УДК: 004.415.3

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПОДІЛУ ВІРТУАЛЬНИХ РЕСУРСІВ В ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Боїло В.М.[1)], **Онищук А.З.**[2)], **Мартинюк В.В.**[3)]
Західноукраїнський національний університет
[1)] магістрант; [2)] аспірант; [3)] магістрант

I. Постановка проблеми

Обчислювальні системи нині є ключовим елементом в інформаційному суспільстві, вимагаючи ефективного управління та розподілу віртуальних ресурсів [1, 2]. Проблема визначення оптимальних методів розподілу ресурсів залишається актуальною у зв'язку зі зростанням потреб в обчислювальних можливостях. [3, 4]

II. Мета роботи

Метою дослідження є розробка математичних моделей та програмного забезпечення для ефективного розподілу віртуальних ресурсів у сучасних обчислювальних системах.

III. Обґрунтування отриманих результатів

Існує безліч різних систем для розподілу віртуальних ресурсів, які використовуються у сферах хмарних обчислень та віртуалізації. Ось кілька прикладів систем, кожна з яких вирішує питання розподілу віртуальних ресурсів унікальним чином:

VMware vSphere / ESXi - це інтегрована система віртуалізації, яка включає гіпервізор ESXi та менеджер ресурсів vCenter. ESXi забезпечує віртуалізацію на рівні сервера, а vCenter дозволяє централізовано керувати та розподіляти віртуальні машини.

OpenStack - це відкрите програмне забезпечення для будування та керування хмарними інфраструктурами. Він включає модуль Nova, який дозволяє автоматизовано створювати та керувати віртуальними машинами, а також ресурсний менеджер, який відповідає за розподіл ресурсів.

Amazon EC2 (Elastic Compute Cloud) - це частина Amazon Web Services (AWS) і надає можливість орендувати віртуальні сервери у хмарному середовищі. EC2 використовує концепцію інстанцій, де користувач може динамічно змінювати розмір та тип віртуальних машин залежно від потреб.

Microsoft Hyper-V - це гіпервізор для віртуалізації операційних систем у середовищі Windows. Hyper-V дозволяє керувати та розподіляти віртуальні машини через Hyper-V Manager або System Center Virtual Machine Manager (SCVMM).

Kubernetes - це система контейнеризації та оркестрації, яка надає можливість ефективно розгортати та керувати контейнерами. Керування ресурсами та розподіл виконуваних завдань є важливою частиною функціональності Kubernetes.

Ці системи демонструють різноманітні підходи до розподілу віртуальних ресурсів, враховуючи різні потреби та вимоги користувачів у хмарних обчисленнях і віртуалізації. [5]

У межах роботи отримано наступні результати:

1. *Математичні моделі розподілу ресурсів*:

Задача розподілу віртуальних ресурсів полягає в оптимальному розміщенні віртуальних машин на фізичних серверах так, щоб забезпечити ефективне використання ресурсів і максимальну продуктивність обчислювальної системи. Математична модель для цієї задачі включає такі аспекти:

- Обмеження ресурсів: Врахування обмежень фізичних серверів, таких як обсяг CPU, обсяг оперативної пам'яті та дисковий простір.
- Вимоги віртуальних машин: Врахування вимог кожної віртуальної машини до ресурсів, таких як кількість CPU, об'єм пам'яті та обсяг дискового простору.
- Цільова функція: Визначення цільової функції, яка максимізує загальну продуктивність системи або мінімізує сумарне використання ресурсів, забезпечуючи баланс між серверами.
- Адаптивність: Розгляд можливості динамічного адаптивного розподілу ресурсів в залежності від змінних обсягів робочих завдань та вимог користувачів.

Оптимізація рішення: Врахування алгоритмів оптимізації, таких як генетичні алгоритми або лінійне програмування, для знаходження оптимального розподілу віртуальних машин на серверах.

2. *Програмне забезпечення*: Розроблено програмне забезпечення на основі отриманих математичних моделей, що дозволяє автоматизовано та оптимально розподіляти віртуальні ресурси.

Проект реалізований мовою програмування Python з використанням бібліотеки DEAP (Distributed Evolutionary Algorithms in Python) для роботи з генетичними алгоритмами.

Структура проекту:

1. Модуль virtual_machine.py:

• Клас VirtualMachine, що представляє віртуальну машину з властивостями, такими як ідентифікатор, вимоги до ресурсів тощо.

2. Модуль physical_server.py:

• Клас PhysicalServer, який моделює фізичний сервер з властивостями, які визначають його обмеження щодо ресурсів.

3. Модуль genetic_algorithm.py:

• Клас GeneticAlgorithm, що використовує бібліотеку DEAP для реалізації генетичного алгоритму для оптимізації розподілу віртуальних машин на фізичних серверах.

4. Модуль main.py:

• Основний файл, який створює екземпляри віртуальних машин і фізичних серверів, ініціалізує генетичний алгоритм, запускає оптимізацію та виводить результати.

Структура програмної частини на мові Python зображена на рисунку 1.

```python
# main.py

from virtual_machine import VirtualMachine
from physical_server import PhysicalServer
from genetic_algorithm import GeneticAlgorithm

# Створення віртуальних машин
vm1 = VirtualMachine(1, {'CPU': 4, 'Memory': 8, 'Disk': 100})
vm2 = VirtualMachine(2, {'CPU': 2, 'Memory': 4, 'Disk': 50})

# Створення фізичних серверів
server1 = PhysicalServer(1, {'CPU': 10, 'Memory': 16, 'Disk': 200})
server2 = PhysicalServer(2, {'CPU': 8, 'Memory': 12, 'Disk': 150})

# Створення екземпляра генетичного алгоритму
genetic_algorithm = GeneticAlgorithm([vm1, vm2], [server1, server2])

# Запуск генетичного алгоритму
solution = genetic_algorithm.run()

# Виведення результатів
print("Optimal Assignment:")
for vm, server in solution.items():
    print(f"VM {vm.id} -> Server {server.id}")
```

Рисунок 1 - Програмний код методу на мові Python

У цьому прикладі бібліотека DEAP використовується для створення популяції, обчислення пристосованості (фітнес-функції), селекції, кросовера та мутації в рамках генетичного алгоритму. Процес оптимізації проводиться на основі пристосованості, що визначає, наскільки ефективно розподілені віртуальні машини на фізичні сервери.

Висновок

Дослідження підтвердило ефективність запропонованих математичних моделей та програмного забезпечення для розподілу віртуальних ресурсів. Запропоновані рішення сприяють підвищенню продуктивності та ефективності обчислювальних систем.

Список використаних джерел

1. Garrison J., Nova K. Cloud Native Infrastructure. - O'Reilly Media, 2017. - 160 p.
2. Dac-Nhuong Le, Raghvendra Kumar, Gia Nhu Nguyen, Jyotir Moy Chatterjee. Cloud Computing and Virtualization. 1st Edition. Wiley-Scrivener, 2018. – 234 p.
3. A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, “Population size in particle swarm optimization,” Swarm and Evolutionary Computation, vol. 58, Article ID 100718, 2020.
4. X. Zhang, H. Liu, and L. Tu, “A modified particle swarm optimization for multimodal multi-objective optimization,” Engineering Applications of Artificial Intelligence, vol. 95, 2020.
5. M. Masdari, S. S. Nabavi, and V. Ahmadi, “An overview of virtual machine placement schemes in cloud computing,” Journal of Network and Computer Applications, vol. 66, pp. 106–127, 2016.