залежить від робочого навантаження і може бути використана для оцінки SLA будь-якої ВМ, що розгортається в ЦОД.

Якщо необхідна кількість ресурсу процесора j -ої ВМ на i -му ФС $res_{ijr}^{CPU}(t) \in [0,1]$ більше ніж виділена кількість ресурсу процесора j -ої ВМ на i -му ФС $res_{ija}^{CPU}(t) \in [0,1]$ для виконання завдання, то це вважається порушенням SLA:

$$res_{ijr}^{CPU}(t) > res_{ija}^{CPU}(t).$$

Штраф за порушення SLA розраховується шляхом ділення сумарного часу порушення вимог SLA після виконання дії $a_t \in A$ на сумарний час порушення вимог SLA на попередньому інтервалі часу перед виконанням дії:

$$p_t^{SLA} = \sum_{i=1}^m \left(\frac{SLA_i(t)}{SLA_i(t-1)} \right), \quad (3.14)$$

де m – кількість ФС в ЦОД;

$SLA_i(t)$ – сумарний час порушення вимог SLA i -го ФС після виконання дії $a_t \in A$;

$SLA_i(t-1)$ – сумарний час порушення вимог SLA i -го ФС на попередньому інтервалі часу перед виконанням дії $a_t \in A$.

Якщо $p_i^{SLA} < 1$ то це означає, що агент вибрав правильну дію, щоб зменшити час порушення вимог SLA.

Штраф за збільшення споживання електроенергії розраховується шляхом ділення значення споживання електроенергії в поточному інтервалі часу на значення споживання електроенергії попереднього інтервалу часу. Отже, p_i^{power} являє собою загальну суму споживання електроенергії усіх ФС:

$$p_i^{power} = \sum_{i=1}^m \left(\frac{power_i(t)}{power_i(t-1)} \right), \quad (3.15)$$

де m – кількість ФС в ЦОД;

$power_i(t)$ – значення споживання електроенергії i -им ФС після виконання дії $a_t \in A$;

$power_i(t-1)$ – значення споживання електроенергії i -им ФС на попередньому інтервалі часу перед виконанням дії $a_t \in A$.

Усі розміщення та розподіли ВМ, а також зміна режиму роботи ФС, повинні завершитися до початку наступного інтервалу управління $t+1$. Тоді Q -значення для кожної пари стан-дія часового інтервалу оновлюється через загальну суму штрафів P_t .

Q -значення пари стан-дія $Q(s_t, a_t)$ являє собою очікувані сумарні витрати електроенергії та часу порушення вимог SLA, спричинені дією, прийнятою у стані $s_t \in S$. Коли агент наступного разу спостерігає за станом, він вибирає режим роботи ФС, який забезпечує мінімальне Q -значення. Агент НП вибере дію, що має найменше Q -значення (час порушення вимог SLA і витрати електроенергії). Алгоритм роботи агента НП на інтервалі управління від t до $t+1$ має наступні етапи:

Крок 1. Для кожної пари $s \in S$ та $a \in A$, ініціалізувати Q -значення, що дорівнює нескінченності (виконується один раз на початку процесу управління);

Крок 2. Отримати поточний стан $s_t \in S$;

Крок 3. Обрати дію $a_t \in A$ випадковим чином або за допомогою (3.2);

Крок 4. Виконати дію $a_t \in A$;

Крок 5. Обчислити штраф за порушення SLA p_t^{SLA} за допомогою рівняння (3.14);

Крок 6. Обчислити штраф за збільшення споживання електроенергії p_t^{power} за допомогою рівняння (3.15);

Крок 7. Обчислити сумарний штраф p_t після зміни режиму роботи, використовуючи рівняння (3.13);

Крок 8. Отримати новий стан $s' \in S$ та оновити значення $Q(s_t, a_t)$ за допомогою рівняння (3.1).

Крок 9. Зберегти новий стан як поточний, $s \leftarrow s'$;

Крок 10. Повернутися до кроку 3.

Під час початку процесу навчання та кожного разу, коли агент не відвідував поточний стан раніше, пропонується застосовувати дії, що враховують нижнє або верхнє порогові значення використання ресурсів, замість вибору Q -значень, що дорівнюють нескінченності. Поріг є більш ефективним ніж випадковий вибір у звичайному Q -навчанні. Якщо використання ресурсів ФС не перевищує 40% від загальної кількості доступних ресурсів, то агент переводить ФС в сплячий режим роботи.

3.5 Висновок до розділу

В даному розділі наведено постановку досліджуваної задачі зменшення споживання електроенергії та часу порушення вимог SLA в центрі обробки даних та математичну модель центру обробки даних.

Розроблено метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням та наведено модель агента навчання з підкріпленням.

Результати розділу були опубліковані в [7].

4 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Інформаційне забезпечення

4.1.1 Вхідні дані

Вхідними даними є:

- показники продуктивності ВМ, а саме показники використання процесора, оперативної пам'яті, мережі;
- технічні характеристики ВМ та ФС.

Показники продуктивності містяться в n файлах (n – кількість ВМ) з розширенням .csv в наступному форматі:

$$\begin{array}{ccc} \text{cpu}_1 & \text{ram}_1 & \text{bw}_1 \\ \text{cpu}_2 & \text{ram}_2 & \text{bw}_2 \\ \dots & \dots & \dots \\ \text{cpu}_k & \text{ram}_k & \text{bw}_k \\ \dots & \dots & \dots \\ \text{cpu}_p & \text{ram}_p & \text{bw}_p \end{array}$$

де p – кількість інтервалів часу;

$\text{cpu}_k \in [0,1]$ – середній показник використання процесора за інтервал часу;

$\text{ram}_k \in [0,1]$ – середній показник використання оперативної пам'яті за інтервал часу;

$\text{bw}_k \in [0,1]$ – середній показник використання мережі за інтервал часу.

Кожен файл містить p рядків з трьома показниками продуктивності ВМ. Стовпці рядка розділені за допомогою ";"\t".

На рисунку 4.1 наведено фрагмент вмісту CSV файлу з вхідними даними.

	E	G	K
2	1.0	184547.2	1.0666666666666667
3	1.0	234879.2	1.2
4	1.0	184548.0	1.0
5	1.2	176160.0	1.2666666666666666
6	1.0	92272.8	1.0
7	1.1333333333333333	201326.13333333333	1.2666666666666666
8	1.0	162178.13333333333	1.0
9	1.2	274025.06666666665	1.2666666666666666
10	1.0	201324.0	1.0
11	1.2	206917.86666666667	1.2666666666666666
12	1.0	125828.0	1.0
13	1.2	198529.33333333334	1.2666666666666666
14	1.0	128624.26666666666	1.0666666666666667
15	1.2	204120.0	1.2
16	0.9333333333333332	153790.66666666666	1.0
17	1.2	159381.86666666667	1.2
18	1.0	150993.86666666667	1.0
19	1.0666666666666667	198528.26666666666	1.2
20	1.0	134216.53333333333	1.0
21	1.0666666666666667	192937.06666666668	1.2
22	0.8666666666666667	192935.73333333334	1.0
23	1.1333333333333333	153789.06666666668	1.2
24	1.0	106254.13333333333	1.0

Рисунок 4.1 – Фрагмент вмісту CSV файлу з вхідними даними

Технічні характеристики VM та ФС містяться в машинозчитуваному форматі ключ-значення в файлі з розширенням .json. В таблиці 4.1 наведений опис ключів, які містяться у файлі.

Таблиця 4.1 – Опис ключів, які містяться у JSON файлі з характеристики VM та ФС

Ключ	Опис
inputFolder	Значення типу String, шлях до директорії, в якій знаходяться директорії з файлами, що містять показники продуктивності VM. Наприклад, "F:\\workload\\bitbrains"
experimentName	Значення типу String, ім'я директорії, в якій знаходяться файли з показниками продуктивності VM. Наприклад, "trace_rnd_8"
schedulingInterval	Значення типу int, інтервал часу в секундах. Наприклад, 300
simulationLimit	Значення типу int, час моделювання роботи ЦОД в секундах. Наприклад, 777600
learningRate	Значення типу double, від 0 до 1 включно, швидкість (темп) навчання. Наприклад, 0.5

Продовження таблиці 4.1

Ключ	Опис
discountFactor	Значення типу double, від 0 до 1 включно, коефіцієнт знецінювання. Наприклад, 0.5
cofImportanceSla	Значення типу double, від 0 до 1 включно, коефіцієнт що визначає відносну важливість штрафу за порушення SLA. Наприклад, 0.5
cofImportancePower	Значення типу double, від 0 до 1 включно, коефіцієнт що визначає відносну важливість штрафу за споживання електроенергії. Наприклад, 0.5
vmTypesCount	Значення типу int, кількість типів ВМ. Наприклад, 4
vmMips	Масив значень типу int (кількість значень рівна vmTypesCount), кількість мільйонів інструкцій за секунду, які виконує обчислювальний елемент (ядро процесора ВМ). Наприклад, [500, 1300, 2200, 2200]
vmPes	Масив значень типу int (кількість значень рівна vmTypesCount), кількість обчислювальних елементів (ядер процесора ВМ). Наприклад, [2, 4, 8, 16]
vmRam	Масив значень типу int (кількість значень рівна vmTypesCount), об'єм оперативної пам'яті ВМ в мегабайтах. Наприклад, [2048, 4096, 8192, 16384]
vmBw	Значення типу int, пропускна здатність мережі ВМ в кілобітах за секунду (кілобайт = 1000 байт). Наприклад, 100000
vmSize	Значення типу int, розмір образу ВМ в мегабайтах (кілобайт = 1000 байт). Наприклад, 20000
hostTypesCount	Значення типу int, кількість типів ФС. Наприклад, 4
hostTypes	Масив значень типу String, модель ФС. Наприклад, ["DellPowerEdgeR640", "DellPowerEdgeR740", "DellPowerEdgeR830", "DellPowerEdgeR940"]
hostMips	Масив значень типу int (кількість значень рівна hostTypesCount), кількість мільйонів інструкцій за секунду, які виконує обчислювальний елемент (ядро процесора ФС). Наприклад, [2500, 2500, 2200, 2500]
hostPes	Масив значень типу int (кількість значень рівна hostTypesCount), кількість обчислювальних елементів (ядер процесора ФС). Наприклад, [56, 56, 88, 112]
hostRam	Масив значень типу int (кількість значень рівна hostTypesCount), об'єм оперативної пам'яті ФС в мегабайтах. Наприклад, [196608, 196608, 262144, 393216]
hostBw	Значення типу int, пропускна здатність мережі ФС в кілобітах за секунду (кілобайт = 1000 байт). Наприклад, 10000000
hostStorage	Значення типу int, розмір сховища даних ФС в мегабайтах (кілобайт = 1000 байт). Наприклад, 8000000
hostsCount	Значення типу int, кількість ФС в ЦОД. Наприклад, 200

Приклад вмісту JSON файлу з технічними характеристиками ВМ та ФС:

```
{
  "inputFolder": "F:\\Diss\\Projects\\Diss\\workload\\bitbrains",
  "experimentName": "trace_rnd_8",
```

```

"schedulingInterval": 300,
"simulationLimit": 777600,

"learningRate": 0.5,
"discountFactor": 0.5,
"cofImportanceSla": 0.5,
"cofImportancePower": 0.5,

"vmTypesCount": 4,
"vmMips": [500, 1300, 2200, 2200],
"vmPes": [2, 4, 8, 16],
"vmRam": [2048, 4096, 8192, 16384],
"vmBw": 100000,
"vmSize": 20000,

"hostTypesCount": 4,
"hostTypes": ["DellPowerEdgeR640", "DellPowerEdgeR740", "DellPowerEdgeR830",
"DellPowerEdgeR940"],
"hostMips": [2500, 2500, 2200, 2500],
"hostPes": [56, 56, 88, 112],
"hostRam": [196608, 196608, 262144, 393216],
"hostBw": 10000000,
"hostStorage": 8000000,
"hostsCount": 200
}

```

4.1.2 Вихідні дані

Вихідними даними є:

- поточний час моделювання;
- час порушення вимог SLA по усім ФС;
- показники споживання електроенергії ЦОД;
- кількість міграцій ВМ.

