DOI: https://doi.org/10.26565/2304-6201-2022-53-02

УДК 004.051

Дослідження альтернативних технологій контейнерів для віртуалізації процесів розміщення компонентів лінійок програмних продуктів

Р.О. Гамзаєв[1], В.Х. Мурадова [1,2], М.В. Ткачук[1]
[1]*Харківський національний університет імені В. Н. Каразіна, майдан Свободи 4, м. Харків, 61022, Україна*
[2]*Харківський національний університет радіоелектроніки, проспект Науки 14, м.. Харків,61166,Україна*

Гамзаєв Рустам Олександрович *кандидат технічних наук, доцент, докторант кафедри моделювання систем і технологій, Харківський національний університет імені В.Н. Каразіна, м.Харків, Майдан Свободи 4, 61022*; e-mail: rustam.gamzayev@karazin.ua; https://orcid.org/0000-0002-2713-5664

Мурадова Всаля Худашірін кизи *кандидат технічних наук, старший викладач кафедри моделювання систем і технологій, Харківський національний університет імені В.Н. Каразіна, м. Харків, Майдан Свободи 4, 61022; e-mail : v.muradova@karazin.ua ;* https://orcid.org/0000-0001-6304- 8325

Ткачук Микола Вячеславович *доктор технічних наук, професор, завідувач кафедри моделювання систем і технологій, Харківський національний університет імені В.Н. Каразіна, м. Харків, Майдан Свободи 4, 61022; e-mail: mykola.tkachuk@karazin.ua;* https://orcid.org/0000-0003-0852-108

У статті пропонується підхід до підтримки гнучкої розробки лінійок програмних продуктів (ЛППП) з використанням методів керування варіабельністю в межах методології Scrum. Головною метою роботи є аналіз контейнерів для віртуалізації середовищ при розгортанні ЛППП. Структурована інформаційна база для запропонованого підходу управління варіабельністю розгортання, показана його роль у загальному методі Scrum та запропоновано концептуальну схему процесу управління на етапі розгортання додатку. Проведено експериментальний аналіз та розраховані метрики для двох типів контейнерів Docker та Vagrant.

Ключові слова: *варіабельність, Docker, Vagrant, варіабельність, контейнеризація, ЛППП*

A study on alternative container-based technologies for virtualization of components deployment in software product lines

R.A. Gamzayev[1], V.Kh. Muradova [1,2], M.V. Tkachuk[1]
[1]*V. N. Karazin kharkiv national university, freedom square 4, Kharkive, 61022, Ukraine*
[2]*Kharkiv national university of radioelectronics, avenue science 14, 61166, Ukraine*

Gamzayev Rustam *PhD, asc. professor, post-doctorate of the Department of Systems and Technology Modeling,V.N. Karazin Kharkiv National University, Kharkiv, Freedom Square 4, 61022; e-mail: rustam.gamzayev@karazin.ua ;* https://orcid.org/*0000-0002-2713-5664*

Muradova Vusala *Candidate of technical sciences, senior lecturer of the Department of Systems and Technology Modeling, V.N. Karazin Kharkiv National University, Kharkiv, Freedom Square 4, 61022; e-mail : v.muradova@karazin.ua;* https://orcid.org/0000-0001-6304- 8325

Tkachuk Mykola *Dc. of techn. science, professor, head of the Department of Systems and Technology Modeling,*
V.N. Karazin Kharkiv National University, Kharkiv, Freedom Square 4, 61022, e-mail: mykola.tkachuk@karazin.ua; https://orcid.org/0000-0003-0852-108

The application containerization approach allows creating virtualization environments that could be used as a code. It allows running application in the isolated container that could be reproduced on any other hardware or cloud environment. One of the benefits of the containerization approach is the possibility to allocate necessary hardware resources like a RAM, CPU and storage. An approach to support agile development of software product lines (SPL) by using variability management

© Гамзаев Р. А., Мурадова В. Х., Ткачук М. В., 2022

techniques within the framework of the Scrum methodology has been proposed in the article. The main goal of the work is to analyze containers for virtualization of the runtime environment when deploying SPL. The information base for the proposed approach to managing the variability of deployment has been structured. The role of the approach in the general method of Scrum has been shown, and a conceptual diagram of the management process at the stage of application deployment has been proposed. The experimental analysis has been carried out and metrics for two types of containers, Docker and Vagrant, have been calculated. The following two metrics, namely, portability and productivity, for both containers have been analyzed. These metrics for the test component software solution have been calculated and executed in cloud environment with different configurations. The portability metric indicates how easily the application can be migrated to other platform basing on the time required to start container with the application. The second metric is the time necessary for the same operations in different container.

