TELECOMMUNICATIONS MEAS
ЗАСОБИ ТЕЛЕКОМУНІКАЦІЙ

УДК 621.396.004 **DOI:10.30837/rt.2022.1.208.06**

Л.О. ТОКАР, канд. техн. наук

ОСОБЛИВОСТІ ПОБУДОВИ ВІРТУАЛЬНИХ АТС

Вступ

Важливим рішенням для будь-якої корпоративної компанії або офісної структури є вибір якісного та дешевого телефонного зв'язку. На ринку пропонуються готові рішення щодо вибору хмарних АТС, які вимагають додаткові витрати.

Хмарна АТС дає можливість використання багатоканальних номерів, пропонуючи інтелектуальну переадресацію дзвінків, гнучку аналітику, можливість запису й зберігання інформації на надійному зовнішньому сервері. Крім того, надає безліч інших функцій: інтерактивне голосове меню, повідомлення про пропущені виклики, голосова пошта, електронний факс й т.і. Але основна її перевага - це висока якість телефонії, що не досяжне за допомогою підключення стільникового й аналогового зв'язку.

Хмарна АТС дозволяє економити і отримувати телефонію високої якості. Основними вигодами виступають розширення функціоналу, зручність у використанні при істотному скороченні постійних матеріальних витрат. До її плюсів відноситься високий рівень технологічності. Хмарна АТС забезпечує компанії більш високоефективним зв'язком, комфортним управлінням, якісним голосовим каналом, а головне - є недорогою в порівнянні з аналоговою АТС.

Виходячи з того, що впровадження класичної АТС займає тривалий час, вибір рішення на основі IP-телефонії очевидний. На відміну від апаратної міні-АТС, масштабування хмарної АТС виконується в міру необхідності, простим додаванням віртуалізованих ресурсів. Крім того, всі технічні проблеми залишаються на стороні провайдера, а компанія отримує вже готову послугу.

Використання віртуальної АТС не як додаткового сервісу, а як окремої конфігурації виділеного сервера, поєднуючи всі переваги хмарних технологій і систем віртуалізації, дасть можливість отримання гнучкості і повної доступності системи для налаштувань, це і визначає актуальність даної публікації. В роботі проведено аналіз технологій віртуалізації в залежності від складності виконання та області застосування, показано їх переваги та недоліки. Розглянуто конфігурацію налаштування середовища віртуалізації на прикладі Asterisk. Проведено дослідження для порівняння контейнерної віртуалізації з гіпервізорною для визначення економії оперативної пам'яті у хост-системі.

Основна частина

Одним із способів перекладу інфраструктури комп'ютера на динамічний рівень є віртуалізація. Технології віртуалізації набирають більшу популярність, що є слідством росту обчислювальних потужностей комп'ютерів. Віртуалізація являє собою програмну технологію, що дає можливість одночасно виконувати декілька ОС і додатків на одному сервері. Модель віртуалізації показано на рис. 1 [1].

За допомогою віртуалізації створюється віртуальний образ операційної системи комп'ютера, обчислювальної мережі або накопичувального пристрою. Віртуальна машина співіснує «всередині» комп'ютера зі звичайною операційною системою. В результаті розвитку технологій віртуалізації, з'являються багатоядерні процесори, зростає пропускна здатність інтерфейсів комп'ютерів, а також ємність та швидкодія систем зберігання даних [2].

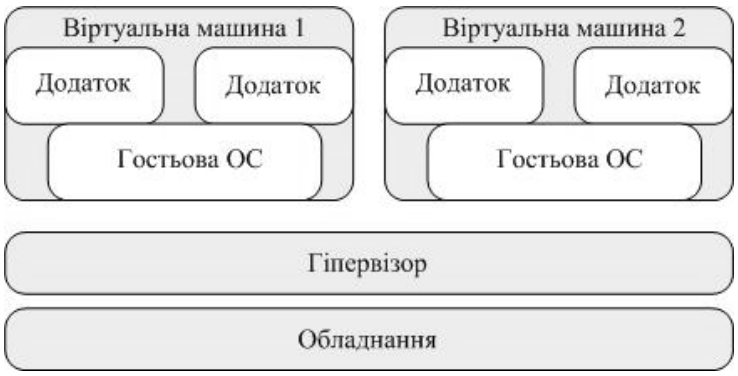

Рис. 1. Модель віртуалізації

Новий аспект віртуалізації було названо командною або бінарною віртуализацією. В цьому випадку віртуальні команди переводяться (транслюються) на фізичні команди основного обладнання. Зазвичай це відбувається динамічно. Якщо відбувається розгалуження, то новий сегмент коду забирається і перекладається [3].