Вихідні дані містяться у файлі з розширенням .csv в наступному форматі:

$$\begin{array}{cccc}
 time_1 & sla_1 & power_1 & migCount_1 \\
 time_2 & sla_2 & power_2 & migCount_2 \\
 \dots & \dots & \dots & \dots \\
 time_k & sla_k & power_k & migCount_k , \\
 \dots & \dots & \dots & \dots \\
 time_p & sla_p & power_p & migCount_p
 \end{array}$$

де p – кількість інтервалів часу;

$time_k \in [0, time_p]$ – час моделювання після k інтервалів часу;

$sla_k \in [0, +\infty]$ – сумарний час порушення вимог SLA по усім ФС за k інтервалів часу;

$power_k \in [0, +\infty]$ – показник споживання електроенергії ЦОД за k інтервалів часу;

$migCount_k \in [0, +\infty]$ – кількість міграцій ВМ за k інтервалів часу;

Файл містить p рядків з чотирма показниками. Стовпці рядка розділені за допомогою ";".

На рисунку 4.2 наведено фрагмент вмісту CSV файлу з вихідними даними.

	A	B	C	D
1	300.10000	0.000000	0.000000	0.000000
2	600.10000	58.982400	1.232480	3.000000
3	900.10000	75.366400	2.457904	5.000000
4	1200.1000	75.366400	3.657733	5.000000
5	1500.1000	75.366400	4.856650	5.000000
6	1800.1000	75.366400	6.059206	5.000000
7	2100.1000	75.366400	7.263263	5.000000
8	2400.1000	75.366400	8.463404	5.000000
9	2700.1000	75.366400	9.661961	5.000000
10	3000.1000	75.366400	10.855509	5.000000
11	3300.1000	75.366400	12.047619	5.000000
12	3600.1000	75.366400	13.246040	5.000000
13	3900.1000	75.366400	14.452354	5.000000
14	4200.1000	85.196800	15.652621	7.000000
15	4500.1000	85.196800	16.851808	7.000000
16	4800.1000	85.196800	18.048243	7.000000
17	5100.1000	85.196800	19.243131	7.000000
18	5400.1000	85.196800	20.438555	7.000000
19	5700.1000	85.196800	21.634503	7.000000
20	6000.1000	85.196800	22.829322	7.000000
21	6300.1000	85.196800	24.021964	7.000000
22	6600.1000	95.027200	25.208638	9.000000
23	6900.1000	95.027200	26.396037	9.000000

Рисунок 4.2 – Фрагмент вмісту CSV файлу з вихідними даними

Також до вихідних даних відносяться дані, які генеруються в процесі моделювання роботи ЦОД та зберігаються у файлі з розширенням .log.

4.2 Програмне та технічне забезпечення

4.2.1 Засоби розробки

Для програмної реалізації методу динамічного розміщення VM на основі НП використано мову Java, фреймворк для моделювання хмарної інфраструктури та сервісів CloudSim 4.0 [41], інтегроване середовище програмування IntelliJ IDEA [42].

На відміну від інших фреймворків, CloudSim дозволяє здійснювати повноцінне моделювання і симуляцію хмарних обчислювальних систем, інфраструктур, сховищ даних, веб-сервісів, розподілу ресурсів між віртуальними машинами та ін. При розробці моделі хмарного середовища, користувачу необхідно здійснити доопрацювання ключових для своєї моделі компонентів для досягнення результатів, які будуть максимально наближені до реальності.

Найбільш важливими компонентами при моделюванні є компоненти, які відповідають за політику управління ресурсами. До задач, які вирішують ці компоненти відносяться наступні:

- виділення процесорних потужностей, оперативної пам'яті та інших ресурсів для різних сутностей системи, що моделюється;
- розгортання віртуальних машин на вузлах системи, що моделюється;
- розподіл задач між віртуальними машинами системи, що моделюється.

Принцип роботи моделі передбачає доопрацювання необхідних базових компонентів платформи та попередньої характеристики системи, що моделюється та сценарію проведення симуляції у вигляді вихідного коду. Після запуску симуляції всі дані про систему, що моделюється передаються в ядро CloudSim, де проводиться симуляція.

CloudSim не передбачає змін в системі, що моделюється або сценаріях проведення симуляції безпосередньо під час функціонування моделі, що накладає деякі обмеження на можливості фреймворка.

IntelliJ IDEA підтримує інструменти (у вигляді плагінів) для проведення тестування, системи контролю версій (наприклад, Git), засоби автоматизації роботи з програмними проектами (наприклад, Maven).

До переваг IntelliJ IDEA можна віднести:

- статистичний аналіз коду при завантаженні чи при його вводиті;
- виявлення можливих проблем та пропонування можливих варіантів вирішення цих проблем;
- розумна ергономіка;
- вбудовані інструменти.

4.2.2 Вимоги до технічного забезпечення

4.2.2.1 Загальні вимоги

Технічні характеристики робочих комп'ютерів користувачів повинні відповідати вимогам до середовища виконання Java-застосунків Java SE Runtime Environment 8 (Microsoft Windows) [43], яке повинне бути інстальоване на робочому комп'ютері користувача з операційною системою Microsoft Windows Vista, 7, 8.

4.2.3 Архітектура програмного забезпечення

4.2.3.1 Діаграма класів

У додатку А, плакаті 5 представлена схема структурна класів ПЗ, які відповідають за виконання таких функцій, як розбір JSON файлу з технічними характеристиками ВМ та ФС, створення усіх об'єктів в ЦОД (ВМ, ФС, брокера, задач для виконання віртуальними машинами), збереження даних процесу моделювання,

збереження вихідних даних в CSV файл, запуск моделювання роботи ЦОД, оптимізацію розміщення ВМ на основі НП, оптимізацію розміщення ВМ за допомогою евристичних методів, відображення графічного інтерфейсу користувача, візуалізації отриманих результатів моделювання ЦОД.

Схема містить 22 класи, а саме:

- Main – клас, що реалізує графічний інтерфейс користувача та візуалізацію отриманих результатів моделювання ЦОД;
- ParseConfig – клас, що відповідає за розбір JSON файлу з технічними характеристиками ВМ та ФС;
- Runner – клас, що ініціює моделювання ЦОД;
- SetupEntities – клас, що відповідає за створення об'єктів в ЦОД (ВМ, ФС, брокера, задач для виконання віртуальними машинами);
- VmAllocationPolicyMigrationAgent – клас, що реалізує алгоритм розподілу ВМ;
- HostPowerModeSelectionPolicyAgent – клас, що реалізує алгоритм динамічного розміщення ВМ на основі НП;
- UtilizationModelBitbrains – клас, що реалізує модель використання ресурсів ВМ на основі даних зібраних з розподіленого ЦОД Bitbrains [44];
- UtilizationModelPlanetLab – клас, що реалізує модель використання ресурсів ВМ на основі даних зібраних за допомогою дослідницької мережі PlanetLab [45];
- DellPowerEdgeR640 – клас, що реалізує модель споживання електроенергії ФС Dell Inc. PowerEdge R640 при різних показниках використання процесора [46];
- DellPowerEdgeR740 – клас, що реалізує модель споживання електроенергії ФС Dell Inc. PowerEdge R740 при різних показниках використання процесора [47];
- DellPowerEdgeR830 – клас, що реалізує модель споживання електроенергії ФС Dell Inc. PowerEdge R830 при різних показниках використання процесора [48];

- DellPowerEdgeR940 – клас, що реалізує модель споживання електроенергії ФС Dell Inc. PowerEdge R940 при різних показниках використання процесора [49];
- VmAllocationPolicyNonPowerAware – клас, що реалізує політику розподілу ВМ, яка не оптимізує розподіл ВМ (не енергоекономна робота ЦОД);
- PowerVmAllocationPolicyMigrationInterQuartileRange – клас, що реалізує політику розподілу ВМ, яка використовує міжквартильний діапазон для виявлення перевантаженого ФС;
- PowerVmAllocationPolicyMigrationLocalRegression – клас, що реалізує політику розподілу ВМ, яка використовує локальну регресію для прогнозування завантаженості ФС і виявлення перевантаженого ФС;
- PowerVmAllocationPolicyMigrationLocalRegressionRobust – клас, що реалізує політику розподілу ВМ, яка використовує надійну локальну регресію для прогнозування завантаженості ФС і виявлення перевантаженого ФС;
- PowerVmAllocationPolicyMigrationMedianAbsoluteDeviation – клас, що реалізує політику розподілу ВМ, яка використовує середнє абсолютне відхилення для виявлення перевантаженого ФС;
- PowerVmAllocationPolicyMigrationStaticThreshold – клас, що реалізує політику розподілу ВМ, яка використовує статичне порогове значення використання ресурсів для виявлення перевантаженого ФС;
- PowerVmSelectionPolicyMaximumCorrelation – політика вибору ВМ для міграції, яка вибирає ВМ з максимальним коефіцієнтом кореляції;
- PowerVmSelectionPolicyMinimumMigrationTime – політика вибору ВМ для міграції, яка вибирає ВМ з мінімальним часом міграції;
- PowerVmSelectionPolicyMinimumUtilization – політика вибору ВМ для міграції, яка вибирає ВМ з мінімальним використанням процесора;
- PowerVmSelectionPolicyRandomSelection – політика вибору ВМ для міграції, яка випадково вибирає ВМ.

Модифіковані класи фреймворка CloudSim 4.0:

- CloudletScheduler;

- CloudletSchedulerDynamicWorkload;
- CloudletSchedulerSpaceShared;
- CloudletSchedulerTimeShared;
- NetworkCloudletSpaceSharedScheduler;
- PowerHostUtilizationHistory;
- Vm;
- PowerVm;
- PowerVmAllocationPolicyMigrationAbstract;
- PowerVmAllocationPolicyMigrationInterQuartileRange;
- PowerVmAllocationPolicyMigrationLocalRegression;
- PowerVmAllocationPolicyMigrationMedianAbsoluteDeviation;
- PowerVmAllocationPolicyMigrationStaticThreshold;
- PowerVmSelectionPolicyMaximumCorrelation;
- PowerDatacenter;
- UtilizationModelPlanetLab.

4.2.3.2 Діаграма послідовності

У додатку А, плакаті 6 представлена схема структурна послідовності динамічного розміщення ВМ на основі НП. Агент НП отримує показники порушення вимог SLA та споживання електроенергії по усім ФС. Потім агент отримує інформацію про поточний стан фізичних серверів та зберігає його. Обчислюється штраф та нове Q -значення. Після цього агент обирає режим роботи для ФС на основі Q -навчання. Якщо агент обрав сплячий режим, то всі ВМ на ФС мігрують до неперевантажених ФС. Для міграції з перевантаженого ФС вибирається ВМ, яка вимагає мінімального часу міграції (ВМ має найменший об'єм оперативної пам'яті), ніж інші ВМ на ФС. Після вибору ВМ відбувається пошук цільової ФС.

