УДК 004.42 DOI https://doi.org/10.32782/1995-0519.2023.3.4

ДОСЛІДЖЕННЯ ЗАСТОСУВАННЯ КОНТЕЙНЕРНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗГОРТАННЯ ПРОГРАМ НА СУПЕРКОМП'ЮТЕРАХ

Галина Киричек
кандидат технічних наук, доцент кафедри комп'ютерних систем та мереж
Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, 69063, kirgal08@gmail.com
ORCID: 0000-0002-0405-7122

Владислав Смірнов
студент факультету комп'ютерних наук та технологій
Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, 69063, inko.vlad@gmail.com

Марія Тягунова
кандидат технічних наук, доцент кафедри комп'ютерних систем та мереж
Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, 69063, mary.tyagunova@gmail.com
ORCID: 0000-0002-9166-5897

Статтю присвячено аналізу та дослідженню можливостей використання контейнерних технологій для спрощення та прискорення розгортання програм на суперкомп'ютерах. Розкрито основні характеристики та особливості контейнерних технологій, їхні переваги та недоліки порівняно з віртуалізованим методом розгортання програм, зокрема з погляду ефективності, ресурсоємності, безпеки та відмовостійкості. Також розглянуто застосування Kubernetes як інструменту для керування контейнерами на суперкомп'ютерах. Дослідження показало, що контейнери дають змогу значно спростити процес розгортання програм на суперкомп'ютерах, скоротити час на підготовку програмного забезпечення та зменшити ризик виникнення відмов обладнання. Контейнери також дають змогу ефективно створювати та масштабувати розподілені системи обчислень, що може поліпшити продуктивність виконання завдань у таких середовищах. Контейнеризація також вирішує проблему ізоляції робочих навантажень на суперкомп'ютерах, що поліпшує безпеку та захист даних.

Ключові слова: суперкомп'ютери, кластер, контейнери, Docker, Kubernetes, YAML.

Постановка проблеми. Хмарні обчислення стають усе популярнішими у сучасному світі, оскільки вони як послуга пропонують багато переваг для різних галузей та завдяки ним користувачі з різних країн можуть отримувати доступ до інформації та ресурсів через мережу [1]. Таким чином, хмарні системи сприяють удосконаленню операцій для зменшення різнопланових витрат, що дає змогу підвищити ефективність та цінність послуг для отримання та обробки інформації завдяки об'єднанню та доступності ресурсів [2]. Такі можливості нам надає програмне забезпечення, яке працює за принципами: SaaS (програмне забезпечення як послуга); PaaS (платформа як послуга) та IaaS (інфраструктура як послуга). Хмарні обчислення проводяться на системах High-performance computing (HPC) [3], які мають розподілення навантаження, високу ефективність та швидкість обробки великих даних, де кожна секунда є важливою. Якраз такі можливості і надають суперкомп'ютери, що складаються з великої кількості процесорів, оперативної пам'яті та сховища даних [4]. Подібні великі апаратні можливості вимагають чіткої, структурованої та масштабованої системи управління. Сьогодні найбільшу популярність для застосування такого рішення набуває метод розгортання програм у контейнерах. У статті ми розглянемо та проведемо дослідження системи з відкритим вихідним кодом Kubernetes [5].

Метою роботи є дослідження застосування технології розгортання програм у контейнері на суперкомп'ютері HPC (High performance computing) за допомогою Kubenetes, ураховуючи ефективність, ресурсні витрати, безпеку та відмовостійкість, оскільки актуальність контейнерної технології є високою та конкурентною порівняно із застарілими методами. Завдання роботи полягає у: порівнянні контейнерізації та методу розгортання у віртуальних машинах; про-

веденні аналізу переваг та недоліків використання Kubernetes під час розгортання програм на суперкомп'ютерах HPC (High performance computing); дослідженні роботи Kubernetes за принципами SaaS [6], PaaS [7] та IaaS [8]; налаштуванні та оркеструванні контейнерів на HPC-системах через Kubernetes [5].