Для організації віртуалізації існує декілька способів, за допомогою яких досягаються однакові результати через різні рівні абстракції. У кожного способу є свої переваги і недоліки, але головне те, що кожен з них знаходить своє місце в залежності від області застосування.

Один з найскладніших методів віртуалізації забезпечується емуляцією апаратних засобів. У цьому методі VM (virtual machine) апаратних засобів для емуляції обладнання створюється на хост-системі [4]. Але головною проблемою при емуляції апаратних засобів визнають суттєве уповільнення при виконанні програм в такому середовищі. Оскільки кожна команда повинна моделюватися на основних апаратних засобах, при цьому уповільнення в 100 разів при емуляції є звичайною справою. Незважаючи на це, такий метод емуляції має суттєві переваги. Наприклад, управління операційною системою для PowerPC на системі з ARM процесором. Також можна управляти численними віртуальними машинами, кожна з яких буде моделювати інший процесор [5]. Модель емуяції обладнання показано на рис. 2.

Рис. 2. Емуляція обладнання

Повна (апаратна) віртуалізація, або «рідна» віртуалізація, є іншим способом віртуалізації. Ця модель використовує менеджер віртуальних машин (гіпервізор), який здійснює зв'язок між гостьовою операційною системою і апаратними засобами системи [6]. Усередині гіпервізора повинно бути встановлено й налаштовано певний захист, так як основні апаратні засоби не належать ОС, а розділяються гіпервізором. При побудові великих корпоративних систем, як правило, використовується саме апаратна віртуалізація, що показано на рис. 3.

Рис. 3. Апаратна віртуалізація

При цьому великі вендори, такі як VMware, IBM й Microsoft, розробляють свої платформи віртуалізації на базі технологій апаратної віртуалізації Intel VT, AMD-V.

Паравіртуалізація - це інший популярний спосіб, який має деяку схожість з повною віртуалізацією. Цей метод використовує гіпервізор для поділу доступу до основних апаратних засобів, але об'єднує код, що стосується віртуалізації, в безпосередньо операційну систему [7]. Паравіртуалізація розділяє процес з гостьовою операційною системою. Паравіртуалізація вимагає, щоб гостьова ОС була змінена для гіпервізора, і це є недоліком методу. Однак, паравіртуалізація пропонує високу продуктивність, майже як у реальної системи. При цьому, як і при повній віртуалізації, одночасно можуть підтримуватися різні операційні системи. Але певним недоліком паравіртуалізації можна вважати обмежену кількість підтримуваних ОС. Оскільки є необхідність вносити зміни в код ядра ОС, що не завжди представляється можливим через закритість деяких ОС.

Віртуалізація рівня операційної системи. Цей метод підтримує єдину операційну систему, що зображено на рис. 4. В найзагальнішому випадку - просто ізолює незалежні віртуальні сервери (контейнери) один від одного. Для поділу ресурсів одного сервера між контейнерами, дана віртуалізація вимагає внесення змін в ядро операційної системи (наприклад, як у випадку з OpenVZ) [8]. При цьому перевагою її є рідна продуктивність, без «накладних витрат» на віртуалізацію пристроїв.

Рис. 4. Віртуалізація рівня операційної системи

PROXMOX VE - система віртуалізації з відкритим вихідним кодом. Управляється за допомогою веб-інтерфейсу або командного рядка Linux. Може працювати як окрема віртуальна машина, а також об'єднуватися в кластер, використовуючи вбудовані можливості гіпервізора. Для гостьових ОС на базі Linux є готові шаблони віртуальних машин, що завантажуються з офіційного сайту. PROXMOX VE є одним з кращих безкоштовних рішень для віртуалізації з відкритим вихідним кодом [9].

В даний час існує безліч причин використання віртуалізації. Найважливішою причиною є, так звана, серверна консолідація. Простіше кажучи, можливість віртуалізувати безліч систем на окремому сервері. Це дає можливість організаціям заощадити на потужності, місці й адмініструванні через наявність меншої кількості серверів. При цьому важливим фактором є абстрагування від обладнання. Наприклад, сервера іноді виходять з ладу. При цьому є можливість перерозподілити навантаження на обладнання. Відсутність прив'язки, до якогось «заліза» істотно полегшує життя IT-відділу й знижує ризик простою підприємства.

Інша причина використання віртуалізації полягає в тому, що буває спочатку важко визначити навантаження на сервер. При цьому процедура віртуалізації підтримує так звану живу міграцію (live migration). Жива міграція дозволяє ОС, яка переміщається на новий сервер, і її додаткам збалансувати навантаження на доступному обладнанні.