4.2.3.3 Специфікація функцій

Специфікація функцій наведена в таблиці 4.2.

Таблиця 4.2 – Специфікація функцій

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Main	createDataset	Створення набору даних	XYDataset	List<Double> timeList, List<Double> list, String key	Дані осі X, дані осі Y, назва набору даних
Main	plotSavedDatasets	Візуалізація збережених даних	-	-	-
Main	resetCharts	Стерти графіки	-	-	-
Main	plotCharts	Візуалізація графіків	-	String chartName, List<Double> timeList, List<Double> slaList, List<Double> powerList, List<Double> migrationCount List	Списки з показниками
Main	setUpPanels	Ініціалізація панелей	-	-	-
Main	setUpButtons	Ініціалізація кнопок	-	-	-
Main	setUpLabels	Ініціалізація написів	-	-	-
Main	setUpComboBox	Ініціалізація випадаючого списку	-	-	-
Main	setUpCharts	Ініціалізація систем координат	-	-	-

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Main	getExceptionMessage	Отримання повідомлення про помилку	String	Exception e	Виключення
Main	main	Головний метод	-	String[] args	Аргументи командного рядка
ParseConfig	getData	Розбір JSON файлу	-	String configName	Ім'я JSON файлу
Runner	initLogOutput	Збереження даних процесу моделювання	-	String experimentName, String policyName	Назва експерименту, назва алгоритму
Runner	printResults	Збереження вихідних даних в файл	-	String experimentName, String policyName	Назва експерименту, назва алгоритму
Runner	init	Ініціалізація моделювання	-	String experimentFolder	Ім'я директорії з вхідними даними
Runner	start	Початок моделювання	-	String experimentName, VmAllocationPolicy vmAllocationPolicy, String policyName	Назва експерименту, політика розміщення ВМ, назва алгоритму
Runner	nonPowerAwareModelling	Моделювання ЦОД в енергоекономічному режимі роботи	-	String inputFolder, String experimentName, String policyName	ім'я директорії з вхідними даними, назва експерименту, назва алгоритму

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Runner	getVmAllocationPolicy	Отримати політику розподілу ВМ	VmAllocationPolicy	String vmAllocationPolicyName, String vmSelectionPolicyName	Назва політики розподілу ВМ, назва політики вибору ВМ
Runner	getVmSelectionPolicy	Отримати політику вибору ВМ	PowerVmSelectionPolicy	String vmSelectionPolicyName	Назва політики вибору ВМ
SetupEntities	createVmList	Створення ВМ	List<Vm>	int brokerId, int vmsCount	Id брокера, кількість ВМ
SetupEntities	createHostList	Створення ФС	List<PowerHost>	int hostsCount	Кількість ФС
SetupEntities	createBroker	Створення брокера	DatacenterBroker	-	-
SetupEntities	createDatacenter	Створення ЦОД	Datacenter	String name, Class<? extends Datacenter> datacenterClass, List<PowerHost> hostList, VmAllocationPolicy vmAllocationPolicy	Назва ЦОД, список ФС, політика розподілу ВМ
SetupEntities	createCloudletList	Створення задач для ВМ	List<Cloudlet>	int brokerId, String inputFolderName	Id брокера, ім'я директорії з вхідними даними
VmAllocationPolicyMigrationAgent	optimizeAllocation	Оптимізація розміщення ВМ	List<Map<String, Object>>	List<? extends Vm> vmList	Список ВМ
VmAllocationPolicyMigrationAgent	getHostPowerMode	Отримання режиму роботи ФС	int[]	-	-

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
VmAllocationPolicyMigrationAgent	printOverUtilizedHosts	Вивести перевантажені ФС	-	List<PowerHostUtilizationHistory> overUtilizedHosts	Перевантажені ФС
VmAllocationPolicyMigrationAgent	findHostForVm	Знайти ФС для ВМ	PowerHost	Vm vm, Set<? extends Host> excludedHosts	ВМ, список ФС
VmAllocationPolicyMigrationAgent	isHostOverUtilizedAfterAllocation	Перевірка перевантаження ФС після розміщення на ньому ВМ	boolean	PowerHost host, Vm vm	ФС, ВМ
VmAllocationPolicyMigrationAgent	getNewVmPlacement	Отримання плану розміщення ВМ	List<Map<String, Object>>	List<? extends Vm> vmsToMigrate, Set<? extends Host> excludedHosts	ВМ для міграції, виключені з розгляду ФС
VmAllocationPolicyMigrationAgent	getNewVmPlacementFromUnderUtilizedHost	Отримання плану розміщення ВМ з недовантажених ФС	List<Map<String, Object>>	List<? extends Vm> vmsToMigrate, Set<? extends Host> excludedHosts	ВМ для міграції, виключені з розгляду ФС
VmAllocationPolicyMigrationAgent	getVmsToMigrateFromHosts	Вибір ВМ для міграції з ФС	List<? extends Vm>	List<PowerHostUtilizationHistory> overUtilizedHosts	Перевантажені ФС
VmAllocationPolicyMigrationAgent	getVmsToMigrateFromUnderUtilizedHost	Вибір ВМ для міграції з недовантаженого ФС	List<? extends Vm>	PowerHost host	ФС
VmAllocationPolicyMigrationAgent	getOverUtilizedHosts	Отримати перевантажені ФС	List<PowerHostUtilizationHistory>	-	-

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
VmAllocationPolicyMigrationAgent	getSwitchedOffHosts	Отримати ФС в сплячому режимі	List<PowerHostUtilizationHistory>	-	-
VmAllocationPolicyMigrationAgent	isSwitchedOffHost	Перевірка чи ФС в сплячому режимі	boolean	PowerHostUtilizationHistory host	ФС
VmAllocationPolicyMigrationAgent	saveAllocation	Зберегти план розміщення	-	-	-
VmAllocationPolicyMigrationAgent	restoreAllocation	Відновити план розміщення	-	-	-
VmAllocationPolicyMigrationAgent	getPowerAfterAllocation	Отримати значення споживання електроенергії ФС після розміщення ВМ	double	PowerHost host, Vm vm	ФС, ВМ
VmAllocationPolicyMigrationAgent	getMaxUtilizationAfterAllocation	Отримати максимальне значення споживання електроенергії ФС після розміщення ВМ	double	PowerHost host, Vm vm	ФС, ВМ
VmAllocationPolicyMigrationAgent	getUtilizationOfCpuMips	Отримати використання процесора у MIPS	double	PowerHost host	ФС
HostPowerModeSelectionPolicyAgent	getHostPowerMode	Отримання режиму роботи ФС	int[]	-	-
HostPowerModeSelectionPolicyAgent	observeState	Отримання стану ЦОД	String	-	-
HostPowerModeSelectionPolicyAgent	saveState	Зберегти стан ЦОД	int	String state	Стан ЦОД

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
HostPowerModeSelection PolicyAgent	getSlaViolationTime	Отримання часу порушення вимог SLA	-	List<HostDynamicWorkload> hosts	Список ФС
HostPowerModeSelection PolicyAgent	getTotalPower	Отримання значення споживання електроенергії та кількості міграцій	-	List<Host> hosts	Список ФС
HostPowerModeSelection PolicyAgent	getPartPenalty	Обчислення штрафу	double	List<Double> list	Список показників
HostPowerModeSelection PolicyAgent	getPenalty	Обчислення загального штрафу	double	-	-
HostPowerModeSelection PolicyAgent	getNewQValue	Отримання Q-значення	double	double penalty, double estimateOptimalFutureValue	Штраф, попереднє Q-значення
HostPowerModeSelection PolicyAgent	setLearningRate	Встановлення швидкості (темпу) навчання	-	double learningRate	Швидкість (темп) навчання
HostPowerModeSelection PolicyAgent	setDiscountFactor	Встановлення коефіцієнта знецінювання	-	double discountFactor	Коефіцієнт знецінювання
HostPowerModeSelection PolicyAgent	setCofImportanceSla	Встановлення коефіцієнта, що визначає відносну важливість штрафу за порушення SLA	-	double cofImportanceSla	Коефіцієнт що визначає відносну важливість штрафу за порушення SLA

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
HostPowerModeSelectionPolicyAgent	setCofImportancePower	Встановлення коефіцієнта, що визначає відносну важливість штрафу за споживання електроенергії	-	double cofImportancePower	Коефіцієнт що визначає відносну важливість штрафу за споживання електроенергії
UtilizationModelBitbrains	getUtilization	Отримати значення споживання електроенергії за проміжок часу	double	double time	Проміжок часу
UtilizationModelPlanetLab	getUtilization	Отримати значення споживання електроенергії за проміжок часу	double	double time	Проміжок часу
DellPowerEdgeR640	getPowerData	Отримати значення споживання електроенергії	double	int index	Індекс
DellPowerEdgeR740	getPowerData	Отримати значення споживання електроенергії	double	int index	Індекс
DellPowerEdgeR830	getPowerData	Отримати значення споживання електроенергії	double	int index	Індекс

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
DellPowerEdgeR940	getPowerData	Отримати значення споживання електроенергії	double	int index	Індекс
VmAllocationPolicyNonPowerAware	optimizeAllocation	Оптимізація розміщення ВМ	List<Map<String, Object>>	List<? extends Vm> vmList	Список ВМ
PowerVmAllocationPolicyMigrationInterQuartileRange	isHostOverUtilized	Перевірка перевантаженості ФС	boolean	PowerHost host	ФС
PowerVmAllocationPolicyMigrationInterQuartileRange	isHostOverUtilizedBy	Перевірка перевантаженості ФС за ресурсом	boolean	PowerHost host, double[] utilizationHistory	ФС, історія використання ресурсу
PowerVmAllocationPolicyMigrationInterQuartileRange	getHostUtilizationIqr	Отримати значення використання ресурсу	double	double[] utilizationHistory	Історія використання ресурсу
PowerVmAllocationPolicyMigrationInterQuartileRange	setSafetyParameter	Встановлення параметру безпеки	-	double safetyParameter	Параметр безпеки
PowerVmAllocationPolicyMigrationLocalRegression	isHostOverUtilized	Перевірка перевантаженості ФС	boolean	PowerHost host	ФС
PowerVmAllocationPolicyMigrationLocalRegression	isHostOverUtilizedBy	Перевірка перевантаженості ФС за ресурсом	boolean	PowerHostUtilizationHistory host, double[] utilizationHistory	ФС, історія використання ресурсу
PowerVmAllocationPolicyMigrationLocalRegression	getParameterEstimates	Отримати оцінку використання ресурсу	double[]	double[] utilizationHistoryReversed	Історія використання ресурсу
PowerVmAllocationPolicyMigrationLocalRegression	getMaximumVmMigrationTime	Отримати максимальний час міграції	double	PowerHostUtilizationHistory host	ФС

