УДК 004.7
DOI https://doi.org/10.32838/2663-5941/2021.1-1/13

Киричек Г.Г.
Національний університет «Запорізька політехніка»

Гаркуша В.Ю.
Національний університет «Запорізька політехніка»

ВІРТУАЛІЗАЦІЯ ХОСТІВ НА ОСНОВІ PROXMOX VE В УМОВАХ НАДЛИШКОВОГО ВИКОРИСТАННЯ РЕСУРСІВ

На даний час Proxmox є вільним програмним забезпеченням, яке використовується під час виконання завдань із віртуалізації на базі QEMU\KVM. Він дозволяє розгортати та керувати віртуальними машинами і контейнерами, включає інструменти вебконсолі та командного рядка і надає API REST для сторонніх інструментів. При цьому підтримуються два типи віртуалізації: на основі контейнерів із LXC і з підтримкою повної віртуалізації з KVM. У роботі реалізовано засіб для проведення досліджень апаратної віртуалізації шляхом використання технологій Intel Hyper-V або AMD-V у процесорі. Метою роботи є проведення аналізу принципів віртуалізації і механізмів розподілення ресурсів у Proxmox VE та його модифікація для оптимізації використання ресурсів сервера системи. Об'єктом дослідження є система віртуалізації хостів на базі Proxmox VE в умовах надлишкового використання ресурсів. Предметом дослідження є моделі, методи, інструментальне та програмне забезпечення для реалізації системи віртуалізації хостів на базі Proxmox VE. Авторами для виконання віртуалізації обрано підхід із використанням QEMU-емулятора шляхом створення віртуальних машин із самостійним ядром. Під час проведення аналізу та задля порівняння обрано такі системи: RedHat OpenStack, Proxmox VE та Kubernetes. Основними завданнями роботи є порівняння та вибір типу віртуалізації; моделювання загальних модулів системи; реалізація алгоритмів та методів її розроблення; вибір методу конфігурації сервера та апаратного складника системи; конфігурування програмного комплексу Proxmox VE та дослідження характеру і динаміки використання ресурсів; аналіз отриманих результатів. Система підтримує конфігурацію високої доступності та систему кворум. Головним елементом архітектури є вузол, який виконує функцію керування та використання віртуальних машин, має підтримку шаблонування і менеджменту пам'яті віртуальних машин. Базовою абстракцією є віртуальна машина, яка має виділене ядро та підтримує власні модулі цього ядра.

Ключові слова: *віртуалізація, вузол, віртуальна машина, сервер, програмне забезпечення, ядро.*

Постановка проблеми. Сьогодні реалізація та підтримка власного датацентру для малого та середнього бізнесу є неефективним. Тому більш раціонально використовувати послуги хмарних провайдерів типу Infrastructure-as-a-Service (IaaS). Для цього необхідно орендувати віртуальну машину, яка своїми можливостями не поступається звичайний сервер, а навіть перевершує його. Саме підхід із запуском великої кількості віртуальних машин на одному потужному сервері зараз є найбільш актуальним [1].

Розрізняють програмну та апаратну віртуалізації, а у апаратній – паравіртуалізацію та повну. Повна, окрім операційних систем (далі – ОС) та програм, підтримує емуляцію апаратних засобів. Паравіртуалізація – запуск ОС без емуляції апаратних засобів та координування з апаратними засобами через хуки до головної системи кінцевого вузла. Для апаратної – необхідно мати спеціальний процесор, який підтримує технології Intel VT або AMD-V. А програмна означає, що для ОС віртуальної машини не створюється нове ядро ОС, тобто модулі фізичного ядра використовуються у віртуальному. Авторами для дослідження всіх особливостей віртуалізації пропонується спроектувати сервер віртуалізації [2].

Аналіз останніх досліджень та публікацій. У світі інформаційних технологій (далі – ІТ) віртуалізація серверів є перспективною та важливою темою, що потребує виконання низки завдань у цьому напрямі. Сьогодні є деякі методи та способи їх реалізації [3–5]. При цьому кожне окреме програмне забезпечення (далі – ПЗ) зазвичай використовує мінімальну апаратну конфігурацію мережі, кінцевих вузлів, серверу та сховищ даних, які є фізичною платформою під час реалізації хмарного кластера.