Аналіз останніх досліджень і публікацій. Андрю Йонг у доповіді A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds [9] розглядає застосування контейнерних технологій для розгортання програм на суперкомп'ютерах. Контейнеризація являє собою нову стратегію управління програмним забезпеченням у розподілених системах, яка дає змогу розробникам зменшити зусилля, необхідні для розгортання програмних додатків на різних обчислювальних платформах. Контейнери дають змогу точно визначити конфігурацію робочого середовища для програмного коду, включаючи операційну систему, бібліотеки, змінні середовища та залежності. У галузі високопродуктивних обчислень постійно зростає розмаїття апаратних засобів та системних архітектур, що ускладнює розгортання нових додатків. Розробники потребують тестування, налагодження та оптимізації продуктивності для різних HPC-платформ [3]. Окрім того, черги на використання потужних суперкомп'ютерів можуть затримувати розроблення коду та ускладнювати тестування. Це змушує розробників шукати маломасштабні тестові середовища або використовувати зовнішні ресурси для розроблення та експлуатації високопродуктивних обчислень, тому застосування контейнерів на суперкомп'ютерах є однією з потенційних сфер досліджень.

У дослідженні Container orchestration on HPC systems through Kubernetes [10] зазначається важливість Kubernetes із системою керування ресурсами для контролю та надсилання завдань на суперкомп’ютери, кластери та мережі TORQUE [11] у хмарному кластері HPC. Завдання з великим обсягом обчислень чи даних отримують покращену продуктивність завдяки ефективному плануванню у хмарному кластері. Уміст вхідного вузла зазвичай представлено у звичайному YAML-форматі [12] Kubernetes замість використання PBS-скрипту TORQUE. При цьому оркестратор може аналізувати розмір завдання, щоб визначити, чи слід його виконувати на HPC-кластерах, та для відправки завдання на HPC, автоматично генерується відповідний вбудований YAML-файл. Kubernetes має гнучку модульну архітектуру, яка абстрагує базову інфраструктуру і дозволяє внутрішнє налаштування. Він підтримує різні фреймворки для обробки великих обсягів даних, такі як Hadoop MapReduce [13], Apache Spark [14], і може інтегруватися з Ansible [4] – популярним інструментом для оркестрування програмного забезпечення у хмарних кластерах. Kubernetes має потужний набір інструментів для керування життєвим циклом додатків, включаючи автоматичне перерозгортання у разі відмови та управління станом, окрім того, у нього є розширена система планування, яка дає змогу вказувати різні планувальники для кожного завдання.

Виклад основного матеріалу. Хмарні обчислення вимагають високої портативності. Контейнеризація забезпечує сумісність середовища, інкапсулюючи додатки разом із бібліотеками, конфігураційними файлами та іншими залежностями. Додатки легко переміщаються та розгортаються між кластерами. Контейнеризація як технологія віртуалізації замість імітації повної операційної системи, як це робить віртуальна машина, застосовує контейнери. Вони використовують лише операційну систему хоста, і ця особливість робить контейнери легшими за віртуальні машини.

Контейнери призначені для запуску мікросервісів, і в одному контейнері зазвичай розміщується один додаток, однак додатки з контейнерами можуть бути і складними. Наприклад, у виробництві виникають ситуації, коли потрібні сотні окремих контейнерів, тому контейнерні-оркестратори повинні забезпечити ефективне надання ресурсів та автоматичне масштабування. Але контейнерна методологія розгортання програм з’явилася не одразу (рис. 1), передумовою стало використання програм із застосуванням віртуальних машин, які мали свої переваги та недоліки.

Розглянемо підхід розгортання програм у віртуальних машинах. Ця концепція дає змогу працювати декільком віртуальним машинам на одному фізичному центральному сервері та запускати багато програм на одній фізичній машині, даючи змогу ізолювати їх під час запуску в різних операційних системах. Віртуальні машини є менш ефективними з погляду розміру та продуктивності. Контейнери схожі на віртуальні машини, але є більш легким варіантом ефективного використання ресурсів, швидкого розгортання, ізольованості, портабельності, масштабованості, оркестрації та керування. Контейнери