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
PowerVmAllocationPolicy MigrationLocalRegression Robust	getParameterEstimates	Отримати оцінку використання ресурсу	double[]	double[] utilizationHistoryReversed	Історія використання ресурсу
PowerVmAllocationPolicy MigrationMedianAbsolute Deviation	isHostOverUtilized	Перевірка перевантаженості ФС	boolean	PowerHost host	ФС
PowerVmAllocationPolicy MigrationMedianAbsolute Deviation	isHostOverUtilizedBy	Перевірка перевантаженості ФС за ресурсом	boolean	PowerHost host, double[] utilizationHistory	ФС, історія використання ресурсу
PowerVmAllocationPolicy MigrationMedianAbsolute Deviation	getHostUtilizationMad	Отримати значення використання ресурсу	double	double[] utilizationHistory	Історія використання ресурсу
PowerVmAllocationPolicy MigrationMedianAbsolute Deviation	setSafetyParameter	Встановлення параметру безпеки	-	double safetyParameter	Параметр безпеки
PowerVmAllocationPolicy MigrationStaticThreshold	isHostOverUtilized	Перевірка перевантаженості ФС	boolean	PowerHost host	ФС
PowerVmAllocationPolicy MigrationStaticThreshold	isHostOverUtilizedCpu	Перевірка перевантаженості ФС за процесором	boolean	PowerHost host	ФС
PowerVmAllocationPolicy MigrationStaticThreshold	isHostOverUtilizedRam	Перевірка перевантаженості ФС за оперативною пам'яттю	boolean	PowerHost host	ФС
PowerVmAllocationPolicy MigrationStaticThreshold	isHostOverUtilizedBw	Перевірка перевантаженості ФС за мережею	boolean	PowerHost host	ФС
PowerVmSelectionPolicy MaximumCorrelation	getVmToMigrate	Отримати ВМ для міграції	Vm	final PowerHost host	ФС
PowerVmSelectionPolicy MaximumCorrelation	getUtilizationMatrix	Отримати матрицю використання процесора	double[][]	final List<PowerVm> vmList	Список ВМ

Продовження таблиці 4.2

Клас	Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
PowerVmSelectionPolicy MaximumCorrelation	getMinUtilizationHistorySize	Отримати мінімальний розмір історії використання процесора	int	final List<PowerVm> vmList	Список VM
PowerVmSelectionPolicy MaximumCorrelation	getCorrelationCoefficients	Отримати коефіцієнти кореляції	List<Double>	final double[][] data	Матриця
PowerVmSelectionPolicy MinimumMigrationTime	getVmToMigrate	Отримати VM для міграції	Vm	final PowerHost host	ФС
PowerVmSelectionPolicy MinimumUtilization	getVmToMigrate	Отримати VM для міграції	Vm	final PowerHost host	ФС
PowerVmSelectionPolicy RandomSelection	getVmToMigrate	Отримати VM для міграції	Vm	final PowerHost host	ФС

4.2.4 Керівництво користувача

Перед початком роботи з ПЗ необхідно розпакувати zip-архів Diss.zip в бажану директорію. Архів має наступний вміст (рисунок 4.3):

- директорія workload – директорія, в якій знаходяться директорії з файлами, що містять показники продуктивності VM (опис формату файлів з вхідними даними дивитися в пункті 4.1.1);
- файл Diss.jar – Java-архів, в якому міститься ПЗ;
- файл run.bat – пакетний файл, в якому міститься команда для запуску ПЗ;
- файл config.json – JSON файл з характеристики VM та ФС (опис ключів JSON файлу дивитися в пункті 4.1.1).

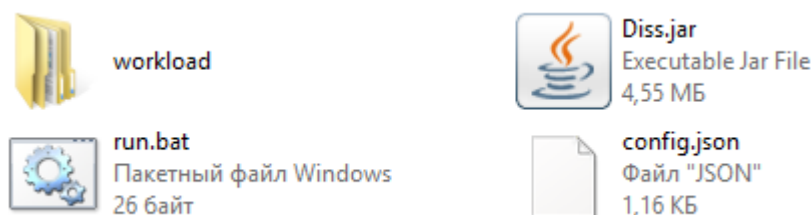


Рисунок 4.3 – Вміст архіву Diss.zip

У файлі `config.json` необхідно відредагувати шлях до директорії, в якій знаходяться директорії з файлами, що містять показники продуктивності ВМ (значення ключа `inputFolder`).

Для запуску ПЗ необхідно запуснути файл `run.bat` (файл `run.bat` та `Diss.jar` мають бути в одній директорії, повинна бути створена змінна середовища `JAVA_HOME`), після чого відкриється вікно командного рядка (рисунок 4.4) та головне вікно ПЗ (рисунок 4.5).



Рисунок 4.4 – Вікно командного рядка



Рисунок 4.5 – Головне вікно ПЗ

Для завантаження JSON файлу з характеристики VM та ФС потрібно натиснути кнопку «Вибрати файл» у головному вікні ПЗ (рисунок 4.6).

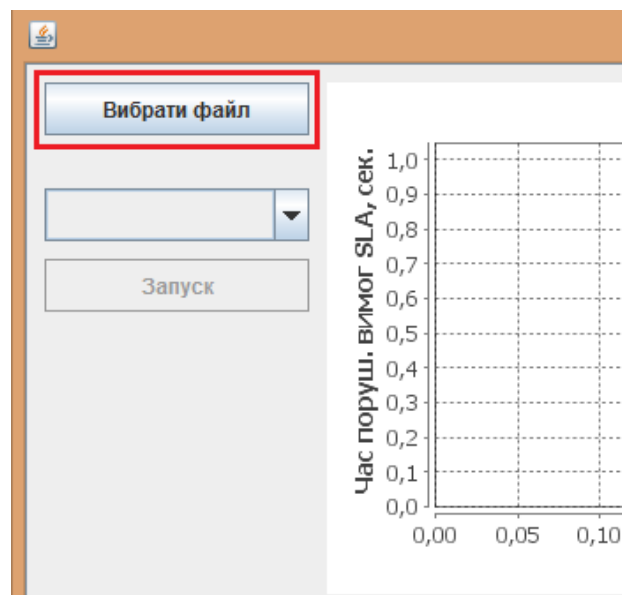


Рисунок 4.6 – Кнопка «Вибрати файл»

Після цього відкриється діалогове вікно для вибору JSON файлу (рисунок 4.7).

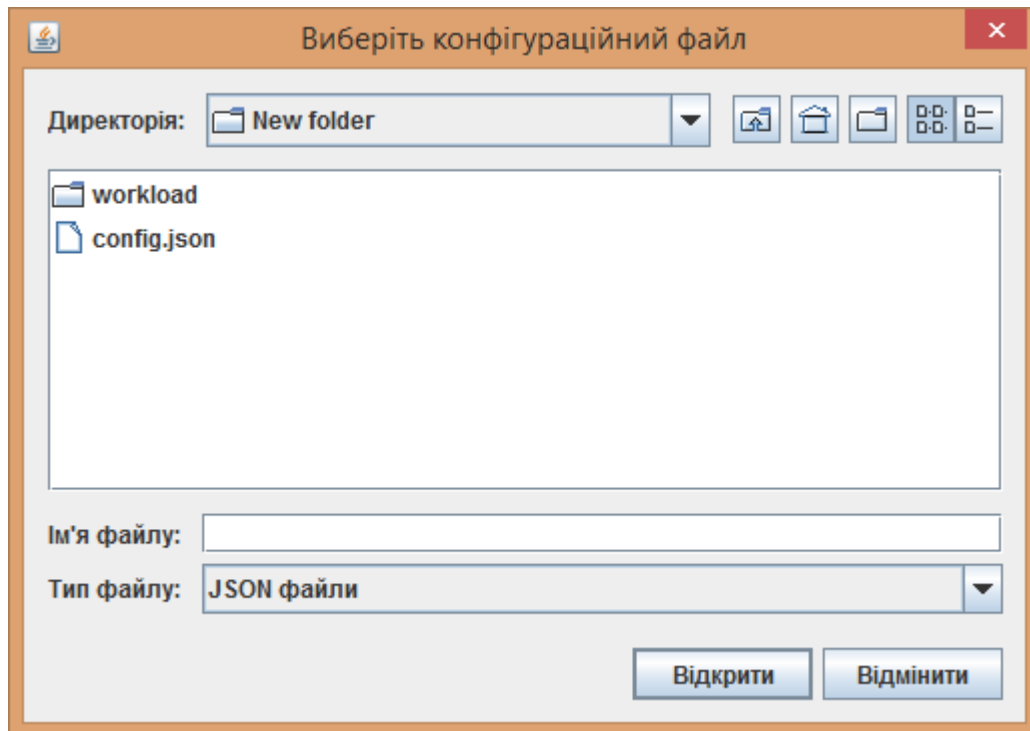


Рисунок 4.7 – Діалогове вікно для вибору JSON файлу

Потім у випадаючому списку потрібно обрати алгоритм для оптимізації розміщення VM (рисунок 4.8) та запустити моделювання роботи ЦОД, натиснувши на кнопку «Запуск» (рисунок 4.9).

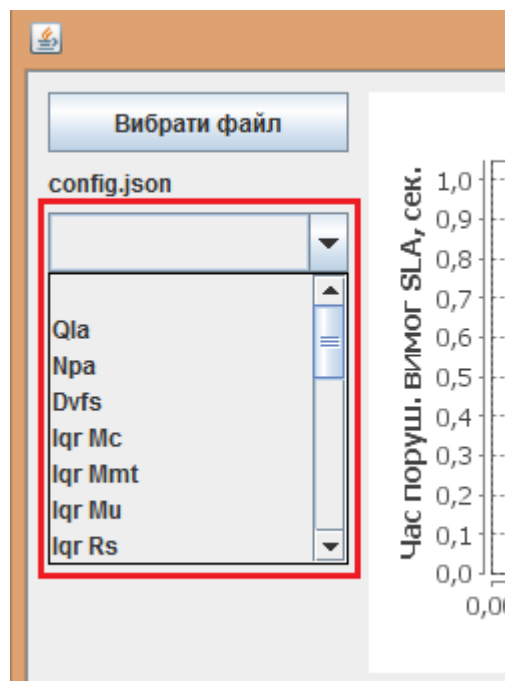


Рисунок 4.8 – Випадаючий список з алгоритмами оптимізації розміщення VM

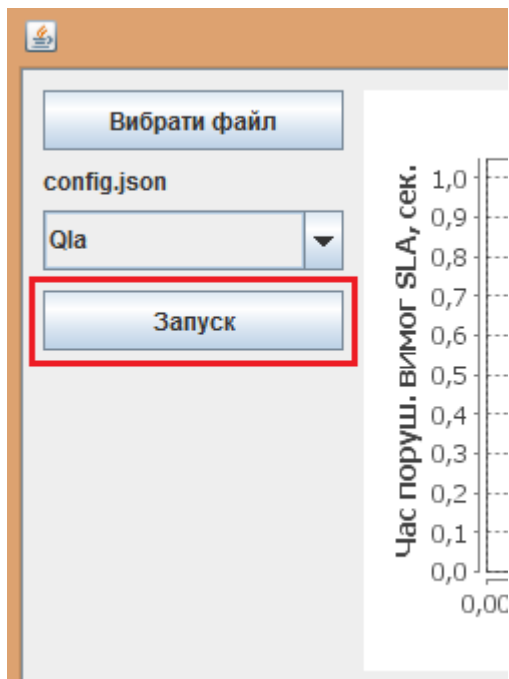


Рисунок 4.9 – Запуск моделювання роботи ЦОД

Після запуску моделювання у вікні командного рядка відобразатиметься час моделювання (рисунок 4.10).