RedHat OpenStack є безкоштовною відкритою та хмарною обчислювальною платформою, що поставляється послугою IaaS для клієнтів

у загальнодоступних і приватних хмарах [3]. В основі проекту лежить принцип модульної архітектури. Як приклад модулів розглянуто три обов'язкові модулі Nova, Keystone та Neutron. Nova підтримує створення віртуальних машин, серверів без апаратної частини але має обмежену підтримку системних контейнерів [4]. Neutron – модуль, який забезпечує мережеве підключення «як послугу» між інтерфейсними пристроями, які керуються іншими службами OpenStack та реалізує OpenStack Networking API. Він керує всіма факторними мережами для інфраструктури віртуальних мереж (VNI) та аспектами рівня доступу фізичної мережевої інфраструктури (PNI). При цьому користувачі можуть використовувати програмно-визначені мережеві технології (SDN) для підтримки масштабованості [5]. Keystone є службою, яка забезпечує автентифікацію клієнта API, підключення сервісу та розподілену авторизацію з використанням API Identity OpenStack. Вона може інтегруватися із службами каталогів, які підтримують стандартні облікові дані користувача і паролі. Система розроблена для широкого горизонтального розгортання, є стандартом галузі IaaS, має гнучку систему модулів і можливості для вирішення широкого кола завдань але має такі мінуси, як занадто складна архітектура; високий рівень витрат на обслуговування системи і потребу в мінімальному обсязі пам'яті накопичувача в 50 гігабайтів [3].

Kubernetes є системою масштабування та розгортання контейнерів. Вона призначення для реалізації кластера на платформі хмарних провайдерів або на основі bare-metal серверів [6]. Система використовує абстракцію контейнерів для забезпечення гнучкого використання ресурсів та має розподілену архітектуру. Вона є досить відмовостійкою та універсальною але має труднощі в налаштуванні внутрішньої мережі кластера, контролюванні доступу до кластера (rbac система) та менш ізольовану абстракцію контейнера замість kvm.

Proxmox є вільним програмним забезпеченням, яке використовується під час виконання завдань із віртуалізації на базі QEMU\KVM (дистрибутив на базі Debian із модифікованим ядром Ubuntu LTS) [7]. Він дозволяє розгортати та керувати віртуальними машинами і контейнерами, включає інструменти вебконсолі та командного рядка і надає API REST для сторонніх інструментів. При цьому підтримуються два типи віртуалізації: на основі контейнерів з LXC і з підтримкою повної віртуалізації з KVM [8].

Постановка завдання. Метою роботи є проведення аналізу принципів віртуалізації і механізмів розподілення ресурсів у Proxmox VE та його модифікація для оптимізації використання ресурсів сервера системи. Об'єктом дослідження є система віртуалізації хостів на базі Proxmox VE в умовах надлишкового використання ресурсів. Предметом дослідження є моделі, методи, інструментальне та програмне забезпечення реалізації системи віртуалізації хостів на базі Proxmox VE. Під час проведення аналізу та задля порівняння обрано такі системи: RedHat OpenStack, Proxmox VE та Kubernetes. Під час проведення досліджень та реалізації загальних етапів роботи необхідно виконати такі завдання: провести порівняння та вибір типу віртуалізації; здійснити моделювання загальних модулів системи; виконати реалізацію алгоритмів та методів її створення; вибрати методи конфігурації серверу та апаратного складника системи; здійснити конфігурування програмного комплексу Proxmox VE та дослідити характер і динаміку використання ресурсів системи, а також провести аналіз отриманих результатів.

Ця система підтримує конфігурацію високої доступності (High Availability), систему кворуму, шаблонування, менеджмент пам'яті віртуальних машин, Cloud-init, а також вона позбавлена недоліків стосовно керування або особливих потреб в інфраструктурі. Вона встановлюється та запускається як звичайний дистрибутив Linux. Базовою абстракцією є віртуальна машина, яка має виділене ядро та власні модулі ядра. Тобто KVM повністю ізольована від системного ядра ОС (на відміну від контейнерів) (рис. 1). Головний елемент архітектури – вузол (нода), яка виконує функцію керування та використання віртуальних машин [9].

Виклад основного матеріалу дослідження. Процес реалізації та конфігурування умовно розділимо на 3 етапи: проектування апаратної частини, конфігурування сервера та програмного забезпечення. До конфігурування сервера належить налаштування RAID масивів, установлення ОС та сервера віртуалізації Proxmox VE. Налаштуванням програмної частини є конфігурація Proxmox VE, створення першої віртуальної машини та на її основі її шаблонів, реалізація і дослідження інших віртуальних машин (рис. 2).