Використовуючи можливості сучасних ПК, можна легко розгорнути будь-який віртуальний сервер навіть на домашньому комп'ютері, а потім легко перенести його на інше обладнання. Віртуалізація також важлива для рішень, необхідних розробникам. Наприклад, віртуалізація дозволяє керувати кількома операційними системами, і якщо одна з них зазнає краху через помилки, то гіпервізор й інші операційні системи продовжують працювати. Це дозволяє зробити налагодження ядра системи подібно додаткам, що призначені для користувача.

Одним з методів віртуалізації є контейнеризація. Контейнеризація - це легка віртуалізація та ізоляція ресурсів на рівні операційної системи, яка дозволяє запускати додаток і необхідний йому мінімум системних бібліотек в повністю стандартизованому контейнері, що з'єднують з хостом або чим-небудь зовнішнім по відношенню до нього за допомогою певних інтерфейсів. Контейнер не залежить від ресурсів або архітектури хосту, на якому він працює [10].

Всі компоненти, що необхідні для запуску програми, упаковуються як один образ та можуть бути використані повторно. Додаток в контейнері працює в ізольованому середовищі і не використовує пам'ять, процесор або диск хостової операційної системи. На рис. 5 зображено графічне порівняння віртуалізації та контейнеризації [11].

Рис. 5. Графічне порівняння віртуалізації та контейнеризації

В цілому можна виділити наступні переваги використання віртуалізації:

1) Скорочення витрат на придбання й підтримку обладнання. У сучасних умовах практично в кожній компанії завжди знайдеться один або два сервера мають декілька ролей, наприклад, поштовий сервер, файловий сервер, сервер бази даних й т.і. Безумовно, на одній фізичній машині можна піднімати по кілька програмних комплексів (серверів), що виконують різні завдання. Але дуже часто бувають ситуації, коли встановлення нового ПО вимагає незалежної серверної одиниці. В такому випадку якраз і буде необхідною віртуальна машина з потрібною ОС. Сюди ж можна віднести випадки, коли в мережі необхідно мати кілька незалежних один від одного віртуальних серверів зі своїм набором служб і своїми характеристи-

ками, які повинні існувати як незалежні вузли мережі. Типовий приклад - це послуги хостингу.

2) Скорочення серверного парку. Перевага віртуалізації полягає в тому, що можна значно скоротити кількість фізичних ПК. В результаті менше часу і грошей витрачається на пошук, закупівлю й заміну обладнання. Поряд з цим скорочуються площі, що виділяються під зміст серверної бази.

3) Скорочення штату IT-співробітників. На обслуговування меншої кількості фізичних машин потрібно менше людей. З точки зору керівництва компанії, скорочення штату - це скорочення серйозною статті витрат підприємства.

4) Простота в обслуговуванні. Додати жорсткий диск або розширити існуючий, збільшити кількість оперативної пам'яті, все це займає певний час у разі з фізичним сервером. Відключення, від'єднання зі стійки, підключення нового обладнання, включення - в разі використання віртуалізації всі ці дії опускаються, і операція зводиться до кількох клацань миші або командам адміністратора.

5) Клонування й резервування. Ще одним плюсом віртуалізації є простота клонування віртуальних машин. Наприклад, компанія відкриває новий офіс. При цьому серверна інфраструктура центрального офісу стандартизована та являє собою кілька серверів з однаковими налаштуваннями. Розгортання такої інфраструктури зводиться до простого копіювання образів на сервер нового офісу, конфігурації мережевого обладнання і зміни налаштувань в прикладному ПО.

Використання віртуальної АТС дозволяє поєднувати переваги Інтернет-сервісів з простотою класичної телефонної станції. Віртуальні АТС називають хмарними, оскільки основна частина заліза функціонує на стороні сервіс-провайдера, а клієнт отримує послугу в чистому вигляді. Віртуальна АТС має значно більшу пропускну здатність і ніяк не лімітована по маршрутизації. Її можна налаштувати за різними сценаріями.

Основними рішеннями IP-телефонії є фірмові і відкриті IP-PBX, SIP-провайдери, а також віртуальні і хмарні АТС. На практиці найбільший відсоток займають відкриті АТС. Найбільш відомими безкоштовними програмними продуктами, що розповсюджуються в вихідному коді, сьогодні є: Asterisk, Yate, SipXecs, FreeSWITCH.

Для вибору адекватного рішення розглянемо деякі перспективні підходи до побудови хмарної АТС.