***Keywords**: variability, Docker, Vagrant, containerization, SPL*

1. Вступ. Актуальність дослідження

Сучасні системи розробки програмного забезпечення (ПЗ) складаються з ряду сервісів, компонентів і класів, які можуть бути використані повторно. Це особливо важливо при розробці (ЛПП) і вимагає встановлення механізму управління варіабельністю вимог. Часто такі задачі погано формалізовані і вимагають комплексного програмно-аналітичного рішення. Отримані рішення дозволяють досягти певного ступеня подібності розроблюваних структур, як результат можна стверджувати, що відмінності виникають внаслідок варіацій програмних артефактів.

Під час розробки ЛПП, процес управління варіабельністю зв'язує етапи аналізу вимог та етап будуванням певної одиниці ЛПП [1]. На етапі впровадження часто виникає необхідність ефективного конфігурування та налаштування розроблених програмних компонентів з урахуванням ресурсів та інтерфейсів зовнішніх бібліотек, операційних платформ, програмних сервісів тощо. Відсутність процесів накопичення даних про деталі етапів розгортання кожного з повторно використовуваних компонентів призводить до необхідності повторення цих етапів у випадку використання варіабельних компонентів.

Зокрема, у роботі [2] показано, що завдяки наявності зв'язків між вимогами та артефактами, було можливим покращити повноту та точність відстеження вимог при гнучкій розробці ПЗ шляхом розробки знання-орієнтованих моделей та технології накопичення інформації, аналіз та управління даними про вимоги користувачів. Проте запропонований у роботі [2] підхід був застосований для розробки окремого продукту та не має можливості застосовувати дані між різними проектами. Таким чином, метою цього дослідження є розробка підходу до аналізу можливостей забезпечення варіабельності на етапі розгортання компонентів ПЗ шляхом використанням альтернативних сучасних технологій для контейнеризації застосунків.

2. Технологічна схема управління варіабельністю розгортання програмних компонентів в контексті методології Scrum

В [3] була запропонована інтегрована модельно-технологічна схема розробки та супроводу ЛПП, яка поєднує в собі 2-х рівневу концептуальну модель процесів створення будь-якої ЛПП з роботи К. Поля (K. Pohl et al.) [1] з узагальненою технологічною схемою гнучкої розробки ПЗ (напр., за методологією Scrum та ін.), що дозволяє розглядати всі основні фази життєвого циклу (ЖЦ) ЛПП з урахуванням можливостей забезпечення варіабельності відповідних проектних артефактів, а саме:

1) багатовимірних специфікацій вимог користувачів (user stories), які формуються на основі аналізу певної предметної області (ПрО) за участю експертів та з використання методів обробки знань [4];
2) альтернативних доменних моделей (domain models), які будуються на основі опрацювання цих вимог, і які можуть бути використані в подальшому для генерації каркас вихідного коду (source code framework) з необхідним коефіцієнтом його повторного використання [5];
3) різних варіантів архітектури програмних компонентів (software components), шо входять до складу цільової ЛПП, і які мають бути розміщені (components deployment) на відповідній операційній платформі в тій бізнес-організації, яка буде в подальшому використовувати функціональні можливості цієї ЛПП.

В роботах [3 – 5] для автоматизації процесів обробки даних та експертних знань на етапах (1) – (2) були представлені відповідні методи та інструментальні засоби, які структурно і функціонально входять в згадану вище інтегровану модельно-технологічну схему розробки

ЛПП в вигляді контурів управління зі зворотним зв'язком (feedback control loop). Тепер, зважаючи на необхідність забезпечити також і на етапі (3) можливість вибору та використання альтернативних технологій розгортання програмних компонентів, пропонується розширена схема процесів ЖЦ ЛПП у поєднанні с методологією Scrum, яка представлена на рисунку 2.1.

