DOI: https://doi.org/10.33216/1998-7927-2026-299-1-16-27

УДК 004.75

ВІРТУАЛІЗАЦІЯ ТА КОНТЕЙНЕРІЗАЦІЯ ЯК ЗАСОБИ ОРГАНІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ СЕРЕДОВИЩ

Рязанцев О.І., Кардашук В.С, Бортник К.Я., Барбарук В.М.

VIRTUALIZATION AND CONTAINERIZATION AS MEANS OF ORGANIZING COMPUTING ENVIRONMENTS

Ryazantsev O.I., Kardashuk V.S., Bortnyk K.Y., Barbaruk V.M.

В статті досліджено використання віртуалізації та контейнерізації для організації обчислювальних процесів. Проведено ґрунтовний аналіз технологій віртуалізації та контейнеризації, принципи роботи та сфери застосування. Встановлено, що віртуалізація дозволяє створювати ізольовані обчислювальні середовища з високим рівнем безпеки та гнучкості, проте супроводжується додатковими витратами на ресурси. Контейнеризація забезпечує легкість розгортання, ефективне використання апаратних ресурсів та швидкодію, що робить її особливо актуальною для сучасних DevOps-процесів і хмарних інфраструктур. Розглянуто переваги та недоліки кожного підходу, а також сценарії їх практичного застосування. Здійснено огляд платформ VirtualBox та Docker, які використовуються для реалізації віртуалізованих і контейнерних середовищ відповідно. Проведено порівняльний аналіз функціональних можливостей, переваг і обмежень кожної платформи. Визначено, що VirtualBox забезпечує гнучке налаштування віртуальних машин і підтримку різних гостьових ОС, тоді як Docker орієнтований на легкі, швидкі та масштабовані контейнери. На основі аналізу сформовано експериментальне середовище для тестування продуктивності, що дозволило об'єктивно оцінити ефективність обох технологій у реальних умовах. Виконано практичне дослідження та порівняльний аналіз двох підходів до віртуалізації обчислювальних середовищ: класичної віртуалізації на основі віртуальних машин (VirtualBox) та контейнеризації з використанням платформи Docker. Проведена оцінка ефективності кожного підходу з точки зору продуктивності, використання обчислювальних ресурсів, рівня ізоляції, безпеки та зручності розгортання. Результати практичного дослідження підтвердили, що віртуальні машини та контейнери мають різні області ефективного застосування. Віртуальні машини доцільно використовувати у випадках, коли пріоритетом є максимальна ізоляція та універсальність середовища, тоді як контейнеризація є оптимальним рішенням для швидкого розгортання, ефективного використання ресурсів та побудови масштабованих сучасних інформаційних систем. Отримані результати можуть бути використані для вибору технології віртуалізації залежно від вимог конкретної задачі при практичному застосуванні.

Ключові слова: *віртуалізація, контейнерізація, масштабованість, операційна система, сервер, додаток.*

Вступ. Сучасний етап еволюції інформаційних технологій вирізняється стрімким зростанням обсягів даних, які потребують обробки, підвищенням складності програмних систем та розширенням вимог до гнучкості й продуктивності обчислювальних ресурсів. Для забезпечення високої продуктивності, надійності й масштабованості інформаційних систем дедалі ширше впроваджуються технології віртуалізації та контейнеризації, які спрямовані на оптимізацію використання апаратних ресурсів, ізоляцію середовищ виконання, а також спрощення процесів розгортання й управління сервісами [1].

Технологія віртуалізації дозволяє створювати декілька віртуальних машин на одній фізичній платформі, кожна з яких функціонує у власному операційному середовищі з відокремленими програмними компонентами [2]. Такий підхід сприяє

гнучкому розподіленню ресурсів і забезпечує високий рівень ізоляції, проте супроводжується додатковими накладними витратами, що потенційно можуть впливати на загальну ефективність системи. Як альтернатива, контейнеризація пропонує можливість запуску ізольованих програмних компонентів без вимоги створення окремої гостьової операційної системи. Це значно прискорює розгортання програмного середовища й мінімізує споживання апаратних ресурсів. У контексті таких тенденцій постає актуальна задача проведення порівняльного аналізу продуктивності та ефективності використання обчислювальних ресурсів у середовищах, основаних на технологіях віртуалізації та контейнеризації. Глибоке дослідження дозволяє визначити найбільш доцільні підходи для виконання різного типу обчислювальних задач та сервісів, а також розробити практичні рекомендації для побудови оптимальних інфраструктур.

Постановка проблеми. Для об'єднання різних обчислювальних ресурсів та перенесення їх функцій із фізичної машини на віртуальну використовують технологію віртуалізації. Завдяки цьому можна використовувати окремі ресурси, такі як оперативна пам’ять (RAM), процесор (CPU) та інші складові, для створення декількох незалежних віртуальних машин на одному сервері. Це дозволяє оптимізувати витрати та зменшити потребу у придбанні додаткового обладнання та підвищити ефективність роботи всієї IT-інфраструктури компанії. Віртуалізація платформ передбачає створення програмних систем на базі вже існуючих апаратно-програмних комплексів, які можуть як залежати від цих комплексів, так і працювати незалежно від них. Система, що забезпечує апаратні ресурси та програмне середовище, називається хостовою, а ті системи, що виконуються в її середовищі — гостьовими [3]. Для стабільної роботи гостьових систем на основі хостової платформи необхідно, щоб апаратне та програмне забезпечення хоста вирізнялося високою надійністю і надавало потрібний набір інтерфейсів для доступу до своїх ресурсів. Існує кілька типів віртуалізації платформ, кожен із яких має власний підхід до реалізації концепції віртуалізації. Види такої віртуалізації залежать від рівня симуляції апаратного забезпечення.

Аналіз останніх досліджень та публікацій. Технологію віртуалізації платформ здебільшого розділяють на кілька основних типів.

У рамках повної емуляції віртуальна машина реплікує все апаратне забезпечення, забезпечуючи функціонування «гостьової» операційної системи (ОС) без її модифікації. Такий метод дозволяє емулювати різні апаратні архітектури. Основна проблема пов’язана зі значним уповільненням роботи емульованої апаратної частини, що негативно позначається на швидкодії ОС і ускладнює комфортне користування нею.