Система віртуалізації проектована як набір віртуальних машин для різних потреб користувачів, а саме: запуску програмного забезпечення у ізольованому середовищі; управління віртуальними машинами та використання більшої кількості

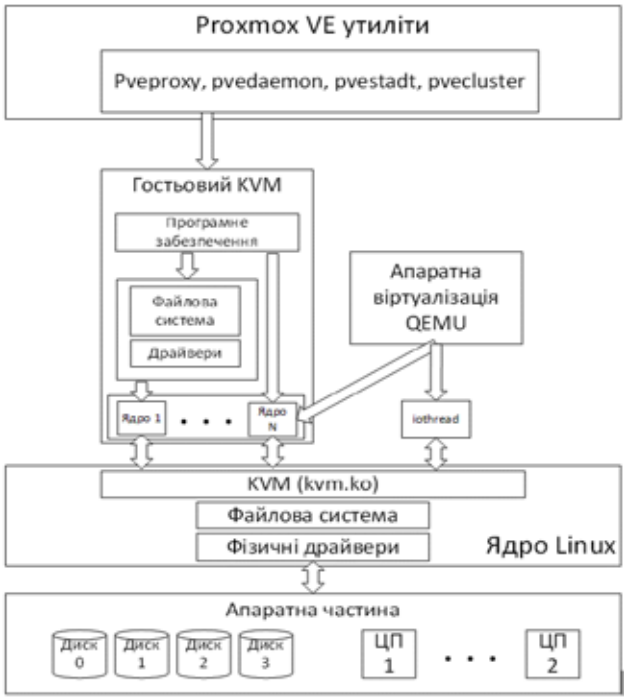

Рис. 1. Модель роботи Proxmox

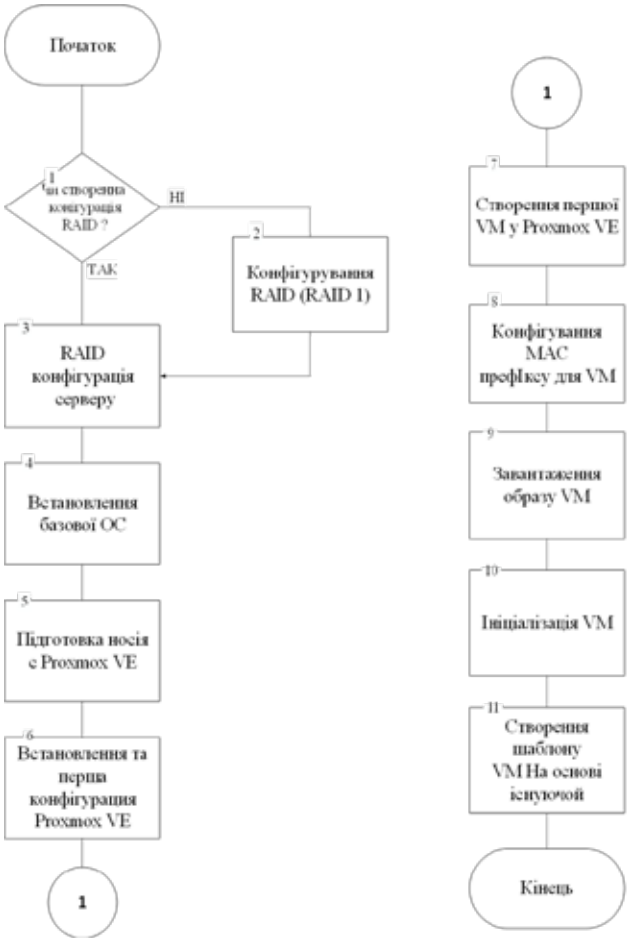

Рис. 2. Алгоритм реалізації системи віртуалізації на основі Proxmox VE

віртуальних машин ніж передбачено фізичними можливостями системи [9].

Апаратний складний системи реалізовано на основі сервера HPE ProLiant DL380 Gen10 8LFF with Universal Media Bay Configure-to-order Server та додаткових фізичних складників: відеокарти MSI GeForce GTX1650 4GB DDR5 OC Low Profile; жорстких дисків SATA 8TB 7200RPM 6GB/S 256MB GOLD WD8004FRYZ WDC та SSD 2.5" Samsung 860 PRO 512GB SATA V-NAND 3D MLC. Сервер використовує 28 ядер CPU, 56 потоків; 96 ГБ оперативної пам'яті (ОП); 512 ГБ SSD та 8 ТБ HDD накопичувачів. Налаштовано 2 RAID масиви типу 1 (SSD та HDD) та встановлено ОС із базовою конфігурацією Ubuntu Server 18.04.