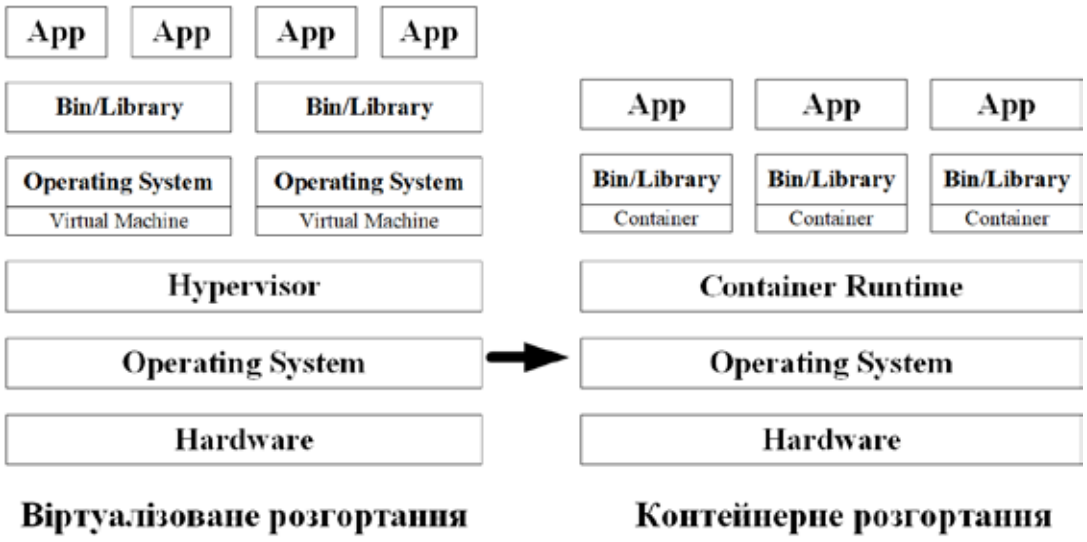

Рис. 1. Еволюція методів розгортання програм

є портабельними, їх можна переносити та запускати на різноманітних платформах, які підтримують контейнеризацію, без необхідності зміни кодової бази та налаштувань, тоді як віртуальні машини вимагають налаштувань та адаптації для кожної платформи окремо. Контейнери мають можливість автоматичного масштабування. Це дає змогу керувати кількістю екземплярів застосунку залежно від навантажень і зберігати надвисоку продуктивність, тоді як віртуальні машини вимагають створення та налаштування кожної машини окремо. Контейнери надають зручні інструменти для керування та оркестрації контейнеризованих застосунків. Система керування контейнерами Kubernetes забезпечує автоматизованість процесів, а віртуальні машини мають власну операційну систему і служби, які необхідно налаштовувати та оновлювати незалежно один від одного. Але контейнерний метод розгортання застосовуються виключно для мікросервісів, тоді як віртуальні машини підтримують монолітні програми, тому розгортання на них програм із парадигмою мікросервісів не є актуальним методом [8; 9]. Виходячи із цього, ми можемо навести перелік переваг контейнерів над віртуальними машинами, це: ефективне використання ресурсів; швидке розгортання; ізоляція та безпека; портабельність та сумісність; гнучкість і масштабованість; латентність та еластичність і можливість створювати контейнери з ізольованим середовищем для тестування.

Після наведеного аналізу маємо, що під час розгортання мікросервісів на суперкомп'ютерах оптимальним є застосування методу розгортання у контейнерах. При цьому необхідно враховувати, що у разі використання одного контейнера можна скористатися платформою на зразок Docker, хоча у високопродуктивних обчисленнях навряд ми обмежимся одним контейнером, бо якщо йдеться про високонавантажені обчислення та великі дані у розподілених системах, виникає критична необхідність у керуванні великою кількістю контейнерів, їх оркестрації, автоматичного розгортання та масштабування. Тому для виконання таких завдань проаналізуємо систему з відкритим кодом Kubernetes, яка має переваги: працює на основі кластерів, які складаються з набору вузлів, що обробляють роботу контейнерів і керуються системою управління; забезпечує самоуправління додатками за допомогою контролерів, що дає змогу автоматично відновлювати та підтримувати бажаний стан додатку в разі відмов або змін у кластері; дає змогу додавати та використовувати різні функціональні модулі системи, де основні компоненти Kubernetes: Kubernetes API Server; Scheduler і Controller Manager можна замінити або розширити під час використання додаткових функцій або інтеграцій; має вбудовані механізми для балансування навантаження між контейнерами (round-robin, least connections або IP-hash), що дає змогу розподілити трафік рівномірно й ефективно; реалізує автоматичну маршрутизацію мережевого трафіку до контейнерів, де правила задаються параметрами: URL-шаблони, піддомени або порти, а Kubernetes маршрутизує трафік; надає засоби управління сховищами даних (для забезпечення постійного зберігання даних використовуються об'єкти PersistentVolume

та PersistentVolumeClaim) та має інтеграцію з різними системами сховищ даних: NFS, iSCSI, AWS EBS тощо.