Часткова емуляція, або нативна віртуалізація, передбачає віртуалізацію лише мінімально необхідної частини апаратного забезпечення для забезпечення ізольованого запуску віртуальної машини. Завдяки цьому підходу можуть працювати «гостьові» ОС, які розраховані на ту саму архітектуру, що й хост. Це дозволяє одночасно запускати кілька екземплярів «гостьових» систем. Такий вид віртуалізації значно підвищує продуктивність у порівнянні з повною емуляцією і є дуже актуальним на сьогоднішній день. Головними перевагами такого підходу є відносна простота впровадження, універсальність та висока надійність рішення, оскільки всі функції управління виконує «хостова» ОС. Однак недоліком цього методу залишається залежність віртуальних машин від архітектури апаратної платформи. Часткова віртуалізація, а також «віртуалізація адресного простору» («address space virtualization»). При такому підході, віртуальна машина симулює кілька екземплярів апаратного оточення (але не всього), зокрема, простору адрес. Такий вид віртуалізації дозволяє спільно використовувати ресурси та ізолювати процеси, але не дозволяє розділяти екземпляри «гостьових» ОС. При застосуванні паравіртуалізації немає необхідності симулювати апаратне забезпечення, однак, замість цього (або на додаток до цього), використовується спеціальний програмний інтерфейс (API, application programming interface) для взаємодії з «гостьовою» операційною системою. Це вимагає підтримки з боку виробників операційних систем, які слабо вірять у можливості такого методу віртуалізації, у зв'язку з чим цей вид віртуалізації розвивається дуже слабо, хоча і існують порівняння, що показують, що швидкодія паравіртуалізації вища.

Віртуалізація рівня ОС (віртуалізація систем). Даний вид віртуалізації застосовується з метою створення декількох захищених

віртуальних серверів на одному фізичному. «Гостьова» система, в даному випадку, розділяє використання одного ядра операційної системи хостингу з іншими «гостьовими» операційними системами. Даний тип віртуалізації застосовується при організації систем хостингу, коли в рамках одного екземпляра ядра потрібно підтримувати кілька віртуальних серверів клієнтів. Віртуалізація на рівні додатків відрізняється від інших видів віртуалізації. У той час як попередні способи передбачають створення віртуальних середовищ або машин для ізоляції додатків, у цьому випадку сам додаток розміщується в спеціальному контейнері, який містить усі необхідні компоненти для його функціонування: файли реєстру, конфігураційні файли, а також користувацькі та системні об'єкти. У результаті отримується додаток, який не потребує стандартної інсталяції на відповідній платформі. При перенесенні такого додатка на інший пристрій і запуску, створене для нього віртуальне середовище усуває потенційні конфлікти з операційною системою чи іншими програмами. Віртуалізація ресурсів, на відміну від віртуалізації платформ, має більш широке і розпливчасте значення і являє собою масу різних підходів, спрямованих на підвищення зручності користування системами в цілому. Оскільки віртуалізація платформ орієнтована більше на конкретні цілі, то її застосування може підвищити ефективність використання комп'ютерного обладнання.

Мета статті. Основною метою статті є підвищення ефективності використання обчислювальних ресурсів на основі аналізу, порівняння та експериментального оцінювання продуктивності віртуальних машин і контейнерів.

Виклад основного матеріалу дослідження. На сьогоднішній день можна виділити наступні варіанти використання продуктів віртуалізації:

1. Консолідація серверів. На даний момент додатки, що працюють на серверах в IT-інфраструктурі (IT – Information Technology) компаній або інших установах, створюють невелике навантаження на апаратні ресурси серверів. Віртуалізація дозволяє розмістити все на одному фізичному сервері, збільшивши його завантаження, підвищивши тим самим ефективність використання апаратури, що дозволяє істотно заощадити на обладнанні, обслуговуванні та електроенергії.

2. Розробка і тестування додатків. Безліч продуктів віртуалізації дозволяють запускати кілька різних операційних систем одночасно, дозволяючи тим самим розробникам і тестувальникам програмного забезпечення тестувати їх додатки на різних платформах і конфігураціях.

3. Використання в бізнесі. Цей варіант використання віртуальних машин є найбільш широким і творчим. До нього відноситься все, що може знадобитися при повсякденному використанні IT ресурсів в бізнесі. На основі віртуальних машин можна легко створювати резервні копії робочих станцій і серверів, що забезпечують мінімальний час відновлення після збоїв тощо.

4. Використання віртуальних робочих столів. VDI (Virtual Desktop Infrastructure) — це технологія, що дозволяє створювати віртуальну IT-інфраструктуру і розгортати повноцінні робочі місця на базі одного сервера, на якому працює безліч віртуальних машин [4].

Віртуальні персональні комп'ютери, з якими працюють користувачі, нічим не відрізняються від звичайних персональних комп'ютерів. Однак технологія VDI дозволяє тримати всю необхідну для роботи інформацію під рукою в будь-якому місці, де є доступ до Інтернету (вдома, у ділових поїздках, на відпочинку). Віддалена робота стає дуже простою і комфортною. Вся інформація з віртуального комп'ютера зберігається в спеціальних дата-центрах. Це забезпечує її гарантований захист від втрати (всі дані проходять процедуру автоматичного резервного копіювання), випадкового або навмисного видалення, а також від доступу до неї сторонніх осіб. Обслуговування віртуальних робочих місць (зокрема встановлення програмного забезпечення, оновлення додатків тощо) здійснюється централізовано, що дозволяє мінімізувати витрати часу, а також значно знизити навантаження на системних адміністраторів. У процесі віртуалізації IT-інфраструктури створюється віртуальна інфраструктура – комплекс систем на основі віртуальних машин, що забезпечують функціонування всієї IT-інфраструктури, що володіє багатьма новими можливостями при збереженні існуючої схеми діяльності IT-ресурсів. Розробники різних платформ віртуалізації можуть надати інформацію про успішні проєкти з впровадження віртуальної інфраструктури у великих банках, промислових компаніях, лікарнях, освітніх установах.

Віртуалізація програмного забезпечення дозволяє створювати середовище з визначеною апаратною конфігурацією, що забезпечує можливість запуску гостьової операційної системи. Простіше кажучи, ви можете працювати з операційною системою Android на платформі Microsoft Windows, використовуючи ресурси вашого пристрою. Ця технологія поділяється на два основні типи: віртуалізацію додатків та віртуалізацію процесів та сервісів. Віртуалізація на рівні операційної системи зазнала значних змін упродовж останніх кількох років. Так, реалізація віртуалізації на апаратному рівні є доволі складною, оскільки потребує апаратної емуляції. Як альтернатива, віртуалізація на рівні ОС використовує контейнерні механізми, що базуються на функціях ядра, таких як CGroups, простори імен, CHROOT та інші. Ці механізми забезпечують створення ізольованих примірників операційного середовища, відомих як контейнери, на базі хостової машини [5].