Proxmox VE складається з набору демонів, які створюють віртуальні машини, забезпечують різні дії з боку менеджменту та забезпечують роботу з боку внутрішньої мережі віртуальних машин: pve-cluster – зберігає конфігурації у нормальному стані та забезпечує їх розповсюдження у кластері; pvedaemon – сервер REST API, обслуговує запит від pveproxy, який прослуховує загальнодоступні порти та працює як некореневий користувач; pveproxy – проксі-сервер REST API, прослуховує порт 8006 та працює від "www-data" користувача, пересилає запит на інші вузли (або pvedaemon); pvestatd – демон статусу PVE, запитує стан усіх ресурсів (віртуальних машин, контейнерів та сховища) і надсилає результат усім членам кластера.

Після встановлення виконуємо запуск першої віртуальної машини, конвертацію її у шаблон, конфігуруємо спеціальні MAC префікси та робимо налаштування мосту між мережами віртуальних машин та сервера системи [10,11]. Для налаштування мосту між мережами віртуальних машин та мережею сервера використовуємо файл конфігурації bridge та модулі «PVE::QemuServer», «PVE::Tools qw(run_command)» та «PVE::Network».

Дослідження та тестування системи. Аналіз поведінки системи віртуалізацій в умовах надлишкового використання ресурсів проводимо в динаміці використання цих ресурсів під час створення та запуску додаткових віртуальних машин, які реалізовані на основі одного шаблону і повинні використовувати однакову кількість апаратних ресурсів системи. Далі наведено принцип роботи системи моніторингу Proxmox VE (рис. 3).

Після створення віртуальних машин та на основі графіків моніторингу вебінтерфейсу і відповідей із сервера отримані результати використання ресурсів сервера у вигляді графіків залежності кількості ресурсів від фактичної кількості