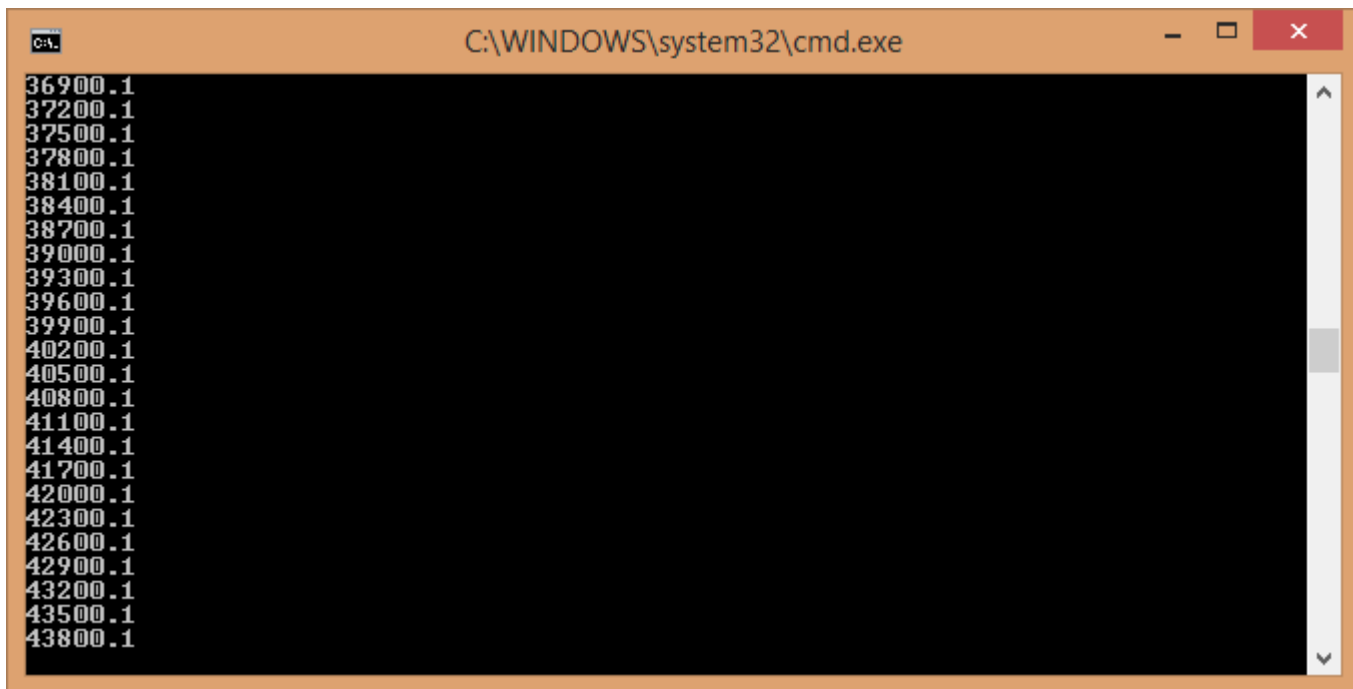


Рисунок 4.10 – Відображення часу моделювання

Після завершення моделювання у головному вікні ПЗ відображені показники часу порушення вимог SLA, споживання електроенергії, кількості міграцій VM у вигляді графіків (рисунок 4.11).

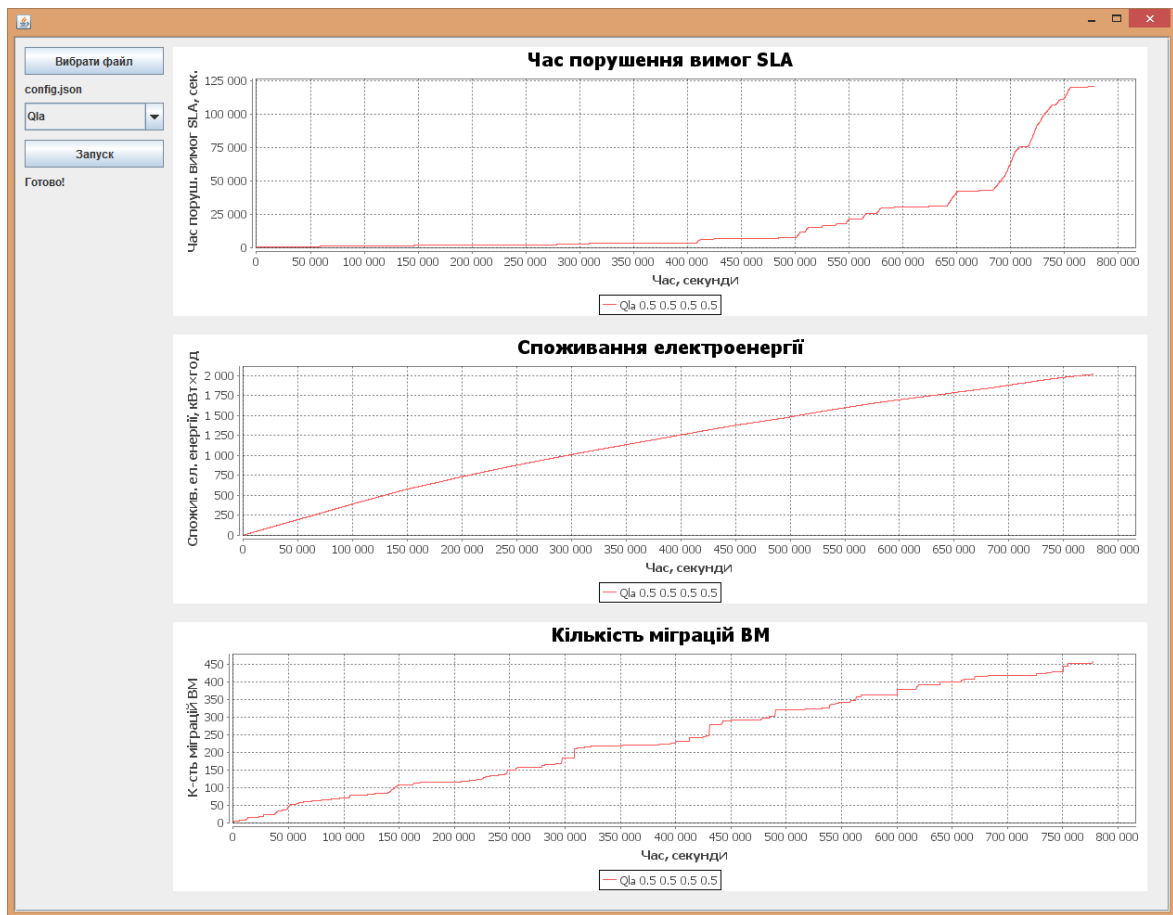


Рисунок 4.11 – Відображення результатів моделювання роботи ЦОД

У директорії output/log створено файл з розширенням .log, у якому містяться дані, які генеруються в процесі моделювання роботи ЦОД (рисунок 4.12).

```

1  Initialising...
2  Starting trace_rnd_8
3  Starting CloudSim version 4.0
4  Broker is starting...
5  Datacenter is starting...
6  Entities started.
7  0.0: Broker: Cloud Resource List received with 1 resource(s)
8  0.0: Broker: Trying to Create VM #0 in Datacenter
9  0.0: Broker: Trying to Create VM #1 in Datacenter
10 0.0: Broker: Trying to Create VM #2 in Datacenter
11 0.0: Broker: Trying to Create VM #3 in Datacenter
12 0.0: Broker: Trying to Create VM #4 in Datacenter
13 0.0: Broker: Trying to Create VM #5 in Datacenter
14 0.0: Broker: Trying to Create VM #6 in Datacenter
15 0.0: Broker: Trying to Create VM #7 in Datacenter
16 0.0: Broker: Trying to Create VM #8 in Datacenter
17 0.0: Broker: Trying to Create VM #9 in Datacenter

```

Рисунок 4.12 – Файл з даними, які генеруються в процесі моделювання роботи ЦОД

У директорії output/metrics створено файл з розширенням .csv, у якому містяться вихідні дані (опис формату файлів з вихідними даними дивитися в пункті 4.1.2).

У разі виникнення помилки під час роботи ПЗ відкриється вікно з повідомленням про помилку (рисунок 4.13).

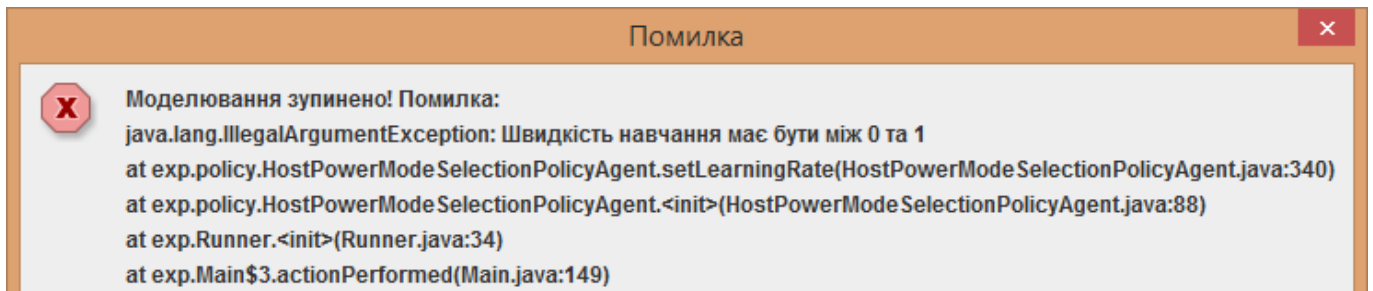


Рисунок 4.13 – Повідомлення про помилку

4.3 Висновок до розділу

У даному розділі наведений опис форматів вхідних і вихідних даних. Наведений опис ключів, які містяться у JSON файлі з характеристики VM та ФС, приклад вмісту JSON файлу.

Детально описані переваги та недоліки засобів розробки методу динамічного розміщення VM на основі НП. Фреймворк для моделювання хмарної інфраструктури та сервісів CloudSim 4.0 дозволяє здійснювати повноцінне моделювання і симуляцію хмарних обчислювальних систем, інфраструктур, сховищ даних, веб-сервісів, розподілу ресурсів між віртуальними машинами та ін.

Наведена схема структурна класів ПЗ, модифіковані класи фреймворка CloudSim 4.0, а також специфікацію функцій. Наведено детальне керівництво користувача.

5 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

5.1 Порядок проведення досліджень

Оскільки цільовою хмарною моделлю обслуговування є IaaS, важливо оцінити розроблений метод динамічного розміщення ВМ на основі НП у великомасштабній хмарній інфраструктурі. Але дуже важко провести велику кількість досліджень з реальною інфраструктурою, яка необхідна для оцінки розробленого методу. У зв'язку з цим, методом дослідження було обрано моделювання хмарної інфраструктури за допомогою фреймворка CloudSim 4.0.

Планується виконати моделювання ЦОД, до складу якого входять 200 гетерогенних ФС Dell Inc. PowerEdge R640, Dell Inc. PowerEdge R740, Dell Inc. PowerEdge R830 та Dell Inc. PowerEdge R940.

Споживання електроенергії фізичними серверами лінійно залежить від використання процесора [50]. Для досліджень будуть використані реальні дані про споживання електроенергії, надані The Standard Performance Evaluation Corporation (SPEC) [51]. У таблицях 5.1 та

Таблиця 5.2 наведено споживання електроенергії у ватах обраними ФС при різних показниках використання процесора та технічні характеристики ФС відповідно.