Завдяки використанню контейнерного механізму контейнери розділяють ресурси ядра хост-системи, залишаючись незалежними від повноцінної операційної системи. Віртуалізація серверів метою серверної віртуалізації є створення програмної архітектури, яка дозволяє запускати кілька операційних систем на одному фізичному сервері. При цьому програмне забезпечення кожного віртуального сервера функціонує незалежно від апаратного забезпечення. Процес віртуалізації здійснюється за допомогою гіпервізора і не поступається за ефективністю традиційним рішенням. Гіпервізор створює віртуальні ресурси й функціонує безпосередньо на апаратному забезпеченні, маючи систему головного адміністратора як обов’язкову складову. Основна функція гіпервізора полягає у створенні ізольованих середовищ для роботи кожної віртуальної машини та контролі доступу віртуальних машин і гостьових операційних систем до фізичних апаратних ресурсів комп'ютера. За допомогою віртуалізації систем зберігання даних (СЗД) компанії можуть забезпечувати рівномірний розподіл навантаження з урахуванням важливих параметрів, таких як ефективність, доступність, захист та відмовостійкість. Впровадження такої технології дає можливість: спростити процес міграції корпоративних даних; забезпечити надійне зберігання інформації; оптимально використовувати серверний простір; знизити витрати на розширення IT-інфраструктури. Основна ідея віртуалізації систем зберігання полягає у створенні абстрактного проміжного шару між фізичними пристроями (HDD, SSD, NVMe, SAN, NAS) та логічними дисками, які використовуються операційними системами, гіпервізорами або програмними додатками. Це дозволяє приховувати фізичну структуру даних, забезпечувати їх логічне групування та спрощувати централізоване управління.

У сучасних системах використовуються два основні підходи до віртуалізації зберігання:

а) Віртуалізація на рівні хост-сервера. Цей підхід передбачає реалізацію шару абстракції безпосередньо в гіпервізорі або операційній системі. До прикладів можна віднести віртуальні диски у форматах VDI (VirtualBox), VMDK (VMware), VHDX (Hyper-V). Хост-сервер виконує функції зі створення, розширення, клонування та управління знімками цих віртуальних дисків. Серед переваг такого підходу виділяють простоту реалізації та відсутність потреби в додатковому обладнанні. Проте недоліками є обмежені можливості масштабування і залежність від конкретного програмного забезпечення [6].

б) Віртуалізація на рівні мережевого середовища (SAN/NAS) [7]. Тут абстрагування зберігання реалізується за допомогою спеціалізованих контролерів або програмних рішень, які об'єднують кілька фізичних масивів у єдині віртуальні пули сховищ. Сервери отримують доступ до цих логічних томів через мережу. До основних переваг підходу належать висока масштабованість, покращена доступність даних та розширені можливості балансування навантаження. Водночас основними недоліками виступають складність у адмініструванні та значна вартість обладнання.

Віртуалізація сховищ пропонує низку важливих переваг: гнучке та оперативне управління ресурсами, зокрема швидке створення, розширення та перенесення віртуальних томів; ефективне використання дискового простору завдяки тонкому розподілу ресурсів (thin provisioning) та інтелектуальному управлінню даними; збільшення рівня надійності завдяки реплікації і можливості оперативного відновлення з резервних копій; спрощене централізоване адміністрування, яке полегшує управління великими обчислювальними інфраструктурами; розширювана масштабованість, що дозволяє додавати нові накопичувачі без необхідності зупинки роботи системи.

Однак, попри значні переваги, ця технологія має певні недоліки: можливе збільшення часу доступу до даних через додатковий шар абстракції;

зростання рівня складності системи, що потребує високої кваліфікації ІТ-фахівців; ризик перевантаження фізичних носіїв при неправильному налаштуванні тонкого розподілу ресурсів; залежність від програмного забезпечення, збої якого можуть спричинити проблеми з доступом до всього логічного сховища.

У системах віртуалізації (VirtualBox, VMware ESXi, KVM, Hyper-V) віртуалізація систем зберігання є критично важливою [8]. Вона забезпечує можливість швидкого розгортання віртуальних машин, створення шаблонів, резервування та відкату стану. У середовищах контейнеризації (Docker, Kubernetes) вона використовується для роботи з персистентними томами, що гарантує збереження даних незалежно від життєвого циклу контейнера.

У популярних системах віртуалізації серверів, є свої відмінні риси і переваги. Серед них:

Система віртуалізації Microsoft Hyper-V являє собою професійне рішення в галузі апаратної віртуалізації, яке дозволяє створювати декілька незалежних програмних середовищ на основі одного фізичного сервера. Використовуючи платформу Hyper-V, організації отримують можливість взаємодіяти з апаратним забезпеченням за допомогою інструментарію віртуалізації. Основними напрямами застосування системи Microsoft Hyper-V є: реалізація рішень для віртуалізації робочих місць співробітників підприємств; впровадження і підтримка серверної інфраструктури в центрах обробки даних (ЦОД); надання розробникам безпечного середовища для тестування нових програмних функцій перед їх впровадженням у виробничі системи.

Система віртуалізації VMware ESXi є апаратним гіпервізором, призначеним для реалізації віртуалізації в центрах обробки даних, а також для створення хмарних ІТ-інфраструктур [9]. Ця платформа підтримує роботу з різноманітними операційними системами та застосовується головним чином для: консолідації роботи серверів; підвищення ефективності функціонування ІТ-систем і конкурентоспроможності організації; оптимізації процесів адміністрування ІТ-інфраструктури через використання гіпервізора; зменшення капітальних і операційних витрат; забезпечення безперервності бізнес-процесів; інтеграції механізмів аварійного відновлення для гарантування стійкості системи до збоїв. Таким чином, обидві системи забезпечують ключові інструменти для підвищення ефективності управління ІТ-інфраструктурою, проте відрізняються своїми можливостями і сферою застосування.