Але Kubernetes має і недоліки: вона не є системою PaaS (платформа як послуга), бо працює не на апаратному рівні, а на рівні контейнерів; не розгортає вихідний код, робочі процеси CI/CD; не надає програмне забезпечення обробки даних, бази даних та кластерні системи зберігання, як вбудовані служби, які можуть використовуватися через переносні механізми.

IaaS [8] є базовим рівнем інфраструктури, на якому працює Kubernetes. Це можуть бути публічні хмарні постачальники, наприклад Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform (GCP), які надають віртуальні машини або контейнери для кластера Kubernetes, при цьому користувачі використовують обчислювальну інфраструктуру, дисковий простір для зберігання даних або елементи комп'ютерних мереж.

Використання високопродуктивних обчислень зростає, забезпечуючи широке їх застосування для розширеного моделювання та аналізу даних у корпоративних та промислових додатках [3; 10]. Проведемо дослідження з використання Kubernetes для управління високопродуктивними обчисленнями на кластері [5; 8]. Для контейнера як сутності ми застосуємо Docker. Основними кроками є такі (рис. 2): підготуємо кластер, налаштуємо фізичний кластер HPC з обладнанням та інфраструктурою, які включають вузли кластера, мережеве з'єднання та сховища даних; установимо Kubernetes на кластері, включаючи майстер-вузол і робочі вузли, та застосуємо інструменти для керування кластером Kubernetes; визначимо ресурси кластера, такі як кількість робочих вузлів, обсяги пам'яті, обчислювальні можливості і мережеві налаштування; підготуємо контейнери, створимо або використаємо наявні контейнери з необхідним програмним забезпеченням для виконання HPC-завдань; створимо YAML-файли для розгортання HPC-завдань; завантажимо контейнери з HPC-завданнями на робочі вузли та запустимо; використаємо можливості Kubernetes для масштабування кількості робочих вузлів або розподілу завдань залежно від обсягу обчислень та доступних ресурсів; запустимо моніторинг Kubernetes для відстеження продуктивності, навантаження і стану робочих вузлів, а для оцінки результатів зупинимо HPC-завдання та очистимо ресурси кластера.

У коді реалізації під час використання потрібно вказати реальні параметри.

Налаштування кластера HPC передбачає фізичне налаштування обладнання, створення мережевого з'єднання та сховища даних, а також установлення Docker та Kubernetes, що можна зробити за допомогою, наприклад, інструменту kubeadm. Нижче наведено етапи програмної реалізації:

– для отримання можливості створення образів контейнерів спочатку запустимо Docker, `install -y docker.io`;