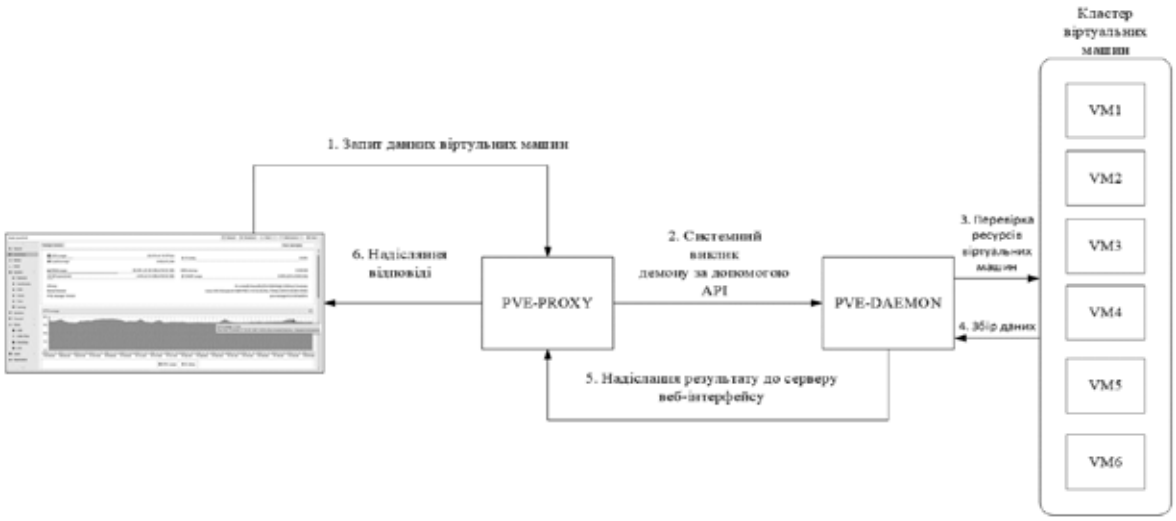

Рис. 3. Схема роботи системи моніторингу

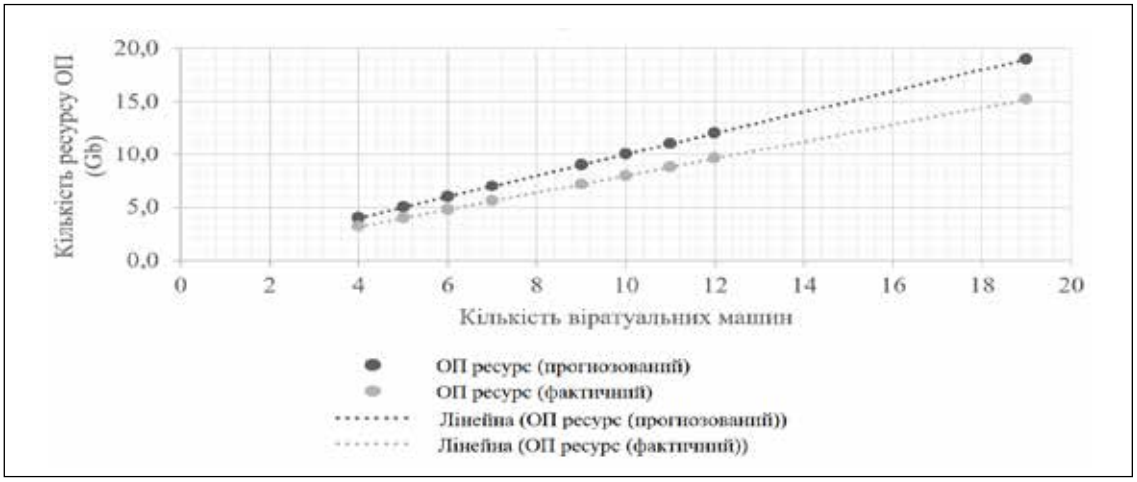

Рис. 4. Залежність кількості віртуальних машин від ресурсу ОП

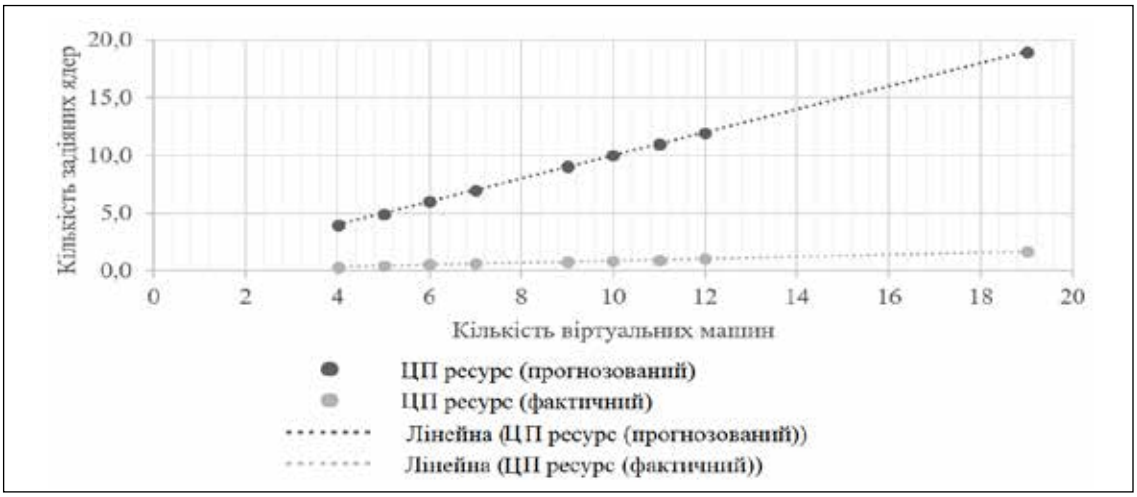

Рис. 5. Залежність кількості віртуальних машин від кількості ядер ЦП

віртуальних машин. На рисунках 4 та 5 показана залежність кількості віртуальних машин від обсягу ресурсу ОП та ядер ЦП.

Проаналізувавши дані системи моніторингу, виявлено, що ресурси ЦП та ОП не є зарезервованими, адже віртуальна машина не використовує

ці ресурси постійно, тому залежність виводимо на основі використання віртуальними машинами ресурсу постійної пам'яті (далі – ПП). Для оптимізації системи реалізуємо додаток, що збирає дані про запущені віртуальні машини та аналізує використання їх ресурсів. Додаток (наведено частково) написаний мовою Python [12] та використовує Proxmox API для аналізу та оптимізації віртуальних машин сервера (рис. 6).