Розробка програмного забезпечення не відбувається ізольовано: на всіх етапах, від контролю до тестування, необхідні операційна система та відповідний набір інструментів. Ці інструменти можуть суттєво відрізнятися залежно від завдань. Наприклад, те, чим користується тестувальник, навряд чи знадобиться спеціалісту з development та operations (DevOps) – об’єднанню команди розробки та ІТ-експлуатації. Крім того, кожен інструмент має свої вимоги до конкретних версій бібліотек, фреймворків та інших залежностей. Також потрібно враховувати, що фінальний продукт, наприклад, розроблений для Linux, може бути призначений для використання під Windows, Android або інші платформи. У результаті утворюється досить різнобічне програмне середовище, яке щоразу складно налаштовувати заново, особливо якщо потрібна конкретна версія інструментів. Це може викликати додаткові труднощі та забирати час. На щастя, таку проблему можна вирішити за допомогою створення образів віртуальних машин або контейнерів.

Віртуалізація серверів є технологією, яка дозволяє на одному фізичному сервері розгортати кілька віртуальних серверів. Це забезпечує більш раціональне використання обчислювальних ресурсів фізичного сервера, що сприяє зниженню витрат і підвищенню ефективності роботи. Зниження витрат: віртуалізація серверів дозволяє скоротити кількість фізичних серверів, що призводить до економії на електроенергії, охолодженні та обладнанні.

Підвищення продуктивності: віртуалізація серверів дозволяє більш ефективно використовувати ресурси фізичних серверів.

Поліпшення масштабованості: віртуалізація серверів дозволяє легко додавати нові сервери в ІТ-інфраструктуру. Підвищення безпеки: віртуалізація серверів дозволяє ізолювати один від одного різні середовища і додатки. Підвищення мобільності: віртуалізація

серверів дозволяє легко переміщати віртуальні сервери між фізичними серверами.

3) Віртуалізація графічних процесорів(GPU).

Зауважимо, що віртуалізація передусім забезпечує ізоляцію інструкцій, які виконуються центральним процесором, одна від одної. До того ж, цей процес охоплює й емуляцію решти периферійних пристроїв, серед яких і відеокарти. Проте, на відміну від оперативної пам'яті чи мережевих портів, відеокарти також здатні виконувати інструкції. Тому для досягнення повноцінної віртуалізації важливо забезпечити їхню ізоляцію, що є доволі нетривіальним завданням. Навіть суто фізичний доступ до відеокарти без належної віртуалізації може обернутися складним викликом. При цьому в домашніх умовах можливо надати кожній віртуальній машині власну відеокарту, але в професійному середовищі цього буде недостатньо.

У сучасному контексті хмарних сервісів ключовими моментами стають масштабованість і ефективність. Високопродуктивна серверна відеокарта або навіть цілий масив таких карт можуть знадобитися декільком клієнтам одночасно із різними рівнями навантаження в різний час. В умовах таких вимог особливо актуальним стає питання максимальної окупності дорогого обладнання на кшталт DGX A1000, графічної системи на основі штучного інтелекту для прискорення обчислень [10]. Тому простої такого обладнання є неприйнятними, а його обчислювальні ресурси потрібно розподіляти між віртуальними машинами, причому динамічно.

Контейнеризація — це підхід до розгортання програмного забезпечення разом із усіма необхідними компонентами, такими як код, бібліотеки, фреймворки та інші залежності, У традиційній розробці програмного забезпечення процес запуску відбувається в конкретному обчислювальному середовищі.

Технологія контейнеризації використовується вже досить тривалий час та стала набирати значну популярність з приходом хмарних технологій. Недоліки традиційних віртуальних машин стали каталізатором зростання популярності контейнерів [11]. Постачальники інструментів для роботи з контейнерами, наприклад, Docker, значною мірою спростили технологію контейнеризації, що сприяло впровадженню даної технології в широкі маси [12]. При спробі перенести програму, наприклад, із локального комп'ютера до хмарної інфраструктури чи з однієї операційної системи на іншу (з Linux на Windows), можуть з'явитися помилки або несумісності. Контейнеризація вирішує цю проблему, оскільки все необхідне для роботи програми поміщається в контейнер, який є ізольованим, незалежним від операційної системи хосту та здатним працювати на будь-якій платформі без додаткових налаштувань. Такий підхід можна порівняти з процесом переїзду. Ви складаєте свої речі по коробках: посуд в одну, одяг в іншу, книги в третю. Кожна категорія ізольована та структурована. Завдяки цьому коробки легко транспортувати з місця на місце, не перепаковуючи кожен предмет окремо. Аналогічно, контейнери дозволяють легко переносити програми між середовищами без необхідності їх перекомпіляції чи налаштування. Всі компоненти, необхідні для роботи додатка (код, середовище запуску, системні інструменти, бібліотеки та налаштування), упаковуються в один образ і можуть бути використані повторно в рамках поточного завдання або для будь-яких інших. Контейнер незалежний від ресурсів і архітектури хоста. Він створює ізольоване середовище для додатка, не використовуючи CPU, RAM або сховище хостової ОС. Всі процеси відбуваються всередині.

На фізичному сервері операційна система може підтримувати велику кількість контейнерів. Усі вони повністю ізольовані один від одного, але при цьому спільно використовують основні компоненти операційної системи, такі як ядро. Цікавою є специфіка роботи додатків у таких контейнерах. Кожен додаток може залежати від певного набору компонентів, необхідних для його функціонування. Ці залежні компоненти доступні виключно всередині того контейнера, в якому розташований сам додаток. Це означає, що у разі будь-яких збоїв програми чи її залежностей у першому контейнері, робота додатка та файлів у другому контейнері залишиться незмінною. У традиційному середовищі ситуація, наприклад, із видаленням реєстру додатком 1 призвела б до порушення роботи як програми 1, так і програми 2. Однак використання контейнерів гарантує повну ізоляцію додатків 1 і 2, тому всі дії програми 1, зокрема зміна чи видалення реєстру, жодним чином не впливатимуть на роботу програми 2.

Усі виконувані файли та залежні компоненти знаходяться всередині контейнера, що забезпечує повну переносимість програми,

яка працює в такому середовищі. Це дозволяє розгортати контейнер на будь-якому вузлі з встановленим диспетчером контейнерів, після чого його можна запускати та використовувати без потреби у додатковій зміні чи налаштуванні. Наприклад, розробник може розпочати розгортання програми у контейнері Hyper-V на Windows 11 (рис. 1).