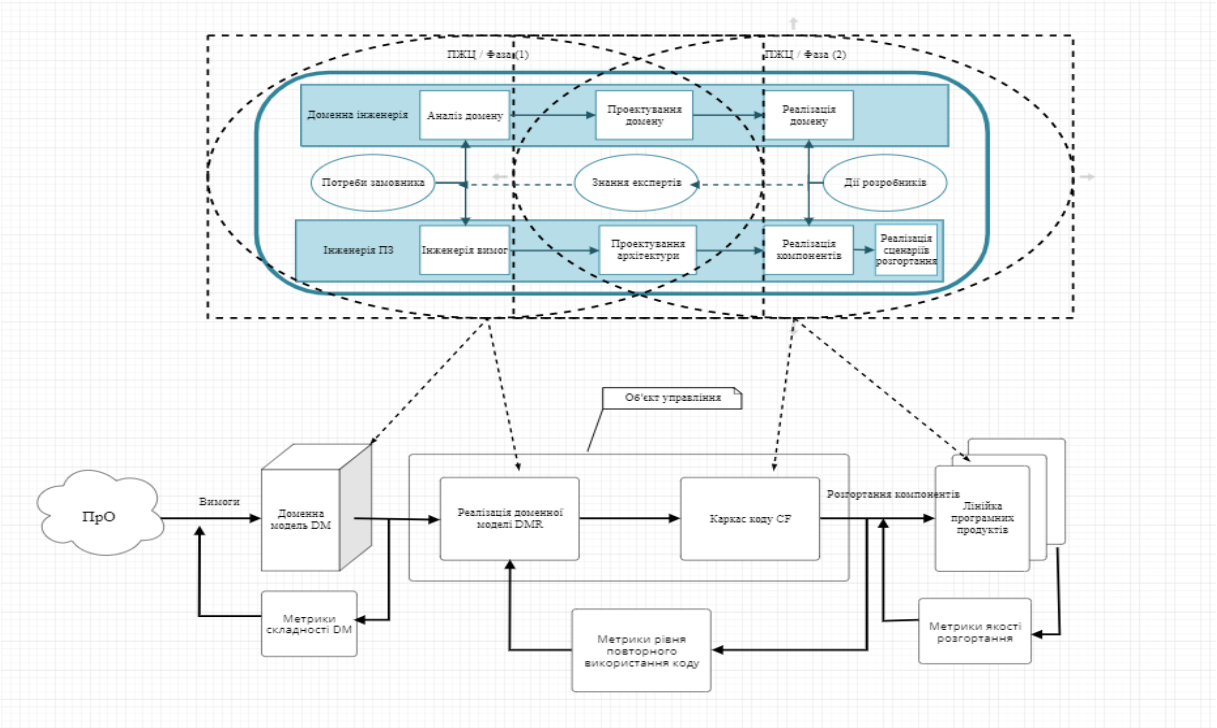

Рис.2.1 Розширена схема процесів ЖЦ ЛПП в контексті методології Scrum

Ця схема містить додатковий конур зворотного зв'язку для управління процесом розгортання програмних компонентів і для забезпечення властивостей варіабельності на цьому етапі ЖЦ ЛПП необхідність розробити підхід до оцінювання якості застосування альтернативних технологій їх розгортання, до яких належать, зокрема, технології контейнеризації та віртуалізації програмних проектів [6]. Ці питання більш детально розглянуті у наступних розділах цієї статті.

3. Методика дослідження процесів розгортання компонентів з використанням альтернативних технологій Docker та Vagrant

Для вирішення поставлених задач цього дослідження необхідно розробити достатньо функціонально повний прототип компонентного прос[грамного рішення (КПР), для розгортання якого застосовуються альтернативні технології контейнеризації та віртуалізації проектів, а також запропонувати відповідні метрики обчислення показників якості виконання цих процесів.

3.1. Прототип тестового КПР для дослідження альтернативних технологій розгортання програмних проектів

Для дослідження особливостей процесів розгортання тестового КПР з використанням альтернативних технологій контейнеризації була розроблена архітектура, яка реалізує функціональні можливості системи адаптивного управління вимогами на основі використання моделі сфокусованого інтерфейсу розробника [7], і вона представлена на рис. 3.1 у вигляді UML – діаграми розгортання компонентів (component deployment diagram).