Аналіз даних виконуємо 10 хвилин з інтервалом у 10 секунд. Якщо додаток не виявляє активності віртуальної машини, то виконується її снапшот і вона архівується (видаляється). За необхідності її можна відновити зі зробленого снапшоту. Під час дослідження кластера на наявність активних та неактивних віртуальних машин знайдено лінійну залежність активності загальної кількості машин та машин, які можна оптимізувати. Під час аналізу активності віртуальних машин виконувалась перевірка задіяння ресурсу ЦП та ОП. Наведено величини ресурсів, за яких віртуальну машину можна вважати неактивною: менше 5% використання ресурсу ЦП та менше 1024 Mb використання ОП. На рисунку 7 наведено те, як використ-

```
from time import sleep
import datetime
from proxmoxer import ProxmoxAPI
import pprint
import pydash
wait_time = 1
iterations = 5
timeout=10
def init_func():
      prox = ProxmoxAPI("10.1.0.12:8006", user='root@pam',
                  password='rootroot', verify_ssl=False)
        vm_full_dict = {}
        vm_opt_dict = {}
        tmp_dict_avg_load = {
                "vm_avg_disk_load": 0,
                "vm_avg_ram_load": 0,
                "vm_avg_cpu_load": 0            }
         # Verification
         for i in range(iterations):
         try:
                  res_list = prox.cluster.resources.get()
                  for i in res_list:
                           if i["status"] == "running":
                           vm_full_dict = iterator_get(vm_full_dict, i)
               sleep(wait_time)
               except Exception as e:     print(e)
…
def vm_check(vm_dick):
        min_cpu = vm_dick["vm_max_disk_load"] * 0.15
min_ram = 1024 if vm_dick["vm_max_ram_load"] > 1024 else (1024 * 0.3)
        min_disk = 1400
cpu_state, ram_state, disk_state = False, False, False
        if vm_dick["vm_avg_cpu_load"] < min_cpu:
                disk_state = True
        if vm_dick["vm_avg_ram_load"] < min_ram:
                ram_state = True
        if vm_dick["vm_avg_ram_load"] < min_disk:
                ram_state = True
        if (cpu_state and ram_state) or disk_state:
                return True
       else:   return False
init_func()
```

Рис. 6

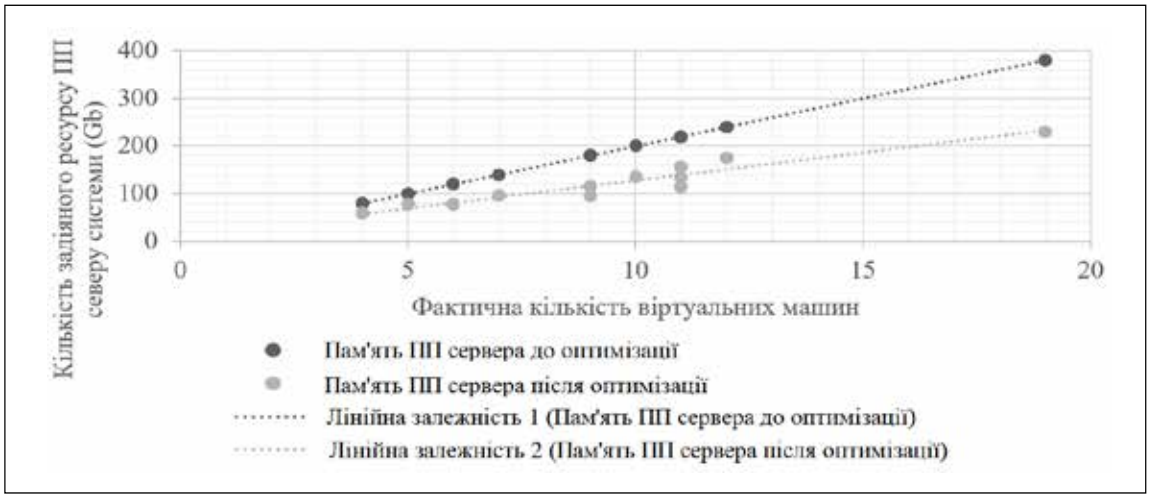

Рис. 7. Залежність ресурсу ПП до та після оптимізації

тання ресурсу ПП позитивно змінюється під час виявлення неактивних віртуальних машин. При цьому після оптимізації резервується 1.36Gb ПП, порівняно з 20Gb за звичайного використання віртуальних машин.