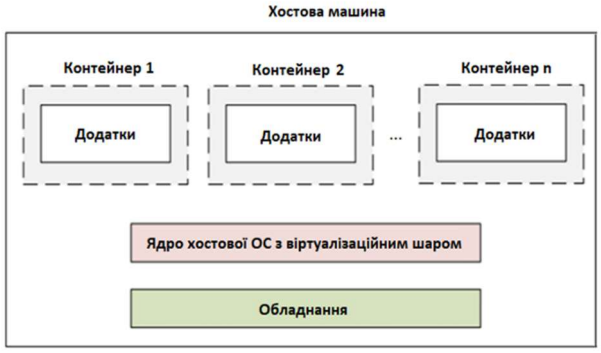

Рис. 1. Схематичне представлення контейнерної віртуалізації

Після завершення підготовки до робочого середовища таку програму можна розгорнути на Windows Server 2025 [13], включно з Nano Server, а також у публічній, приватній чи гібридній хмарі.

Контейнери складаються з кількох рівнів. Першим і базовим рівнем є образ операційної системи, на якому будуються всі наступні шари. Цей образ зберігається у репозиторії, що дозволяє використовувати його за необхідності, посилаючись на відповідне посилання. Наступний рівень, який може бути іноді й останнім, представляє платформу для додатків. Він може бути універсальним для всіх додатків. Наприклад, якщо базовий рівень побудовано на ядрі Windows Server, то платформою додатків можуть слугувати .NET Framework та Internet Information Services (IIS). Другий рівень також можна зберегти як окремий образ, який описує залежність від базового — ядра Windows Server. На рівні додатків розміщується сам додаток, що базується на платформі для додатків і водночас залежить від базового рівня системи.

На базовий рівень та рівень платформи додатків можуть посилатися всі створювані контейнери додатків у будь-який момент. Кожен рівень доступний лише для читання, за винятком верхнього рівня розгорнутого образу. Наприклад, якщо контейнер розгортається з використанням лише ядра Windows Server, то цей рівень стає верхнім у контейнері. Для збереження всіх операцій запису та змін, які виконуються під час роботи, створюється пісочниця. Згодом ці зміни можна зберегти як окремий образ для подальшого використання. Такий самий підхід застосовується і під час розгортання образу з рівнем платформи додатків: цьому рівню присвоюється власна пісочниця, а після виконання програми пісочницю можна зберегти у вигляді образу для наступного використання.

Загальний принцип полягає в наступному: під час розгортання контейнера на вузлі система самостійно визначає наявність базового рівня. Якщо цього рівня немає, вузол завантажує його зі сховища образів. Аналогічний процес повторюється для рівня платформи додатків, після чого створюється необхідний контейнер додатка. У разі потреби створити інший контейнер із такими ж залежностями достатньо виконати команду для створення нового контейнера додатків. Новий контейнер буде готовий до роботи майже миттєво, оскільки всі необхідні компоненти вже є в наявності. Існуючі рішення контейнерної віртуалізації представлені в таблиці 1.

Таблиця 1

Існуючі рішення контейнерної віртуалізації

Тип	Ядро	Коротка характеристика
OpenVZ	Linux	Запуск на одному фізичному сервері безліч ізольованих копій ОС, які називають віртуальні приватні сервери або віртуальні середовища.
Linux upstream containers (LXC)	Linux	Схожа з технологією з OpenVZ, але вирізняється однією суттєвою особливістю — усе необхідне для роботи з LXC уже інтегровано в стандартне ядро Linux.
Windows Server і контейнери Hyper-V	Windows	Використовує номенклатуру контейнерів Windows для опису розділів, створених поверх рівня віртуалізації операційної системи
Docker	Linux	Відкрита платформа для розробки, доставки та запуску додатків. Використовує клієнт-серверну архітектуру
CoreOS	Linux	Спільний проєкт із відкритого впорядкування, який отримав назву Відкрита ініціатива контейнерів (OCI) під Linux

Змінювати розмір межі ресурсів, таких як процесор чи оперативна пам'ять, у контейнерах можна динамічно, без необхідності перезавантаження, завдяки функції ядра CGROUP. Проте у випадку з віртуальними машинами спочатку потрібно зупинити їх, виконати переналаштування, а тоді перезапустити. У підсумку процес масштабування для віртуальних машин є менш простим. Принцип функціонування технологій контейнерів, доступних в Windows Server 2025, показано на рис. 2.

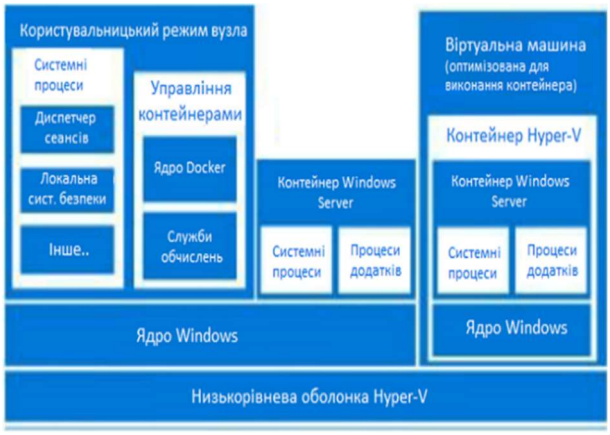

Рис. 2. Контейнери Windows Server 2025 і контейнери Hyper-V на одному і тому ж фізичному сервері

Серед недоліків технології віртуалізації та контейнерізації слід відмітити: складність управління при збільшенні кількості контейнерів, що працюють з додатком, перевантаження ресурсів при контейнери включенні значно більше компонентів і ресурсів, ніж це потрібно для коректної роботи програми, сумісність із Linux, що створює певні незручності під час роботи в середовищі Windows, роблячи його менш інтуїтивним і складнішим у щоденному використанні в порівнянні з Linux, короткий життєвий цикл.

Подальший розвиток використання контейнеризації наступний:

1) Міграція в хмару – це програмна стратегія, яка включає інкапсуляцію застарілих додатків у контейнери та розгортання їх у середовищі хмарних обчислень. Організації можуть модернізувати свої додатки без переписування всього програмного коду.