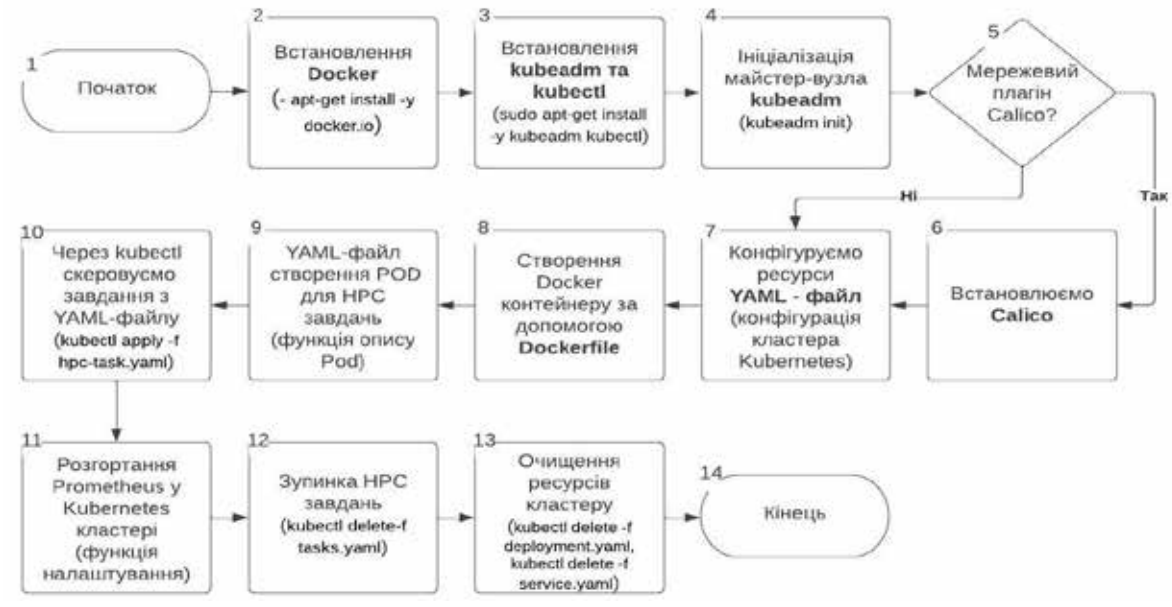

Рис. 2. Алгоритм створення кластера та розгортання контейнеру

– далі встановлюємо kubeadm командою `install -y kubeadm`, завантаживши пакет з ресурсу `packages.cloud.google.com`;

– ініціалізуємо майстер-вузел `kubeadm init`, запускаючи процес налаштування кластера Kubernetes. При цьому відбувається: перевірка залежностей та конфігурація хоста; ініціалізація кластера; запуск контрольної плоскості, що відповідає за керування кластером та прийняття рішення про розміщення та виконання завдань (API-сервер, контролер, планувальник);

– якщо потрібно реалізувати мережеві та маршрутизаційні можливості для контейнерів у кластері, додаємо мережевий плагін, наприклад Calico: `kubectl apply -f`, указавши шлях до `calico.yaml`;

– конфігурація ресурсів налаштовується через YAML – файл, у якому вказано кількість вузлів та їх характеристики, наведено конфігурацію кластера під час ініціалізації та налаштування за допомогою інструмента kubeadm.

На наступному етапі підготуємо та реалізуємо Docker-контейнери, використовуючи Dockerfile. Далі запускаємо YAML-файл («маніфест») для створення Pod із контейнером розгортання HPC-завдань, визначаємо вимоги до ресурсів та обсягів даних.

Після виконаної роботи зі створення ресурсів та їх запуску проведемо аналіз отриманих результатів. Для цього можна використовувати інструменти Kubernetes для моніторингу Prometheus, яка розроблена для збору, візуалізації та аналізу метрик із додатків та інфраструктури. Вона використовує модель збору даних за допомогою HTTP і власну мову запитів PromQL (Prometheus Query Language). У кінці зупиняємо HPC-завдання наведеною командою, яка видаляє усі запущені завдання, описані у YAML-файлі:

```
kubectl delete -f tasks.yaml
```

А очистити ресурси кластера можемо такими командами:

```
kubectl delete -f deployment.yaml
kubectl delete -f service.yaml
```

Фактично вони видаляють розгортання та сервіси, створені для кластера Kubernetes. Таким чином, процес використання Kubernetes для управління високопродуктивними обчисленнями в HPC-середовищі починається з розгортання та конфігурації контейнерів для різних завдань HPC. Моніторинг результатів грає важливу роль у процесі збору та аналізу метрик і допомагає виявити проблеми та поліпшити продуктивність. Оцінка результатів включає порівняння продуктивності з традиційними підходами та оцінкою витрат на інфраструктуру. Це даєзмогу зрозуміти, що використання Kubernetes у HPC-середовищі забезпечує: гнучкість у розгортанні та керуванні завданнями HPC; масштабованість для ефективного використання ресурсів; можливість моніторингу та аналізу результатів. Використання Kubernetes сприяє оптимізації використання ресурсів, зниженню часу розгортання та налаштування середовища.