Висновки. У роботі реалізовано систему віртуалізації та досліджено розподілення і характер використання ресурсів її сервера. Виявлено, шо ресурси ОП та ЦП сервера використовують shared-принцип – активне розподілення ресурсів між віртуальними машинами без їх резервування. Удосконалено роботу системи шляхом використання постійною пам'яттю принципу створення логічного диску для резервування частини пам'яті сервера та виконано дослідження характеру і динаміки її роботи. Для збільшення ресурсу ПП сервера, віртуальні машини, що не задіяні, архівуються за допомогою реалізованої оптимізаційної утиліти. Під час реалізації шаблону ми визначили, що після оптимізації резервується 1.36Gb ПП, порівняно з 20Gb за умов звичайного використання віртуальних машин.

Список літератури:

1. Mohanty H., Bhuyan P., Chenthati D. Big data: A primer. Springer, 2015, vol. 11, 183 p.
2. Holovnia O. Criteria for selecting virtualization software in teaching unix-like operating systems. Information technologies in education, 2015, 24, pp. 119–133.
3. Клементьев И., Устинов В. Технологии виртуализации, Интуит, 2015, URL: https://intuit.ru/studies/courses/673/529/lecture/11915.
4. Kumar, R., Parashar, B.B. Dynamic resource allocation and management using OpenStack. Nova 1, 2010, p. 21.
5. Kominos, C. G., Seyvet, N., Vandikas, K. Bare-metal, virtual machines and containers in OpenStack. In ICIN-2017, IEEE, 2017, pp. 36–43.
6. Luksa, M. Kubernetes in action. Island:Manning Publications, 2018, 613 p.
7. Cheng S. Proxmox High Availability. Packt Publishing Ltd, 2014, 258 p.
8. Ahmed W. Mastering Proxmox: Build virtualized environments using the Proxmox VE hypervisor. Packt Publishing Ltd, 2017, 457 p.
9. Chang, B., Tsai, H., Wang, Y., Huang, C. Resilient distributed computing platforms for big data analysis using Spark and Hadoop. In ICASI, 2016, pp. 1–4.
10. Kirichek, G., Kyrychek, D., Hrushko, S., Timenko, A.: Implementation the Protection Method of Data Transmission in Network. In: ATIT-2019, pp. 29–132.
11. Киричек Г.Г., Гаркуша В.Ю. Процес виявлення зловживань і аномалій в мережі. Наукові праці ДонНТУ. Серія: ІКОТ, 2019, 1–2 (28–29), С. 42–46.
12. Kirichek, G., Tymoshenko, V., Rudkovskyi, O., Hrushko, S.: Decentralized System for Run Services. In: CEUR Workshop Proceedings 2353, 2019, pp. 860–872.

Kirichek G.G., Harkusha V.Yu. HOSTS VIRTUALIZATION ON BASED PROXMOX VE IN CONDITIONS EXCESSIVE USE OF RESOURCES

Currently, Proxmox is a free software that is used to solve virtualization problems based on QEMU \ KVM. It allows you to deploy and manage virtual machines and containers, includes web console and command line tools, and provides a REST API for third-party tools. It supports two types of virtualization: the first – based on containers with LXC and the second – with support for full virtualization with KVM. The work implemented method for research on hardware virtualization using Intel Hyper-V or AMD-V technologies in the processor. The aim of the work is to analyze the principles of virtualization and resource allocation mechanisms in Proxmox VE and its modification to optimize the use of system server resources. The object of the research is the virtualization hosts system based on Proxmox VE in the conditions of excessive using resources. The subject of the research are models, methods, tools and software for the implementation of the virtualization hosts system based on Proxmox VE. The authors, for implementation of virtualization, chose the approach using QEMU emulator, by creating virtual machines with independent core. In carrying out analysis and for comparison, selected systems: RedHat OpenStack, Proxmox VE and Kubernetes. The main objectives of the work are: comparison and choose the type of virtualization; simulation general modules of system; implementation of algorithms and methods for its development; choice of server and hardware component configuration method; configuration software system Proxmox VE and study the nature and dynamics of resource use; analysis of the obtained results. The system supports high availability configuration and quorum system. The main of architecture element is a node that performs the function of managing and using virtual machines, has support template creation and memory management of virtual machines. The basic abstraction is a virtual machine that has a dedicated core and supports its own modules of core.

Key words: *virtualization, node, virtual machine, server, software, core.*