2) Впровадження мікросервісної архітектури. Організації, які прагнуть створювати хмарні додатки з мікросервісами, потребують технології контейнеризації. Архітектура мікросервісів – це підхід до розробки програмного забезпечення, який використовує кілька взаємозалежних програмних компонентів для надання функціонального додатка. Кожен мікросервіс має унікальну і специфічну функцію. Сучасний хмарний додаток складається з декількох мікросервісів. Наприклад, додаток потокового відео може мати мікросервіси для обробки даних, відстеження користувачів, виставлення рахунків і персоналізації. Контейнеризація надає програмний інструмент для пакування мікросервісів у вигляді програм, що розгортаються на різних платформах.

3) Пристрої Інтернету речей (IoT). Пристрої IoT містять обмежені обчислювальні ресурси, що робить оновлення програмного забезпечення вручну складним процесом. Контейнеризація дозволяє розробникам легко розгортати та оновлювати додатки на пристроях IoT.

На перший погляд, віртуалізація та контейнеризація здаються схожими за своїми характеристиками. Проте контейнери й віртуальні машини — це зовсім різні технології. Тому твердження про те, що вони функціонують однаково, є некоректним. Віртуальні машини та контейнери мають різну природу і призначення, а також спрямовані на вирішення різних задач у сфері віртуалізації. Ці відмінності наочно продемонстровані на рис. 3.

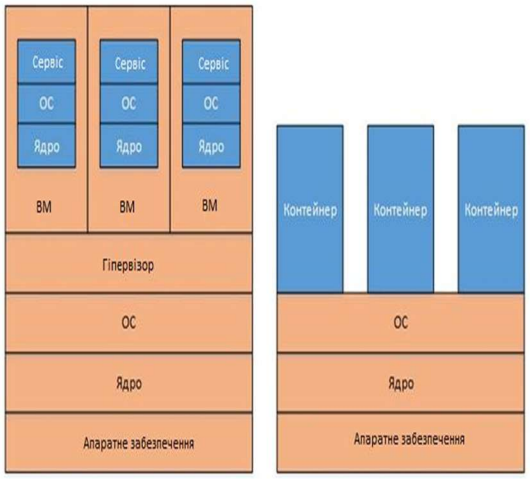

Рис. 3. Відмінність між віртуальною машиною (ВМ) і віртуальним контейнером

Гіпервізор потрібен для кожної ВМ використовується власна гостьова ОС. Дозволяє створювати неоднорідні обчислювальні середовища на одному комп'ютері. Через власну ОС ВМ може займати кілька ГБ, а запуск ОС і всіх додатків займає якийсь час.

Віртуальні машини функціонують на глибшому рівні, ніж контейнери. Вони забезпечують віртуалізацію програмного забезпечення, включаючи процесор, системи введення-виведення, пам'ять тощо. Віртуальні машини повністю містять операційну систему та виконують її незалежно від оточення основної системи, на якій запущений їхній образ. Завдяки цьому вони забезпечують повну ізоляцію процесів. Проте через те, що віртуальні машини включають весь функціональний набір операційної системи, вони доволі ресурсоємні. Така особливість може бути як перевагою, так і недоліком. Головна перевага полягає у високому рівні ізоляції процесів, які виконуються всередині віртуальної машини. З іншого боку, недоліком є обмеженість кількості віртуальних машин, що можуть одночасно працювати на одному сервері, оскільки вони потребують значних ресурсів для своєї роботи [11] .

Розмір віртуальної машини значною мірою впливає на тривалість її запуску та зупинки. Процес запуску передбачає завантаження операційної системи, через що загальний час цього етапу зазвичай є досить тривалим.

На відміну від віртуальних машин, контейнери використовують спільні частини ядра як для гостьової, так і для хостової операційної системи. Вони не оперують концепцією окремої гостьової операційної системи. Завдяки технології контейнеризації забезпечується ізольоване середовище виконання безпосередньо на базі хостової операційної системи. Такий підхід має як переваги, так і певні недоліки. Основною перевагою контейнерів є їх легкість і висока швидкість роботи. Через те, що контейнери спільно використовують ресурси хостової операційної системи, їх споживання ресурсів є мінімальним. Це дозволяє запускати велику кількість невеликих контейнерів на одній машині, що значно важче реалізувати за допомогою віртуальних машин. Проте існують і певні обмеження. Через використання однієї хостової операційної системи неможливо, наприклад, налаштувати брандмауер (iptables) всередині окремого контейнера. Однак процеси в межах кожного контейнера залишаються повністю ізольованими та незалежними від процесів інших контейнерів, які працюють на тій самій хостовій системі.

Легка структура контейнерів відкриває широкий спектр можливостей для їх автоматизованого створення, розгортання, завантаження та копіювання. Завдяки простоті виконання кількох команд або використанню REST API, завдання зі збору, доставки й запуску контейнера стає значно ефективнішим варіантом при розробці систем.

Контейнери мають значну перевагу порівняно з віртуальними машинами, проте останні також мають свої сильні сторони.

Для практичної реалізації віртуального середовища здійснено встановлення програмного забезпечення Oracle VirtualBox, яке використовується як гіпервізор для створення та керування віртуальними машинами [14]. Вибір даного програмного продукту обумовлений його безкоштовністю, кросплатформенністю, підтримкою широкого спектра гостьових операційних систем, а також наявністю розширених засобів конфігурації, що є необхідними для реалізації поставлених у роботі завдань.

Для забезпечення коректності аналізу обрано наступну конфігурацію:

– віртуальна машина: 2 CPU, 8192 MB RAM, ОС Ubuntu 24.04;

– хост-система: ОС Ubuntu 24.04 з встановленим Docker;

– програмне забезпечення: Docker 24.x, Nginx (офіційний образ).

Процес встановлення VirtualBox здійснювався на хост-операційній системі з урахуванням вимог до апаратної віртуалізації. Перед інсталяцією перевірено підтримку технологій Intel VT-x або AMD-V у налаштуваннях BIOS/UEFI, що є необхідною умовою для коректної роботи віртуальних машин. Після цього завантажено дистрибутив актуальної версії VirtualBox з офіційного джерела та виконано стандартну процедуру встановлення. Файли віртуальної машини було розміщено в окремому каталозі на диску хост-системи.

Результати дослідження наведені в таблиці 2.