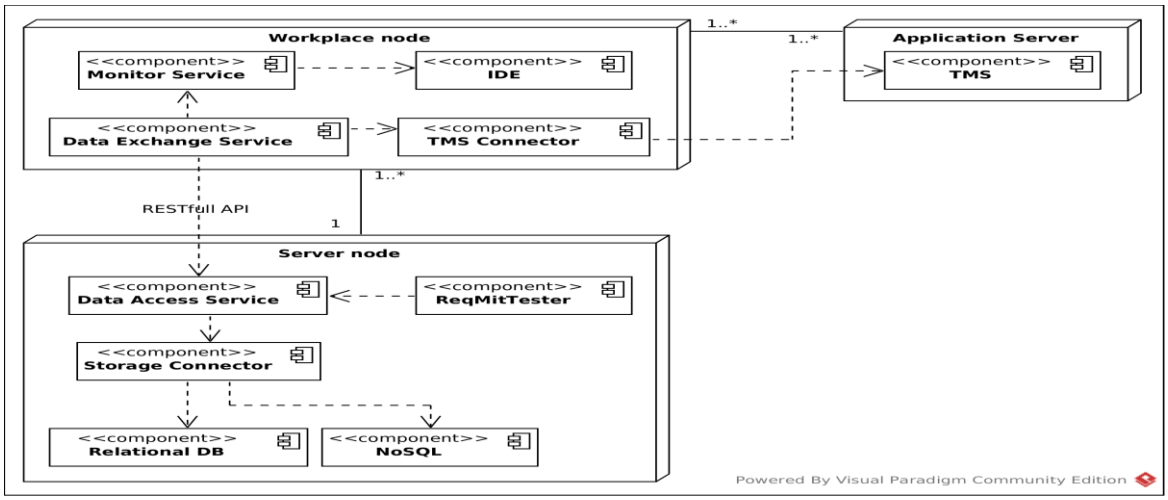

Рис. 3.1 Діаграми розгортання компонентів тестового КПР

У цьому КПР реалізовані наступні основні функціональні можливості: на вузлі Server Node розміщено компоненти Data Access Service є контролером, який займається обробкою запитів від клієнтських застосунків, а також забезпечує доступ з використанням сервісу Storage Connector до ресурсів даних, що знаходяться під управлінням реляційної СКБД Relation DB (MySQL) та альтернативної СКБД NoSQL (Cassandra), і нарешті, компонент ReqMitTester дозволяє розраховувати відповідні метрики і на підставі цього будувати сфокусований інтерфейс розробника ПЗ [7].

Для розгортання цього тестового КПР були написані відповідні скриптові сценарії для технології Docker та Vargant (див. рисунок 3.2).

docker.sh

```
cp docker/containers/app/.env.example docker/containers/app/.env
cp docker/containers/db/.env.example docker/containers/db/.env
docker network create reqmit2-auth-dbnet
docker-compose up -d testdb
docker-compose -f docker/build.yml build
docker-compose build
docker-compose -f docker/build.yml up --force-recreate --no-deps build
docker-compose up -d db cassandra
docker-compose up -d --force-recreate --no-deps api
docker exec -t reqmit2-auth-api ./gradlew flywayMigrate --info
```

vagrant.sh

```
service mysqld start
service cassandrad start
mysql -u mysql -c DROP DATABASE test; CREATE DATABASE test;
./gradlew clean flywayClean check build flywayMigrate
vagrant up
```

Рис. 3.2 Сценарій розгортання КПР з використанням Docker та Vagrant

3.2. Показники для оцінки якості процесів розгортання КПР та метрики їх визначення

Для аналізу використання Docker та Vagrant для розгортання тестового КПР (див. рисунок 3.2) розглядались наступні показники якості [8]:

- переносимість (Portability - Prb);
- продуктивність (Performance - Prm).

Переносимість є мірою того, наскільки легко відповідне КПР може бути перенесена з однієї операційної платформи на іншу, тобто кількісно вона може бути обчислена як величина, що є зворотно пропорційною витратам часу, який є необхідним для здійснення цього процесу. Таким чином, метрика для *Prb* має наступний вигляд:

$$Prb = \frac{1}{T\,(Deployment)}\,, \tag{1}$$

де *T(Deployment)* - це витрати часу на розгортання нової версії проекту.