Таблиця 5.1 – Споживання електроенергії у ватах при різних показниках використання процесора

ФС	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Dell Inc. PowerEdge R640	55	125	151	176	202	232	261	301	357	421	469
Dell Inc. PowerEdge R740	52.3	129	147	170	195	224	255	292	344	408	457
Dell Inc. PowerEdge R830	86.4	181	215	253	287	315	340	377	421	483	562
Dell Inc. PowerEdge R940	106	245	292	336	383	437	502	583	694	820	915

Таблиця 5.2 – Технічні характеристики ФС

ФС	Кількість	Продуктивність ядер, MIPS	Кількість ядер	Оперативна пам'ять, МБ	Пропускна здатність мережі, Кбіт/с	Об'єм жорсткого диску, МБ
Dell Inc. PowerEdge R640	50	2500	56	196608	10000000	8000000
Dell Inc. PowerEdge R740	50	2500	56	196608	10000000	8000000
Dell Inc. PowerEdge R830	50	2200	88	262144	10000000	8000000
Dell Inc. PowerEdge R940	50	2500	112	393216	10000000	8000000

Для моделювання хмарної інфраструктури важливо проводити експерименти з використанням показників продуктивності VM в реальних системах. Тому для досліджень будуть використані реальні дані продуктивності 500 VM зібраних з інтервалами в 5 хвилин з розподіленого ЦОД Vitbrains [44] протягом одного місяця. У таблиці 5.3 наведені технічні характеристики VM.

Таблиця 5.3 – Технічні характеристики VM

VM	Кількість	Продуктивність ядер, MIPS	Кількість ядер	Оперативна пам'ять, МБ	Пропускна здатність мережі, Кбіт/с	Розмір образу, МБ
Тип 1	125	500	2	2048	100000	20000
Тип 2	125	1300	4	4096	100000	20000
Тип 3	125	2200	8	8192	100000	20000
Тип 4	125	2200	16	16384	100000	20000

Оцінка ефективності методу розміщення VM на основі НП проводитиметься за трьома показниками: час порушення вимог SLA, споживання електроенергії та кількість міграцій. На першому етапі потрібно дослідити вплив швидкості навчання α та коефіцієнта знецінювання γ на ефективність роботи методу. Після вибору цих коефіцієнтів, потрібно дослідити вплив коефіцієнтів, що визначають відносну

важливість штрафу за порушення SLA β та штрафу за споживання електроенергії δ на ефективність роботи методу. Останнім етапом буде порівняння розробленого методу з адаптивним евристичним методом консолідації VM в ЦОД для всіх можливих комбінацій політик виявлення перевантажених ФС та політик виявлення VM для міграції. У зв'язку з тривалим процесом моделювання, час моделювання складатиме 9 діб.

5.2 Дослідження впливу коефіцієнтів на ефективність роботи розробленого методу

Проведемо моделювання ЦОД змінюючи коефіцієнти наступним чином: α збільшуватимемо від 0.1 до 1 з кроком 0.25; γ збільшуватимемо від 0 до 1 з кроком 0.25; β та δ залишимо рівними 0.5. В результаті отримаємо 25 комбінацій коефіцієнтів. На рисунку 5.1 наведено результати моделювання ЦОД з врахуванням методу розміщення VM на основі НП.

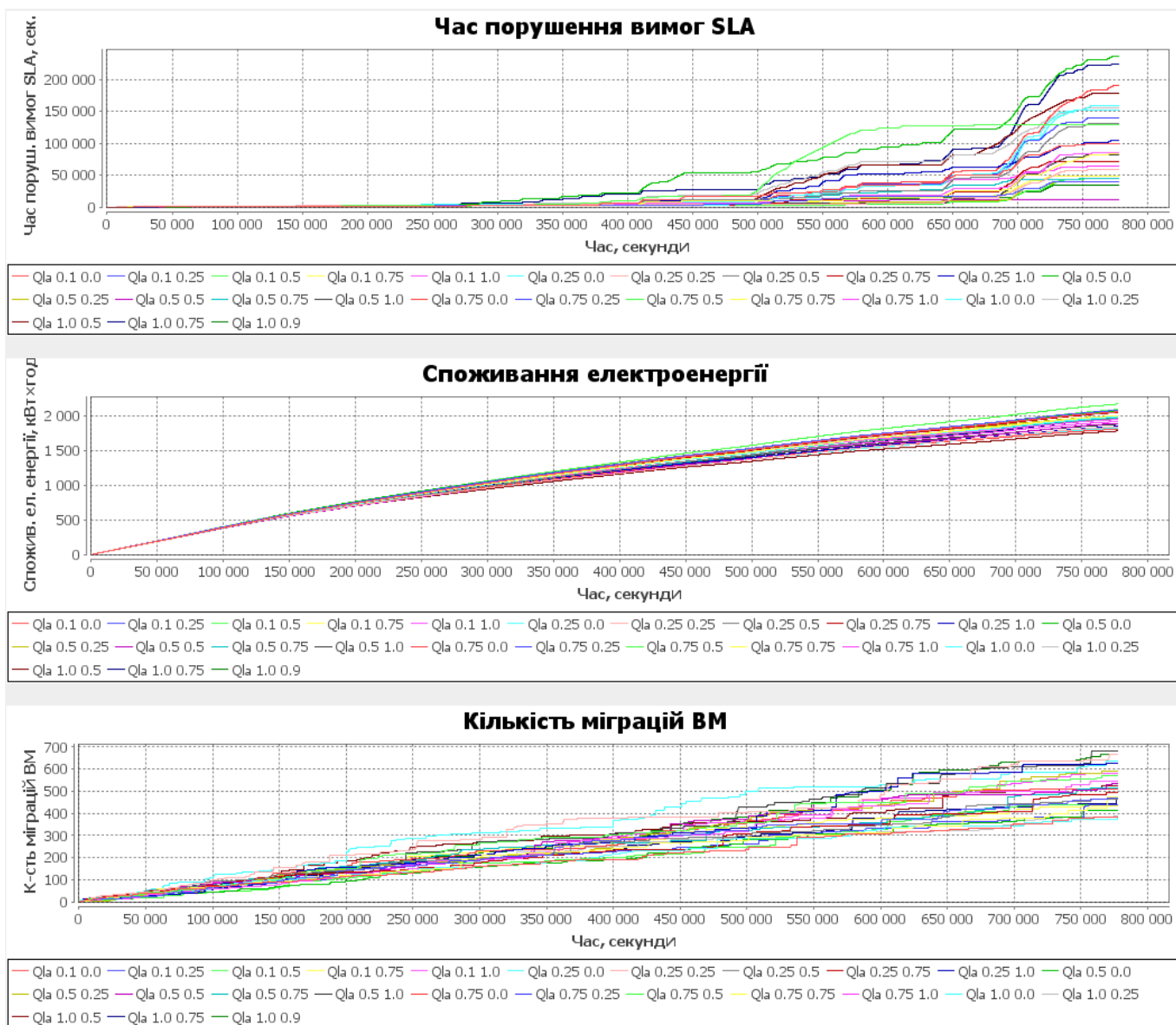


Рисунок 5.1 – Результати моделювання ЦОД при різних значеннях α та γ

У таблиці 5.4 наведені значення показників ефективності в кінцевий момент часу моделювання.

Таблиця 5.4 – Значення показників ефективності при різних значеннях α та γ

Значення коефіцієнтів α та γ ($\beta = 0.5$, $\delta = 0.5$)	Час порушення вимог SLA, сек. ($\times 10^3$)	Споживання електроенергії, кВт×год	Кількість міграцій VM
0.1;0	98.857	2062.546	382
0.1;0.25	39.986	2085.769	467
0.1;0.5	129.013	2095.616	572
0.1;0.75	83.083	2007.882	472
0.1;1	84.671	1952.283	544
0.25;0	152.544	1992.373	378
0.25;0.25	59.028	1846.735	665

Продовження таблиці 5.4

Значення коефіцієнтів α та γ ($\beta = 0.5, \delta = 0.5$)	Час порушення вимог SLA, сек. ($\times 10^3$)	Споживання електроенергії, кВт \times год	Кількість міграцій ВМ
0.25;0.5	131.276	1962.930	464
0.25;0.75	71.737	2054.806	495
0.25;1	104.085	1852.390	628
0.5;0	235.813	2071.020	411
0.5;0.25	39.515	2007.864	592
0.5;0.5	12.086	1897.250	537
0.5;0.75	44.314	1974.655	515
0.5;1	81.986	1883.206	680
0.75;0	190.063	1812.047	512
0.75;0.25	139.555	2061.216	441
0.75;0.5	44.462	2174.933	440
0.75;0.75	48.883	1991.326	435
0.75;1	64.630	1921.843	579
1;0	159.539	1829.403	637
1;0.25	155.708	2054.266	387
1;0.5	178.323	1785.983	524
1;0.75	223.704	1962.960	472
1;0.9	34.959	1956.481	669

Зеленим кольором позначено мінімальні значення показників, червоним кольором – максимальні. Беручи до уваги пріоритетність показника часу порушення вимог SLA, для подальшого дослідження виберемо $\alpha = 0.5$ та $\gamma = 0.5$, які забезпечують мінімальне значення порушення вимог SLA рівне 12086 секундам.

Проведемо моделювання ЦОД змінюючи коефіцієнти наступним чином: β та δ збільшуватимемо від 0 до 1 з кроком 0.1, в сумі β та δ повинні бути рівними 1; відповідно результатам минулого кроку, α та γ залишимо рівними 0.5. В результаті отримаємо 11 комбінацій коефіцієнтів. На рисунку 5.2 наведено результати моделювання ЦОД з врахуванням методу розміщення ВМ на основі НП.