Проведений аналіз дозволив виявити принципові архітектурні відмінності між класичною віртуалізацією та контейнеризацією. Віртуальна машина, що функціонує у середовищі VirtualBox, забезпечує повну ізоляцію гостьової ОС від хост-системи за рахунок використання гіпервізора та віртуалізованого апаратного забезпечення. Такий підхід мінімізує ризики впливу процесів гостьової системи на хост та інші віртуальні середовища. Контейнеризація забезпечує оптимальний баланс між рівнем ізоляції та

ефективністю використання обчислювальних ресурсів.

Таблиця 2

Результати досліджень

Критерій порівняння	Контейнер Docker (Nginx)	Віртуальна машина (Ubuntu + Nginx)
Стан простою	~3.3 міб	~1.2 гіб
Час розгортання	30с	600 с
Кількість кроків розгортання	3	12
Під навантаженням (ad, wrk)	~3.4 міб	~1.0-1.1 гіб
CPU-навантаження	низьке	100% на одному ядрі (wrk)
Кількість процесів	122	122
Налаштування ОС	Використання готового образу	Повна інсталяція та конфігурація
Мережа	Автоматичний проброс портів	Потрібна ручна конфігурація
Ресурси	Легке масштабування, спільні ресурси	Виділення RAM, CPU, диску
Супровід	Оновлення образу, CI/CD інтеграція	Оновлення ОС, драйвери
Швидкість Масштабування	Секунди	Хвилини-години
Тип ізоляції	Ізоляція процесів на рівні ОС	Повна ізоляція на рівні ОС та апаратури
Ядро операційної системи	Спільне ядро з хост-системою	Власне віртуалізоване ядро
Рівень безпеки	Середній	Високий
Ізоляція процесів	Забезпечується Linux namespaces	Повна
Доступ до апаратних ресурсів	Через механізми обмеження ресурсів	Через гіпервізор
Потенційні ризики	Вищі порівняно з VM	Мінімальні
Швидкість розгортання	Висока	Низька
Типові сценарії використання	Мікросхеми, веб-сервіси	Критичні системи, тестування ОС

Висновки. Досліджено сучасні підходи до віртуалізації обчислювальних середовищ, зокрема класичну віртуалізацію на основі віртуальних машин та контейнеризацію як більш сучасну та легковагову альтернативу, що зумовлена широким впровадженням технологій віртуалізації у сфері інформаційних технологій, хмарних обчислень, DevOps-практик та розгортання масштабованих інформаційних систем. Проаналізовано архітектуру віртуальних машин, роль гіпервізора, механізми ізоляції ресурсів, а також принципи роботи контейнерів, які використовують спільне ядро операційної системи. Окрему увагу приділено питанням безпеки, продуктивності та ефективності використання апаратних ресурсів у кожному з підходів. Практично реалізоване розгортання віртуального середовища з використанням програмного забезпечення VirtualBox з ОС Ubuntu, налаштовано апаратні ресурси, мережеву взаємодію, встановлено доповнення Guest Additions та перевірено коректність роботи гостьової ОС. Отримано повноцінне ізольоване середовище, яке повністю імітує роботу окремого фізичного комп'ютера. Розгортання контейнерного середовища з використанням платформи Docker продемонструвала простоту та швидкість розгортання контейнерів у порівнянні з віртуальними машинами. Віртуальні машини є доцільними у випадках, коли необхідна максимальна ізоляція та незалежність середовищ, тоді як контейнери є оптимальним рішенням для швидкого розгортання, масштабування та ефективного використання обчислювальних ресурсів.

Література

1. Що таке контейнерізація та в чому її переваги. URL: https://gigacloud.ua/articles/shho-take-kontejneryzacziya-ta-v-chomu-yiyi-perevagy/ (дата звернення 23.02.2026).
2. Що таке віртуалізація та які переваги вона надає. URL: https://gigacloud.ua/articles/shho-take-virtualizacziya-ta-yaki-perevagy-vona-nadaye/ (дата звернення 23.02.2026).
3. Що таке хост система. URL: https://mean.kultura.cx.ua/zhittya-v-kiievi/shho-take-khost-sistema.html (дата звернення 23.02.2026).
4. Віртуалізація vs контейнеризація – порівняння відмінностей. URL: https://alexhost.com/uk/faq/virtualization-vs-containerization-comparing-differences/ (дата звернення 23.02.2026).
5. Верес Л.А. Аналіз архітектурних особливостей віртуалізації та переваги віртуальних контейнерів. Збірник матеріалів XII Міжнародної науково-технічної конференції

«Перспективи телекомунікацій», м. Київ, 2018. С. 116–118.
6. Віртуалізація серверів. URL: https://bit-dp.com/it-support/virtualizacziya-serveriv/ (дата звернення 20.02.2026).
7. Струбицький Р.П. Методи та алгоритми побудови хмаркових сховищ даних на основі розподілених телекомунікаційних систем. URL: https://lpnu.ua/sites/default/files/2020/dissertation/1551/disstrubyckyy2.pdf (дата звернення 20.02.2026).
8. Віртуалізація серверів: Порівняння VMware, Hyper-V та KVM. URL: https://itez.com.ua/blog/vmware-hyper-v-kvm-detailed-comparison.html (дата звернення 23.02.2026).
9. Сизов А. Гіпервізори VMware VMware ESXi та його можливості. URL: https://serversolutions.com.ua/blogs/news/гіпервізори-vmware-vmware-esxi-та-його-можливості? (дата звернення 23.02.2026).
10. Карпусь В. NVIDIA DGX A100 — система для прискорення ШІ-обчислень з 8 прискорювачами Tesla A100. URL: https://itc.ua/news/nvidia-dgx-a100-sistema-dlya-uskoreniya-ii-vychislenij-s-8-uskoritelyami-tesla-a100-proizvoditelnostyu-5-petaflops-i-czenoj-200-tys/ (дата звернення 23.02.2026).
11. Кардашук В.С., Тарасенко Д.Ю., Барбарук В.М. Віртуалізаційні технології та контейнеризація як засоби організації обчислювальних середовищ. ІТ-Ідея – 2025: збірник науково-практичних праць XI Міжнародного Форуму. – Київ: вид-во Східноукр. ун-ту ім. В. Даля, 2025. С. 106-109.
12. Docker. The #1 containerization software for developers and teams. URL: https://www.docker.com/products/docker-desktop/ (дата звернення 23.02.2026).
13. Windows Server 2025: Нова ера гібридної хмари, безпеки та високої продуктивності. URL: https://techexpert.ua/windows-server-2025-a-new-era-of-hybrid-cloud-security-and-high-performance/ (дата звернення 23.02.2026).
14. Powerful open source virtualization. URL: https://www.virtualbox.org (дата звернення 23.02.2026).