Беручи до уваги деякі технологічні особливості процесу контейнеризації [9] розрахунок за формулою (1) зводиться до визначення витрат часу безпосередньо на процес розгортання операційного середовища (контейнера) «*T(Environment deployment)*» та витрат часу на налаштування та підготовки до розгортання відповідного серверу додатків «*T(Config. and preparation)*». В свою чергу, час «*T(Environment deployment)*» складається з часу, необхідного для створення (build) контейнеру чи віртуальної машини, та часу для їх безпосереднього виконання (run).

Таким чином, для варіанту застосування технології Docker витрати часу на процесу розгортання має наступний вигляд:

$$T\,(Environment\ deployment)_{Docker} = T(Docker\ build) + T(Docker\ run), \tag{2}$$

а для варіанту застосування технології Vagrant вони обраховуються за формулою:

$$T\,(Environment\ deployment)_{Vagrant} = T(Vagrant\ setup) \tag{3}$$

де «T(Vagrant setup)» - це час створення та запуску віртуальної машини Vagrant.

Отже, враховуючи вирази (1) – (3) кінцевий вираз для обчислення метрики оцінки показника якості переносимості *Prt* розраховується за формулою:

$$Prb = \frac{1}{T\big((Env\ deployment)\big) + T\big((Config.\ and\ preparation)\big)} \tag{4}$$

Для обчислення загальної метрики оцінки продуктивності *Prm* процесу розгортання в даній роботі використовуються наступні параметри [10]:

- метрика для визначення ступеню використання центрального процесора M(CPU);
- метрика для визначення об’єму оперативної пам’яті M(RAM);
- метрика для визначення середнього завантаження операційної системи M(LoadAvg).

Метрика *M(CPU)* може бути розрахована як середнє навантаження на центральний процесор внаслідок виконання певних операцій:

$$M(CPU) = 1 - \frac{\sum_{i=1}^{N}(U_{avg}(CPU)_i)}{N} \tag{5}$$

де $U_{avg}(CPU)_i$ це середнє навантаження на центральний процесор для кожної операції; а N - кількість виконаних операцій.

Метрика M(RAM) дозволяє визначити об’єму оперативної пам’яті, яка має бути виділена для розгортання КПР з використанням однієї з альтернативних технологій Docker та Vagrant, і вона має наступний вигляд:

$$\mathrm{M(RAM)} = \frac{1}{V_2(\mathrm{RAM}) - \mathrm{V_1(RAM)}} \tag{6}$$

де $\mathrm{V_1(RAM)}$ - це об’єм оперативної пам’яті, яка використовується до початку процесу розгортання відповідного КПР, а $\mathrm{V_2(RAM)}$ - це об’єм оперативної пам’яті, яка задіяна після завершення процесу його розгортання.

Аналогічним чином може бути визначена метрика середнього завантаження операційної системи *M(LoadAvg)* під час процесу розгортання КПР [10], а саме:

$$\mathrm{M(Load\ Avg)} = \frac{1}{L_2 - L_1} \tag{7}$$

де, L_1 це середнє навантаження системи до початку процесу розгортання відповідного КПР, а L_2 – середнє навантаження після його завершення.

Для обчислення остаточного значення метрики *Prm* на основі окремих метрик (5) – (7) пропонується побудувати їх згортку у вигляді

$$Prm = k_1 M(CPU) + k_2 M(RAM) + k_3 M(Load\ Avg), \tag{8}$$

де відповідні вагові коефіцієнти k_1 , k_2 , k_3 можуть бути визначені з використанням одного з експертних методів, наприклад, за методом аналізу ієрархій (MAI) [11] .

4. Проведення програмних експериментів та аналіз отриманих результатів

Для проведення обчислювальних експериментів використовувався екземпляр хмарного сервісу (cloud instance) «Amazon EC2 m4.2xlarge» [12], який мав 32 Гб оперативної пам'яті та віртуальний процесор 8vCPU. Програмний експеримент з оцінюванням параметра переносимості *Prb* було проведено 5 разів, і відповідний алгоритм в вигляді UML – діаграми діяльності представлено на рисунку 4.1.