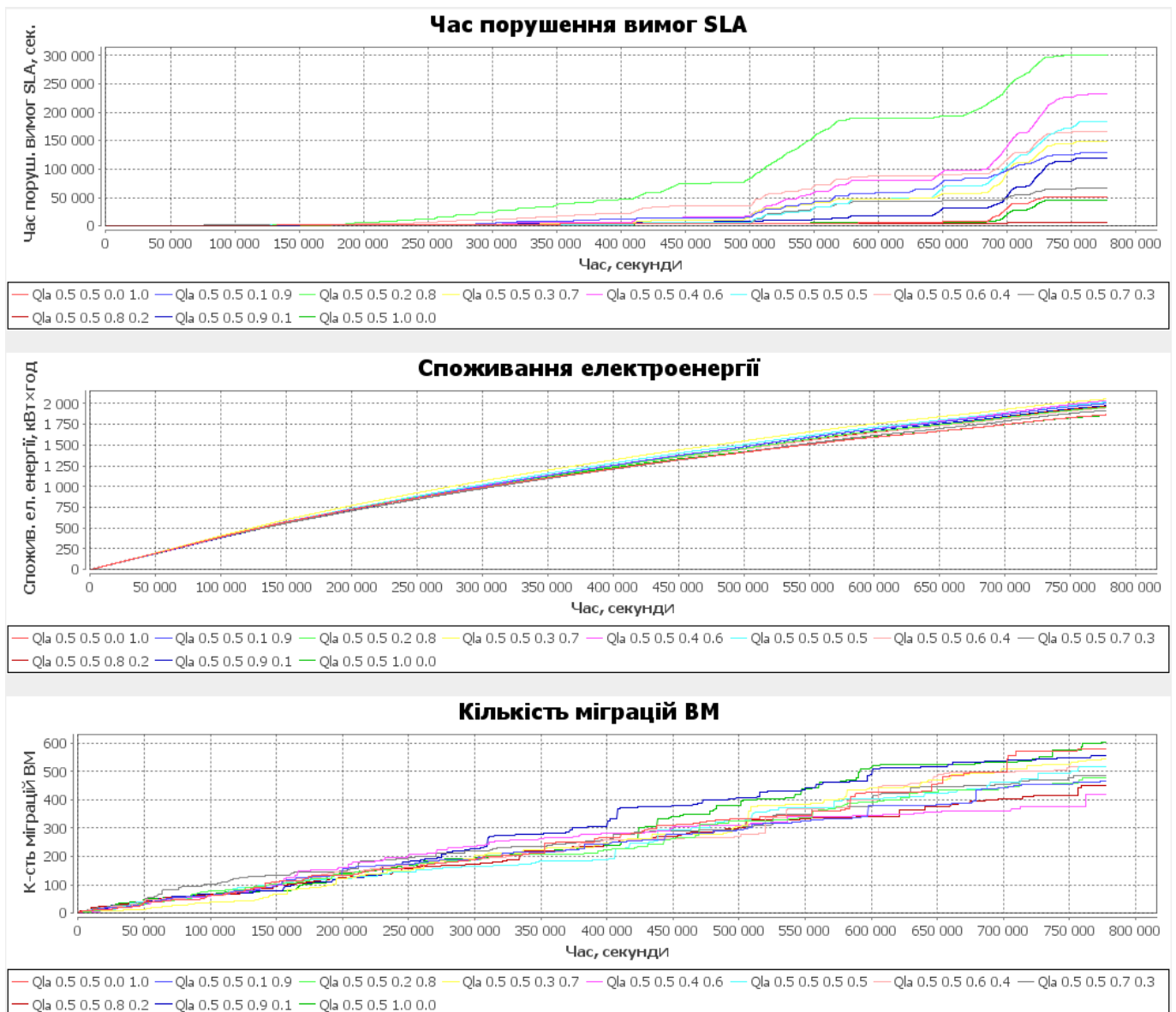


Рисунок 5.2 – Результати моделювання ЦОД при різних значеннях β та δ

У таблиці 5.5 наведені значення показників ефективності в кінцевий момент часу моделювання.

Таблиця 5.5 – Значення показників ефективності при різних значеннях β та δ

Значення коефіцієнтів β та δ ($\alpha = 0.5$, $\gamma = 0.5$)	Час порушення вимог SLA, сек. ($\times 10^3$)	Споживання електроенергії, кВт \times год	Кількість міграцій VM
0;1	51.339	1864.459	581
0.1;0.9	128.063	2000.401	467
0.2;0.8	299.932	1952.644	479
0.3;0.7	148.336	2053.336	543
0.4;0.6	231.593	2033.889	419
0.5;0.5	183.465	2014.854	516

Продовження таблиці 5.5

Значення коефіцієнтів β та δ ($\alpha = 0.5, \gamma = 0.5$)	Час порушення вимог SLA, сек. ($\times 10^3$)	Споживання електроенергії, кВт \times год	Кількість міграцій ВМ
0.6;0.4	166.761	1945.795	517
0.7;0.3	65.719	1916.472	487
0.8;0.2	6.675	1962.777	452
0.9;0.1	119.571	1972.313	558
1;0	45.266	1856.26	602

Зеленим кольором позначено мінімальні значення показників, червоним кольором – максимальні. Для подальшого дослідження виберемо дві комбінації коефіцієнтів: $\beta = 0.8$ та $\delta = 0.2$, які забезпечують мінімальне значення часу порушення вимог SLA рівне 6675 секундам; $\beta = 1$ та $\delta = 0$, які забезпечують мінімальне значення споживання електроенергії рівне 1856.26 кВт \times год.

5.3 Порівняння розробленого методу з адаптивним евристичним методом консолідації віртуальних машин

Порівняємо результати моделювання ЦОД з врахуванням методу динамічного розміщення ВМ на основі НП для двох комбінацій коефіцієнтів отриманих в підрозділі 5.2 та результати моделювання ЦОД з врахуванням адаптивного евристичного методу консолідації ВМ для всіх можливих комбінацій політик виявлення перевантажених ФС та політик виявлення ВМ для міграції. На рисунку 5.3 наведено результати моделювання ЦОД з врахування обох методів.

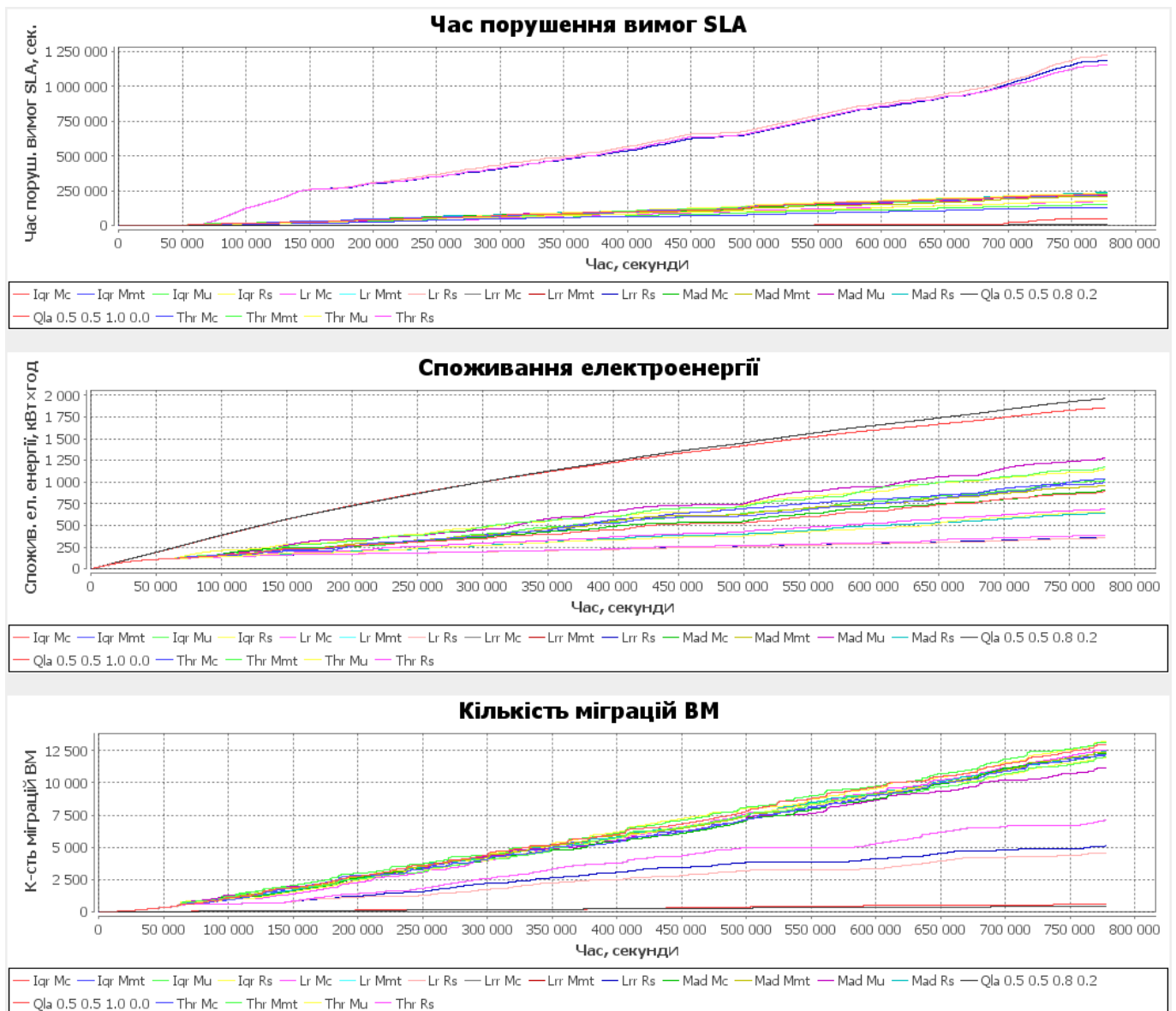


Рисунок 5.3 – Порівняння методу динамічного розміщення VM на основі НП та адаптивного евристичного методу консолідації VM

Thr, Iqr, Mad, Lr, Lrr – політики виявлення перевантажених ФС описані в підпункті 2.1.3.1; Mmt, Rs, Mc, Mu – політики виявлення VM для міграції описані в підпункті 2.1.3.2.

За показниками часу порушення вимог SLA та кількості міграцій метод динамічного розміщення VM на основі НП є ефективнішим ніж всі політики виявлення перевантажених ФС та політик виявлення VM для міграції адаптивного евристичного методу консолідації VM. За показником споживання електроенергії адаптивний евристичний метод є ефективнішим за рахунок великої кількості міграцій

ВМ (враховуючи обмеження до 5 одночасних міграцій з ФС та на ФС [52]), що є неприпустимим в реальній хмарній інфраструктурі.

5.4 Висновки до розділу

У даному розділі було наведено порядок проведення досліджень, визначені вхідні дані та технічні характеристики ФС та ВМ. Було виконано дослідження впливу швидкості навчання α , коефіцієнта знецінювання γ та коефіцієнтів, що визначають відносну важливість штрафу за порушення SLA β і штрафу за споживання електроенергії δ на ефективність роботи методу динамічного розміщення ВМ на основі НП. Вибрано дві комбінації коефіцієнтів, які забезпечують мінімальне значення часу порушення вимог SLA та мінімальне значення споживання електроенергії відповідно.

Визначено, що за показниками часу порушення вимог SLA та кількості міграцій метод динамічного розміщення ВМ на основі НП є ефективнішим ніж всі політики виявлення перевантажених ФС та політик виявлення ВМ для міграції адаптивного евристичного методу консолідації ВМ, але гірший за показником споживання електроенергії.

ЗАГАЛЬНІ ВИСНОВКИ

При виконанні магістерської дисертації було проаналізовано предметне середовище управління віртуалізованими ресурсами центру обробки даних та розглянуті рішення з їх управління. Проведено огляд методів управління обчислювальними ресурсами, а саме адаптивний евристичний метод консолідації віртуальних машин, метод управління розподілом ресурсів центру обробки даних при віртуальному хостингу та метод автоконфігурування віртуальних машин на основі навчання з підкріпленням.

Була наведена постановка задачі зменшення споживання електроенергії та часу порушення вимог SLA, наведена математична модель центру обробки даних. Розроблено метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням, який при виборі управляючих впливів враховує витрати електроенергії та час порушення вимог SLA. Розроблена модель агента, який враховує зміни робочого навантаження на ресурси для прийняття рішення щодо включення або переключення в сплячий режим незавантажених фізичних серверів з метою зменшення витрат електроенергії. Запропонований агент навчання з підкріпленням базується на методі Q -навчання. Перевагою методу динамічного розміщення віртуальних машин є здатність виконувати в режимі онлайн розміщення нових віртуальних машин одночасно з перерозподілом вже працюючих віртуальних машин.

Розроблена модель центру обробки даних за допомогою фреймворка для моделювання хмарної інфраструктури та сервісів CloudSim 4.0 [41]. Виконано програмну реалізацію методу динамічного розміщення віртуальних машин на основі навчання з підкріпленням. Описані формати вхідних та вихідних даних. Наведена схема структурна класів програмного забезпечення, модифіковані класи фреймворка, а також специфікацію функцій. Наведено детальне керівництво користувача.