References

1. Shcho take konteinerizatsiia ta v chomu yii perevahy. URL: https://gigacloud.ua/articles/shho-take-kontejneryzacziya-ta-v-chomu-yiyi-perevagy/ (data zvernennia 23.02.2026).
2. Shcho take virtualizatsiia ta yaki perevahy vona nadaie. URL: https://gigacloud.ua/articles/shho-take-virtualizacziya-ta-yaki-perevagy-vona-nadaye/ (data zvernennia 23.02.2026).
3. Shcho take khost systema. URL: https://mean.kultura.cx.ua/zhittya-v-kiievi/shho-take-khost-sistema.html (data zvernennia 23.02.2026).
4. Virtualizatsiia vs konteineryzatsiia – porivniannia vidminnostei. URL: https://alexhost.com/uk/faq/virtualization-vs-containerization-comparing-differences/ (data zvernennia 23.02.2026).
5. Veres L.A. Analiz arkhitekturnykh osoblyvostei virtualizatsii ta perevahy virtualnykh konteineriv. Zbirnyk materialiv KhII Mizhnarodnoi naukovo-tekhnichnoi konferentsii «Perspektyvy telekomunikatsii», m. Kyiv, 2018. S. 116–118.
6. Virtualizatsiia serveriv. URL: https://bit-dp.com/it-support/virtualizacziya-serveriv/ (data zvernennia 20.02.2026).
7. Strubytskyi R.P. Metody ta alhorytmy pobudovy khmarkovykh skhovyshch danykh na osnovi rozpodilenykh telekomunikatsiinykh system. URL: https://lpnu.ua/sites/default/files/2020/dissertation/1551/disstrubyckyy2.pdf (data zvernennia 20.02.2026).
8. Virtualizatsiia serveriv: Porivniannia VMware, Hyper-V ta KVM. URL: https://itez.com.ua/blog/vmware-hyper-v-kvm-detailed-comparison.html (data zvernennia 23.02.2026).
9. Syzov A. Hipervizory VMware VMware ESXi ta yoho mozhlyvosti. URL: https://serversolutions.com.ua/blogs/news/hipervizory-vmware-vmware-esxi-ta-yoho-mozhlyvosti? (data zvernennia 23.02.2026).
10. Karpus V. NVIDIA DGX A100 — systema dlia pryskorennia ShI-obchyslen z 8 pryskoriuvachamy Tesla A100. URL: https://itc.ua/news/nvidia-dgx-a100-sistema-dlya-uskoreniya-ii-vychislenij-s-8-uskoritelyami-tesla-a100-proizvoditelnostyu-5-petaflops-i-czenoj-200-tys/ (data zvernennia 23.02.2026).
11. Kardashuk V.S., Tarasenko D.Yu., Barbaruk V.M. Virtualizatsiini tekhnolohii ta konteineryzatsiia yak zasoby orhanizatsii obchysliuvalnykh seredovyshch. IT-Ideia – 2025: zbirnyk naukovo-praktychnykh prats KhI Mizhnarodnoho Forumu. – Kyiv: vyd-vo Skhidnoukr. un-tu im. V. Dalia, 2025. S. 106-109.
12. Docker. The #1 containerization software for developers and teams. URL: https://www.docker.com/products/docker-desktop/ (data zvernennia 23.02.2026).
13. Windows Server 2025: Nova era hibrydnoi khmary, bezpeky ta vysokoi produktyvnosti. URL: https://techexpert.ua/windows-server-2025-a-new-era-of-hybrid-cloud-security-and-high-performance/ (data zvernennia 23.02.2026).
14. Powerful open source virtualization. URL: https://www.virtualbox.org (data zvernennia 23.02.2026).

Ryazantsev O.I., Kardashuk V.S., Bortnyk K.Y., Barbaruk V.M. Virtualization and containerization as means of organizing computing environments

The article explores the use of virtualization and containerization for organizing computing processes. A thorough analysis of virtualization and containerization technologies, their principles of operation, and scope of application is conducted. It is established that virtualization allows you to create isolated computing environments with a high level of security and flexibility, but is accompanied by additional resource costs. Containerization provides ease of deployment, efficient use of hardware resources, and speed, which makes it especially relevant for modern DevOps processes and cloud infrastructures. The advantages and disadvantages of each approach are considered, as well as scenarios for their practical application. An overview of the VirtualBox and Docker platforms, which are used to implement virtualized and container environments, respectively, is provided. A comparative analysis of the functionality, advantages, and limitations of each platform is conducted. It is determined that VirtualBox provides flexible configuration of virtual machines and support for various guest Oses, while Docker is focused on lightweight, fast, and scalable containers. Based on the analysis, an experimental environment for testing performance was formed, which made it possible to objectively assess the effectiveness of both technologies in real conditions. A practical study and comparative analysis of two approaches to virtualization of computing environments were performed: classical virtualization based on virtual machines (VirtualBox) and containerization using the Docker platform. The effectiveness of each approach was assessed in terms of performance, use of computing resources, isolation level, security, and ease of deployment. The results of the practical study confirmed that virtual machines and containers have different areas of effective application. Virtual machines are advisable to use in cases where the priority is maximum isolation and versatility of the environment, while containerization is the optimal solution for rapid deployment, efficient use of resources, and building scalable modern information systems. The results obtained can be used to select a virtualization technology depending on the requirements of a specific task in practical application.

Keywords: *virtualization, containerization, scalability, operating system, server, application.*

Рязанцев О.І. – д.т.н., професор, завідувач кафедри комп'ютерних наук та інженерії Східноукраїнського національного університету імені Володимира Даля, a_ryazantsev@snu.edu.ua

Кардашук В.С. – к.т.н., доцент кафедри комп'ютерних наук та інженерії Східноукраїнського національного університету імені Володимира Даля, kardashuk1@gmail.com

Бортник К.Я. – к.т.н., доцент кафедри комп'ютерної інженерії та безпеки Луцького національного технічного університету, katerina.bortnyk@gmail.com

Барбарук В.М. – к.т.н., доцент кафедри інформаційних систем і технологій національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», barbaruk.viktor@gmail.com

Стаття подана 19.12.2025.