Висновки. У ході дослідження визначено можливість застосування Kubernetes у середовищі HPC-додатків. Наведено методи, за якими працює Kubernetes у процесі розгортання додатків у контейнерах із метою їх оркестрації, на прикладі контейнерів Docker. Визначено позитивні і негативні боки цієї платформи та те, що Kubernetes працює за принципами SaaS, PaaS та IaaS. Зроблено висновок, що Kubernetes здатен створити, розгорнути, масштабувати та, за необхідності, знищити створені ресурси, ураховуючи те, що перед його застосуванням необхідно провести аналіз завдань та очікуваних результатів, оскільки Kubernetes не є єдиною системою для розгортання та масштабування мікросервісів у HPC-середовищі. Наведено приклад установлення Kubernetes у кластері, створення і розгортання ресурсів, а також проведено очистку та знищення цих ресурсів, застосовуючи kubeadm, YAML-файл для створення маніфесту на HPC-завдання та kubectl для запуску файла. Після проведення моніторингу та аналізу результатів за допомогою Prometheus можемо говорити про зручне та ефективне використання Kubernetes на суперкомп`ютерах. Дослідження використання Kubernetes у HPC-середовищі відкриває нові можливості. Цей підхід покращує продуктивність, забезпечує ефективне використання ресурсів та прискорює розгортання і налаштування.

ЛІТЕРАТУРА

1. Rudkovskyi O.R., Kirichek G.G. Interaction support system of network aplications, CEUR-WS, Vol-2832, November 27, 2020. P. 11–23.

2. Rashid A., Chaturvedi A. Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*. 2019. T. 7. №. 2. P. 421–426.

3. Rodrigo G.P. et al. Towards understanding HPC users and systems: a NERSC case study. *Journal of Parallel and Distributed Computing.* 2018. T. 111. P. 206–221.

4. Киричек Г.Г., Щетінін М.О. Управління конфігурацією серверів на основі Ansible. *Вчені записки ТНУ імені В.І. Вернадського. Серія «Технічні науки».* 2022. Вип. 33(72). № 1. С. 109–114.

5. Kubernetes Meets High-Performance Computing. URL: https://kubernetes.io/blog/2017/08/kubernetes-meets-high-performance/.

6. Palos-Sanchez, P.R., Arenas-Marquez, F.J., & Aguayo-Camacho, M. Cloud computing (SaaS) adoption as a strategic technology: Results of an empirical study. Mobile Information Systems. 2017. T. 2017.

7. Bharany S. et al. Efficient middleware for the portability of paas services consuming applications among heterogeneous clouds. *Sensors*. 2022. T. 22. №. 13. P. 5013.

8. Adhikari M., Amgoth T. Heuristic-based load-balancing algorithm for IaaS cloud. *Future Generation Computer Systems*. 2018. T. 81. P. 156–165.

9. Younge A.J. et al. A tale of two systems: Using containers to deploy HPC applications on supercomputers and clouds. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2017. P. 74–81.

10. Zhou N. et al. Container orchestration on HPC systems through Kubernetes. *Journal of Cloud Computing*. 2021. T. 10. №. 1. P. 1–14.

11. Petrosyan D., Astsatryan H. Serverless High-Performance Computing over Cloud. *Cybernetics and Information Technologies.* 2022. T. 22. №. 3. P. 82–92.

12. Ben-Kiki O., Evans C., Ingerson B. Yaml ain't markup language (yaml™) version 1.1. *Working Draft 2008*. 2009. T. 5. P. 11.

13. Glushkova D., Jovanovic P., Abelló A. Mapreduce performance model for Hadoop 2.x. *Information systems*. 2019. T. 79. P. 32–43.

14. Salloum S. et al. Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*. 2016. T. 1. P. 145–164.

RESEARCH ON THE APPLICATION OF CONTAINER TECHNOLOGIES FOR PROGRAM DEVELOPMENT ON SUPERCOMPUTERS

Halyna Kyrychek
PhD in Technical Sciences, Associate Professor at the Department of Computer Systems and Networks
National University "Zaporizhzhia Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, 69063, kirgal08@gmail.com
ORCID: 0000-0002-0405-7122

Vladyslav Smirnov
Student at the Faculty of Computer Sciences and Technologies
National University "Zaporizhzhia Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, 69063, inko.vlad@gmail.com

Mariia Tiahunova
PhD in Technical Sciences, Associate Professor at the Department of Computer Systems and Networks
National University "Zaporizhzhia Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, 69063, mary.tyagunova@gmail.com
ORCID: 0000-0002-9166-5897