У ході досліджень було виявлено дві комбінації коефіцієнтів, які забезпечують мінімальне значення часу порушення вимог SLA та мінімальне значення споживання електроенергії відповідно. Результати моделювання центру обробки даних з

врахуванням методу динамічного розміщення віртуальних машин на основі навчання з підкріпленням для двох комбінацій коефіцієнтів порівнювалися з результатами моделювання центру обробки даних з врахуванням адаптивного евристичного методу консолідації віртуальних машин для всіх можливих комбінацій політик виявлення перевантажених фізичних серверів та політик виявлення віртуальних машин для міграції. Виявлено, що розроблений метод дозволяє зменшити час порушення вимог SLA та кількість міграцій віртуальних машин.

За матеріалами дисертації було опубліковано п'ять наукових робіт: стаття в науковому журналі, публікація в базі даних IEEE Xplore Digital Library [53] та три тези доповідей на наукових конференціях [3-7].

ПЕРЕЛІК ПОСИЛАНЬ

1. R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction." MIT Press, p.360, 1998.
2. C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning." Machine Learning, №3(8), pp. 279-292, 1992.
3. Коваль А.А., Жаріков Е.В. Порівняльний аналіз існуючих методів управління ресурсами в умовах хмарних обчислень / А.А. Коваль, Е.В. Жаріков / Матеріали 10-ї Всеукраїнської науково-практичної Web конференції аспірантів, студентів та молодих вчених «Комп'ютерні інтелектуальні системи та мережі». – м. Кривий-Ріг.: ДВНЗ «Криворізький національний університет», 22-24 березня 2017 р. – С. 8-10.
4. Коваль А.А., Терентьев Р.А. Comparative analysis of modeling methods of infrastructure of cloud computing / А.А. Коваль, Р.А. Терентьев / Матеріали 18-ї Всеукраїнської студентської науково-практичної конференції «Наука та техніка ХХІ століття». – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 7 грудня 2017 р. – С. 124-125.
5. Жаріков Е.В., Коваль А.А. Метод розміщення віртуальних машин на основі навчання з підкріпленням / Е.В. Жаріков, А.А. Коваль / Матеріали науково-практичної конференції «Інформатика та обчислювальна техніка-ІОТ-2018». – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 23-24 квітня 2018 р.
6. Жаріков Е.В. Динамічне розміщення віртуальних машин на основі навчання з підкріпленням в хмарних центрах обробки даних / Е.В. Жаріков, А.А. Коваль, Р.А. Терентьев. // Наукові вісті Далівського університету. - 2017. - № 13.
7. O. Rolik, E. Zharikov, A. Koval, S. Telenyk, "Dynamic management of data center resources using reinforcement learning." in Proceedings of the 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), 2018.
8. J. Ll. Berral, I. Goiril, R. Nou, F. Julia, J. Guitart, R. Gavaldà and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in Proceedings of

- the 1st International Conference on Energy-Efficient Computing and Networking, 2010, pp. 215-224.
9. G. Dhiman and T. S. Rosing, "System-level power management using online learning," in Proceedings of the Computer-Aided Design of Integrated Circuits and Systems (CADICS), 2009, pp. 676-689.
 10. J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin. "Vconf: a reinforcement learning approach to virtual machine auto-configuration," in Proceedings of the 6th International Conference on Autonomic Computing (ICAC), 2009, pp. 137-146.
 11. G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in Proceedings of the the IEEE International Conference on Autonomic Computing (ICAC), 2006, pp. 65-73.
 12. Y. Tan, W. Liu, and Q. Qiu, "Adaptive power management using reinforcement learning," in Proceedings of the International Conference on Computer-Aided Design (ICCAD '09), 2009, pp 46-467.
 13. E. Ipek, O. Mutlu, J. F. Martinez, and R. Caruana. "Self-optimizing memory controllers: A reinforcement learning approach," in Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA), 2008, pp. 39-50.
 14. A. Beloglazov, R Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826-831.
 15. X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck, "From data center resource allocation to control theory and back." Cloud Computing, pp. 410-417, 2010.
 16. X. Cheng-Zhong, R. Jia, B. Xiangping, "URL: A Unified Reinforcement Learning Approach for Autonomic Cloud Management," Department of Electrical & Computer Engineering Wayne State University, pp. 1-15.
 17. F. Farahnakian, P. Liljeberg, J. Plosila, "Energy-Efficient Virtual Machines Consolidation in Cloud Data Centers Using Reinforcement Learning," in Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2014, pp. 500-507.

18. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, 2005, pp. 273-286.
19. T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration." NSDI, vol. 7, pp. 17, 2007.
20. N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on, 2007, pp. 119-128.
21. X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner, and others, "1000 islands: Integrated capacity and workload management for the next generation data center," in Autonomic Computing, 2008. ICAC'08. International Conference on, 2008, pp. 172-181.
22. Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in Proceedings of the 2nd ACM Symposium on Cloud Computing, 2011, p. 5.
23. A. Beloglazov, R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers." Concurrency and Computation: Practice and Experience, vol. 24, no. 13, pp. 1397-1420, 2012.
24. M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh, and Q. Althebyan, "Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure." Cluster Computing, vol. 18, no. 2, pp. 919-932, 2015.
25. N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in Proceedings of the 10th IFIP/IEEE Intl. Symp. on Integrated Network Management (IM), 2007, pp.119-128.
26. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines." Journal of Computer Networks, vol. 53, pp. 2923-2938, 2009.

27. Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in Proceedings of the International Conference for the Computer Measurement Group (CMG), 2007, pp. 399-407.
28. M. Wang, X. Meng, L. Zhang, "Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data centers," in Proceedings of IEEE INFOCOM 2011 MINI-CONFERENCE, 2011, pp. 71-75.
29. R. Nathuji and K. Schwan, "Virtual Power: Coordinated Power Management in Virtualized Enterprise Systems," in Proceedings of the 22st ACM Symposium on Operating Systems Principles (SOSP'07), 2007, pp. 265-278.
30. M. Y. Lim, F. Rawson, T. K. Bletsch, V. W. Freeh. "PADD: Power-Aware Domain Distribution," in Proceedings of the 29th International Conference on Distributed Computing Systems (ICDCS), 2009, pp. 239-147.
31. Теленик С.Ф. Управління ресурсами центрів оброблення даних [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов, К. Крижова //Вісник Львів. УН-ТУ. – Серія прикл. матем. інформ., 2009. – Вип. 15. – С.325-340.
32. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов.– В надзаг.: Інформаційно-управляючі комплекси і системи., 2010. – С.106-120.
33. Теленик С.Ф. Технологія управління ІТ-інфраструктурою на основі ресурсного підходу [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов // Вісник ЖДТУ. – № 4 (47) .– В надзаг.: Технічні науки., 2008. – С.180-189.
34. Ролик А. И. Модель управления перераспределением ресурсов информационно-телекоммуникационной системы при изменении значимости бизнес-процессов / А. И. Ролик // Автоматика. Автоматизация. Електротехнічні комплекси та системи., 2007. № 2. – С. 73-82.
35. Скатков А.В. Информационная модель управления ИТ-ресурсами критических систем [Текст]: збірник наукових праць СНУЯЕтаП /А.В. Скатков, К.П. Аникевич, В.И. Шевченко.– В надзаг.: Інформаційні системи і технології., 2011. – С.201-206.

36. Ткачук М. В. Деякі проблеми управління IT-інфраструктурою підприємств: сучасний стан та перспективи розвитку / М. В. Ткачук, В. Є. Сокол // Східно-Європейський журнал передових технологій. – 2010. – № 6/2 (48). – С. 68-72.
37. Стіренко С.Г., Тимошин Ю.А. Ефективне застосування технології віртуалізації для підвищення роботи IT інфраструктури. – К., В збірнику наук. праць "ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ ТА УПРАВЛІННЯ", вип.4(28), НАУ, 2009. – С.125-130.
38. Матвеев І. Н. Виртуализация вычислений и экономические показатели корпоративной IT-инфраструктуры / И.Н. Матвеев, А.И. Кулиш / Вісник Бердянського університету менеджменту і бізнесу. – 2011. – №1(13). – С. 60-63.
39. S. Telenyk, E. Zharikov, O. Rolik, "Architecture and Conceptual Bases of Cloud IT Infrastructure Management." *Advances in Intelligent Systems and Computing*, vol. 512, pp. 41-62, 2017.
40. E. Zharikov, O. Rolik, S. Telenyk, "An integrated approach to cloud data center resource management." in *Proceedings of the 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, 2017, pp. 211-218.
41. CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services [Електронний ресурс] // Режим доступу: <http://www.cloudbus.org/cloudsim/>.
42. IntelliJ IDEA [Електронний ресурс] // Режим доступу: <https://www.jetbrains.com/idea/>.
43. JDK 8 and JRE 8 Installation Start Here [Електронний ресурс] // Режим доступу: https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html.
44. GWA-T-12 Bitbrains [Електронний ресурс] // Режим доступу: <http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains>.
45. planetlab-workload-traces [Електронний ресурс] // Режим доступу: <https://github.com/beloglazov/planetlab-workload-traces>.
46. Standard Performance Evaluation Corporation. Dell Inc. PowerEdge R640 [Електронний ресурс] // Режим доступу:

http://spec.org/power_ssj2008/results/res2017q3/power_ssj2008-20170829-00781.html.

47. Standard Performance Evaluation Corporation. Dell Inc. PowerEdge R740 [Электронный ресурс] // Режим доступа: http://spec.org/power_ssj2008/results/res2017q3/power_ssj2008-20170829-00780.html.
48. Standard Performance Evaluation Corporation. Dell Inc. PowerEdge R830 [Электронный ресурс] // Режим доступа: http://spec.org/power_ssj2008/results/res2016q3/power_ssj2008-20160705-00737.html.
49. Standard Performance Evaluation Corporation. Dell Inc. PowerEdge R940 [Электронный ресурс] // Режим доступа: http://spec.org/power_ssj2008/results/res2017q4/power_ssj2008-20171010-00789.html.
50. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing* 2009; 12(1):1–15.
51. Standard Performance Evaluation Corporation [Электронный ресурс] // Режим доступа: <http://spec.org/>.
52. Limits on Simultaneous Migrations [Электронный ресурс] // Режим доступа: <https://docs.vmware.com/en/VMware-vSphere/6.0/com.vmware.vsphere.vcenterhost.doc/GUID-25EA5833-03B5-4EDD-A167-87578B8009B3.html>.
53. IEEE Xplore Digital Library. Dynamic management of data center resources using reinforcement learning [Электронный ресурс] // Режим доступа: <https://ieeexplore.ieee.org/abstract/document/8336194/>.

ДОДАТОК А Графічний матеріал

ПЛАКАТ 1 Блок-схема алгоритму динамічного розміщення віртуальних машин на основі навчання з підкріпленням

ПЛАКАТ 2 Блок-схема алгоритму розподілу віртуальних машин

ПЛАКАТ 3 Математична модель

ПЛАКАТ 4 Структура вхідних і вихідних даних

ПЛАКАТ 5 Схема структурна класів програмного забезпечення

ПЛАКАТ 6 Схема структурна послідовності

ПЛАКАТ 7 Копії екранних форм

ПЛАКАТ 8 Результати досліджень