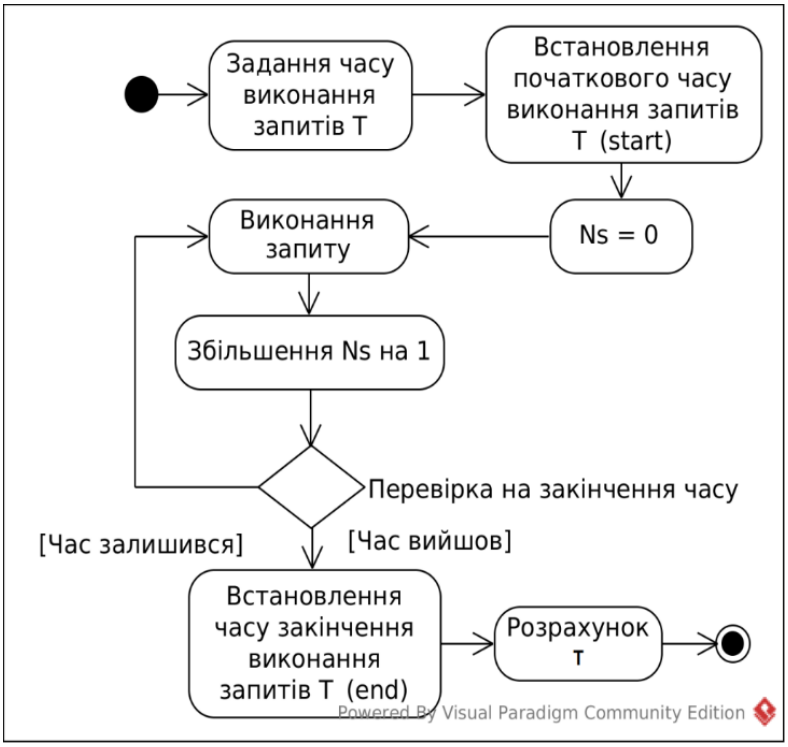

Рис.4.1 Алгоритм проведення експерименту з обчислення метрики Prb

Результати цих експериментів для обох технологій: Docker та Vagrant приведені в Таблиці 1. Де T_1 та T_2 час початку та закінчення експерименту відповідно заданий у Unix часі. ΔT – час на розгортання застосунку у відповідному контейнеру.

Таблиця 1. Результати експериментів для оцінки метрики переносимості

№	T_1(Docker)	T_2(Docker)	ΔT (Docker)	T_1(Vagrant)	T_2(Vagrant)	ΔT (Vagrant)
1	1527592245	1527592498	253 с	1527593552	1527593879	327 с
2	1527592498	1527592741	243 с	1527593879	1527594168	289 с
3	1527592741	1527592978	237 с	1527594168	1527594479	311 с

4	1527592978	1527593245	267 с	1527594479	1527594812	333 с
5	1527593245	1527593552	307 с	1527594812	1527595091	279 с
T(P(Deployment with Docker))			261 с	T(P(Dep. with Vagramt))		290 с

Використовуючи дані з Табл. 1 та формули (1) – (4) були отримані наступні значення для оцінки якості процесу розгортання тестового КПР за метрикою *Prb*

$$Prb\ (\text{Docker}) = 0.0038, \quad Prb\ (\text{Vagrant}) = 0.0035, \tag{9}$$

Тобто, значення метрики переносимості *Prb* при використанні технології Docker більше значення цієї ж метрики при використанні технології Vagrant лише приблизно на 8%, що є досить незначною різницею.

Експеримент для обчислення значення метрики продуктивності *Prm* проводився наступним чином: програмним шляхом виконувались 100000 запитів обробки даних з використанням функціоналу тестового КПР (див. рисунок 4.1), збирались відповідні дані, а потім розраховувались показники метрик з використанням формул (5) – (7), і ці результати наведені в Таблиці 2.

Таблиця 2. Результати проведеного експерименту

Метрика / Технології	Docker	Vagrant
M(CPU)	0.63	0.56
M(RAM)	0.00049	0.00041
M(Load AVG)	0.14	0.083

Наступним кроком для визначення кінцевого значення метрики оцінки продуктивності *Prm* для варіантів використання Docker та Vagrant є застосування MAI за формулою (8). Для цього спочатку була побудована матриця попарних порівнянь пріоритетів для альтернатив Docker / Vagrant (див. Таблиця 3).