Asterisk являє собою повністю програмну АТС, що працює під управлінням операційної системи Linux та забезпечує підтримку практично всіх популярних протоколів IP-телефонії: SIP, H323, SCCP, ADSI. Крім стандартних і загальновідомих, Asterisk також має свій власний протокол - IAX.

Перевага Asterisk полягає в її розширеному функціоналі, в поєднанні з яким немає аналогів відповідністю стандартам. Знаючи переваги додатків Asterisk, його вибирають фундаментом для роботи віртуальної АТС. Таким чином, функціонал хмарної АТС якісно підкріплений новими сучасними можливостями [12].

Використовуючи систему Asterisk, віртуальна АТС дає можливість:

- розмістити на один SIP-логін кілька телефонних ліній;
- налаштувати голосове привітання за індивідуальними сценаріями;
- створити інтерактивне голосове меню для зв'язку з компетентним відділом;
- записувати і прослуховувати розмови абонентів з вашими співробітниками;
- ставити дзвінки в чергу і розподіляти їх між агентами для якісної обробки.

PROXMOX VE підтримує новітні функції віртуалізації пам'яті від виробників процесорів, для мінімізації завантаження процесора і досягнення високої пропускної здатності. PROXMOX VE успадковує потужні функції управління пам'яттю від Linux. Пам'ять віртуальної машини зберігається так само, як пам'ять будь-якого іншого Linux-процесу, і може замінюватися, копіюватися великими сторінками для підвищення продуктивності, узагальнюватися або зберігатися в файлі на диску.

Для збереження даних PROXMOX VE може використовувати будь-який носій, підтримуваний Linux, для зберігання образів віртуальних машин, в тому числі локальні диски з інтерфейсами IDE, SCSI і SATA, Network Attached Storage (NAS), включаючи NFS і SAMBA / CIFS, або SAN з підтримкою iSCSI і Fibre Channel. Для поліпшення пропускної спроможності системи зберігання даних і резервування може використовуватися багато-введення / виведення.

Знову ж таки, оскільки система входить до складу ядра Linux, може використовуватися перевірена і надійна інфраструктура зберігання даних з підтримкою всіх провідних виробників; його набір функцій зберігання перевірений на багатьох виробничих установках. Система підтримує динамічну міграцію, що продемонстровано на рис 6. Цим забезпечується можливість переміщення працюючих віртуальних машин між фізичними вузлами без переривання обслуговування.

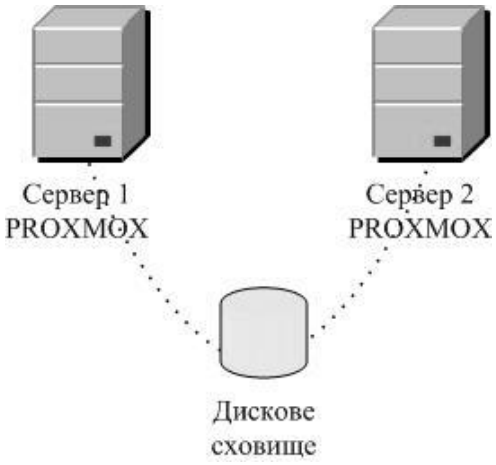

Рис. 6. Динамічна міграція віртуальних машин

Динамічна міграція є прозорою для користувачів: віртуальна машина залишається включеною, мережеві з'єднання - активними, і призначені для користувача додатки продовжують працювати, в той час як віртуальна машина переміщається на новий фізичний сервер.

Велику частину обладнання, що використовується в мережах можна замінити сервісом, які надають оператори та провайдери або орендою виділеного серверу.

Наприклад, орендована віртуальна АТС з параметрами: 10 SIP-абонентів, 10 робочих місць, 1 GSM шлюз коштує в середньому 700 грн/місяць [13]. Процедура з віртуальним сервером зводиться до зміни налаштувань в панелі управління і, при необхідності, перезапуску сервера. Тобто можна почати з мінімальних необхідних вимог і в міру потреби збільшувати використовувані ресурси, масштабуючи роботу. Якщо ресурси більш не потрібні, від них так само легко можна відмовитися.

Фізичний сервер володіє певними параметрами і для їх зміни потрібен час і кошти. Наприклад, для встановлення додаткового модуля оперативної пам'яті потрібно втручання в роботу сервера, чим порушується безперебійність роботи. При оренді сервера використовується 100% його ресурсів - ніяких сусідів по серверному обладнанні не буде.