The article is devoted to a review and study of the use of container technologies for deploying programs on supercomputers. In today's scientific and commercial world, there is an increasing need to work in the cloud on distributed highly loaded systems for the process of collecting, analyzing and processing big data, as well as for receiving software services from remote locations. This is due to the need to improve critical aspects in the areas of maritime navigation, construction, and science. Deployment of programs based on the principle of virtual machines has a long and extensive process of creation and maintenance, containers are gaining relevance, because from the point of view of software development for mobility and supportability, microservices have become relevant, against the background of heavy monolithic architectures. The article discusses the basic principles of container technologies, their advantages and disadvantages, including efficiency, resource costs, security, and fault tolerance. The method of containerized deployment of programs on supercomputers is compared, and its advantages over the traditional, virtualized method are demonstrated. The principles and importance of adhering to the SaaS, PaaS and IaaS models that Kubernetes operates on are considered. Also, the direct use of Kubernetes as a tool for deploying and managing containers on supercomputers. The study showed that the use of containers can simplify the deployment of programs on supercomputers, reduce the time for software preparation, and reduce the risk of hardware failure. The use of containers allows you to efficiently create and scale distributed computing systems, which can significantly improve the efficiency of tasks in such environments. We will use Kubernetes as the main tool for creating and orchestrating resources in the HPC environment. Docker is used as a container entity. We will conduct a practical deployment of resources and their management.

Key words: supercomputers, cluster, containers, Docker, Kubernetes, YAML.

REFERENCES

1. Rudkovskyi, O.R., & Kirichek, G.G. (2020). Interaction support system of network aplications. In CEUR Workshop Proceedings (Vol. 2832, pp. 11–23).
2. Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. International Journal of Computer Sciences and Engineering, 7(2), 421–426.
3. Rodrigo, G.P., Östberg, P.O., Elmroth, E., Antypas, K., Gerber, R., & Ramakrishnan, L. (2018). Towards understanding HPC users and systems: a NERSC case study. Journal of Parallel and Distributed Computing, 111, 206–221.
4. Kirichek G.G., & Shchetinin M.O. (2022). Ansible-based servers configuration management. Journal of Academic notes of TNU named after V.I. Vernadskyi. Series «Technical Sciences», 33 (72). Vol 1, 109–114.
5. Kubernetes Meets High-Performance Computing. Available: https://kubernetes.io/blog/2017/08/kubernetes-meets-high-performance/.
6. Palos-Sanchez, P.R., Arenas-Marquez, F.J., & Aguayo-Camacho, M. (2017). Cloud computing (SaaS) adoption as a strategic technology: Results of an empirical study. Mobile Information Systems, 2017.
7. Bharany, S., Kaur, K., Badotra, S., Rani, S., Kavita, Wozniak, M., & Ijaz, M. F. (2022). Efficient middleware for the portability of paas services consuming applications among heterogeneous clouds. Sensors, 22(13), 5013.
8. Adhikari, M., & Amgoth, T. (2018). Heuristic-based load-balancing algorithm for IaaS cloud. Future Generation Computer Systems, 81, 156–165.
9. Younge, A.J., Pedretti, K., Grant, R.E., & Brightwell, R. (2017, December). A tale of two systems: Using containers to deploy HPC applications on supercomputers and clouds. In 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (pp. 74–81).
10. Zhou, N., Georgiou, Y., Pospieszny, M., Zhong, L., Zhou, H., Niethammer, C., & Hoppe, D. (2021). Container orchestration on HPC systems through Kubernetes. Journal of Cloud Computing, 10(1), 1–14.
11. Petrosyan, D., & Astsatryan, H. (2022). Serverless High-Performance Computing over Cloud. Cybernetics and Information Technologies, 22(3), 82–92.
12. Ben-Kiki, O., Evans, C., & Ingerson, B. (2009). Yaml ain't markup language (yaml™) version 1.1. Working Draft 2008, 5, 11.
13. Glushkova, D., Jovanovic, P., & Abelló, A. (2019). Mapreduce performance model for Hadoop 2. x. Information systems, 79, 32–43.
14. Salloum, S., Dautov, R., Chen, X., Peng, P.X., & Huang, J.Z. (2016). Big data analytics on Apache Spark. International Journal of Data Science and Analytics, 1, 145–164. 64.

Стаття надійшла 15.06.2023