Таблиця 3. Матриця парних порівнянь альтернатив за методом MAI

	M(Load AVG)	M(RAM)	M(CPU)
M(Load AVG)	1	3/1	5/4
M(RAM)	1/3	1	3/6
M(CPU)	4/5	6/3	1

Наступним кроком є визначення значень вагових коефіцієнтів k_1 , k_2 , k_3 для застосування у формулі (8), див. Таблицю 4.

Таблиця 4. Значення вагових коефіцієнтів для методу MAI

	M(Load AVG)	M(RAM)	M(CPU)	Сума по рядкам	Вагові коефіцієнти k_1 , k_2 , k_3
M(Load AVG)	1	3	1.25	5.25	0.48
M(RAM)	0.33	1	0.5	1.83	0.17
M(CPU)	0.8	2	1	3.8	0.35

З використанням окремих метрик продуктивності з Табл. 3, значень вагових коефіцієнтів з Табл. 4 та застосовуючи формулу (8), були отримані комплексні оцінки метрики Prm для технологій Docker і Vagrant відповідно

$$\begin{aligned} Prm(\text{Docker}) &= 0.48 \cdot 0.14 + 0.17 \cdot 0.00049 + 0.35 \cdot 0.63 \approx 0.29 \\ Prm(Vagrant) &= 0.48 \cdot 0.083 + 0.17 \cdot 0.00041 + 0.35 \cdot 0.56 \approx 0.24 \end{aligned} \tag{10}$$

Тобто, аналізуючи результати метрики *Prm з* виразу (9), можна зробити мотивований висновок, що продуктивність функціонування тестового КПР, який було розгорнуто за допомогою контейнерної технології Docker, є приблизно на 17% вищою, ніж у випадку його розгортання з використанням технології віртуалізації Vagrant.

5. Виводи та напрямки подальших досліджень

Як було показано у цієї статті контейнеризація с використанням Docker має значну перевагу з точки зору портативності, для аналізу продуктивності було розраховано кількісне значення завдяки порівнянню різних критеріїв з використанням методу аналізу ієрархій, а отримане значення також показало перевагу Docker контейнеризації. В подальшому планується розробити декілька ЛПП з використанням різних архітектурних підходів, та дослідити ефективність використання контейнеризації, а також проаналізувати оптимальну структуру поділу системи.

ЛІТЕРАТУРА

1. K. Pohl, et al: *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005, 467 pp.
2. Гамзаєв Р. О. *Модель та інформаційна технологія побудови адаптивної матриці трасування вимог у гнучких процесах розробки програмного забезпечення* / Р. О. Гамзаєв, М. В. Ткачук // *Вісник Нац. техн.* ун-*ту "ХПІ" : зб. наук. пр. Темат. вип. : Системний аналіз, управління та інформаційні технології.* – Харків : НТУ "ХПІ". – 2013. – № 2 (976). – С. 49-59.
3. Гамзаєв Р.О., Ткачук М.В., Шевкопляс Д.О. *Застосування знання-орієнтованих методів і технологій для моделювання варіабельності в розробці лінійок програмних продуктів* // Матеріали міжн. науков-техн. конференції КМНТ-2021, (м. Харків, 23-25 квітня 2021 року) – Х.: ХНУ імені В.Н. Каразіна, 2021. - С. 104-107.
4. Gamzayev R.O., Tkachuk M.V., Shevkoplias D.O. *Handling of Expert Knowledge in Software Product Lines Development with Usage of Repertory Grids Method* // *Вісник Харківського національного університету імені В.Н. Каразіна, Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. - № 47, 2020. – С. 13-24.
5. Мартінкус І. О., *Ткачук* М. В., Гамзаєв Р. О. Конструювання лінійок програмних продуктів із застосуванням доменного моделювання та метрик повторного використання коду / І. О. Мартінкус, М. В. Ткачук, Р. О. Гамзаєв // Системи управління, навігації та зв'язку: зб. наук. пр. ЦНДІ НУ. – К., 2017. – Вип. 3(43).– С. 93-97.
6. Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An Updated Performance *Comparison* of Virtual Machines and Linux Containers // IBM Research Report, RC25482 (AUS1407-001) July 21, 2014.
7. Гамзаев Р. О., Мета-модель процесса трассировки требований при разработке программного обеспечения / Гамзаев Р. О., Ткачук Н. В., Мартинкус И. О. // Вісник НТУ «ХПІ». Серія: Нові рішення в *сучасних* технологіях. – Х: НТУ «ХПІ», – 2014. – 26 (1069). – С.121-128
8. Андон Ф.И. Основы инженерии качества программных систем / Андон Ф.И., Коваль Г.И., Коротун Т.М. и др – 2-е изд. – К.: Академпериодика, 2007 – 672с.
9. D. N. Jha, S. Garg, R. Ranjan et al. A Study on the Evaluation of HPC Microservices in Containerized Environment // Concurrency and Computation Practice and Experience 33(1) Published 2 May 2019
10. Miguel G. Xavier; Marcelo V. Neves; Fabio D. Rossi; et al. Performance Evaluation of Container-based Virtualization for High Performance Computing Environments // 21st Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2013, pp. 233 – 240.
11. Саати, Т.Л. Принятие решений. Метод анализа иерархий / Саати Т.Л.; пер. с англ. – М.: Радио и связь, 1993. – 278 с.
12. Офіційний сайт компанії Amazon. - URL: https://aws.amazon.com/ru/ec2/instance-types (дата звернення:05.12.2021).

REFERENCES

1. K. Pohl, et al: *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005, 467 pp.
2. Gamzaev R.O. Model and information technology for building an adaptive matrix of tracing requirements in flexible software development processes / R.O. Gamzaev, M.V. Tkachuk // Visnyk Nats. tech. HPI University: Coll. Science. etc. Topic. issue : Systems analysis, management and information technology. - Kharkiv: NTU "KhPI". - 2013. - № 2 (976). - P. 49-59. [in Ukrainian]
3. Gamzaev R.O., Tkachuk M.V., Shevkoplyas D.O. Application of knowledge-oriented methods and technologies for modeling variability in the development of software product lines // Materials int. scientific and technical conference KMNT-2021, (Kharkiv, April 23-25, 2021) - Kh .: KhNU named after V. N. Karazina, 2021. - P. 104-107. [in Ukrainian]
4. Gamzayev R.O., Tkachuk M.V., Shevkoplias D.O. Handling of Expert Knowledge in Software Product Lines Development with Usage of Repertory Grids Method // Visnyk of Kharkiv National University named after V.N. Karazina, Series "Mathematical modeling. Information Technology. Automated control systems ". - № 47, 2020. - P. 13-24.
5. Martinkus I.O., Tkachuk M.V., Gamzaev R.O. Construction of software product lines using domain modeling and code reuse metrics / I.O. Martinkus, M.V. Tkachuk, R.O. Gamzaev // Control, navigation and communication systems: coll. Science. etc. CNDI OU. - K., 2017. - Vip. 3 (43) .– P. 93-97. [in Ukrainian]
6. Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An Updated Performance Comparison of Virtual Machines and Linux Containers // IBM Research Report, RC25482 (AUS1407-001) July 21, 2014.
1. Gamzaev R.O., Meta-model of the process of tracing requirements in software development / Gamzaev R.O., Tkachuk N.V., Martinkus I.O. // Bulletin of NTU "KhPI". Series: New solutions in modern technologies. - X: НТУ «ХПІ», - 2014. - 26 (1069). - P.121-128. [in Russian]
7. Andon F.I. Fundamentals of software systems quality engineering / Andon F.I., Koval G.I., Korotun T.M. etc. - 2nd ed. - K .: Академпериодика, 2007 - 672c. [in Russian]
8. D. N. Jha, S. Garg, R. Ranjan et al. A Study on the Evaluation of HPC Microservices in Containerized Environment // Concurrency and Computation Practice and Experience 33(1) Published 2 May 2019.
9. Miguel G. Xavier; Marcelo V. Neves; Fabio D. Rossi; et al. Performance Evaluation of Container-based Virtualization for High Performance Computing Environments // 21st Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2013, pp. 233 – 240.
10. Saati, T.L. Decision making. Hierarchy analysis method / Saati TL; lane with English - M .: Radio and communication, 1993. - 278 p. [in Russian]
11. Amazon official website. - URL: https://aws.amazon.com/ru/ec2/instance-types/ (accessed on 05.12.2021)

Надійшла у першій редакції 10.01.2022, в останній - 05.03.2022.