Таким чином, переваги виділеного серверу наступні: не потрібно витрачати кошти на придбання серверів та обладнання; підключення за лічені хвилини; доступність сервера; мінімальні кошти на оренду віртуального виділеного сервера та багатоканального номеру; розширення можливостей серверу: гнучкості і пропускної здатності; вибір мінімальних затримок для сервера в залежності від розташування абонентів.

Оренда виділеного сервера надає можливість самостійно вибрати конфігурацію і програмне середовище, в якій буде розвиватися АТС. Виділений сервер з мінімальними параметрами коштує в середньому 1000 грн/місяць [14].

Віртуальна машина працює незалежно від інших, розташованих з нею на одній хост-машині. Це означає, що збій в роботі одного сервера не впливає на функціонування сусідів.

Розглянемо конфігурацію налаштування середовища віртуалізації на прикладі Asterisk. Накладні витрати на створення контейнерів LXC для АТС Asterisk, дуже невеликі, і ніщо не заважає запускати їх сотнями або тисячами. Це неможливо при використанні повноцінної віртуалізації. При створенні контейнера LXC для Asterisk, треба визначити основні параметри: ім'я контейнера, кількість ядер процесора, обсяг оперативної пам'яті, образ системи для Asterisk, розмір диска, адресу мережі.

Всі ці параметри, за допомогою POST запиту передаються в відповідний процес, що продемонстровано на рис 7. Процес запускає утиліту, яка створює контейнер, або видає помилку, якщо якісь параметри вказані невірно.

Рис. 7. Схема створення контейнера для Asterisk

Для розробки схеми IP-телефонії використовувалася стандартна схема побудови подібних телефонних систем, в якій застосовано існуючу IP-мережу, що виключає закупівлю додаткового мережевого обладнання. Крім того, для уникнення проблем з масштабуванням проєкту, необхідний перехід з тризначного номерного плану на чотиризначний план. Схему реалізації середовища віртуалізації PROXMOX для віртуальної АТС Asterisk зображено на рис. 8.

Далі, необхідно визначиться з уніфікацією обладнання, яке буде здійснювати роботу всієї мережі. Сервер з потужностями, достатніми для організації підключення всіх користувачів, в разі переходу на оптоволоконний зв'язок. У такому випадку пропаде необхідність встановлювати сервера для віддалених ділянок. Системні вимоги для сервера в такому випадку будуть наступними: процесор Xeon E3-1220, пам’ять 1Gb DDR3 RAM, мережа зі швидкістю 200 Мбіт/с.

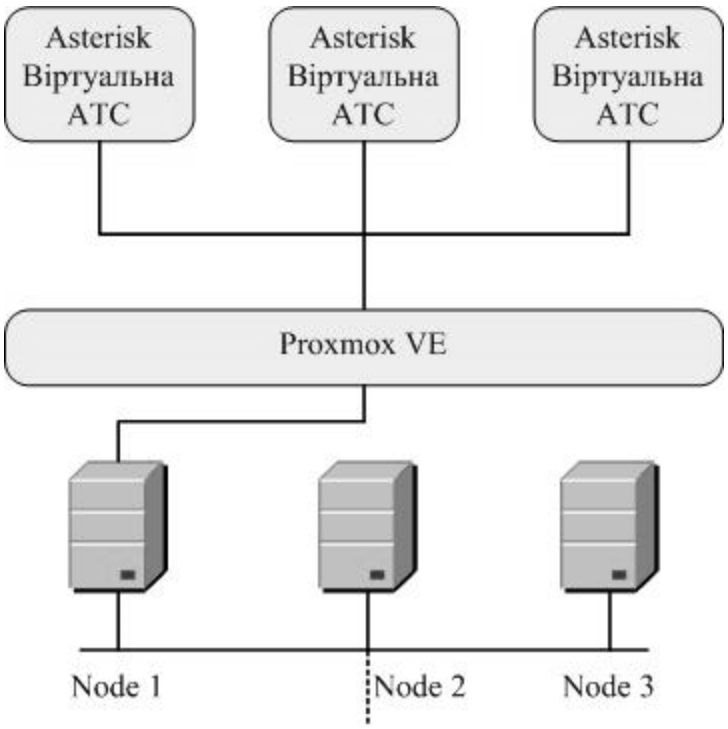

Рис. 8. Модель налаштування середовища віртуалізації

У роботі проведено дослідження, яке з'ясує, наскільки можна провести економію оперативної пам'яті у хост-системі у разі використання контейнерної віртуалізації у порівнянні з гіпервізорною. У якості навантаження на сервер, на якому будуть працювати віртуальні машини, взято процедуру встановлення операційної системи. Ця процедура задіє відразу всі види ресурсів й супроводжується високим процесорним навантаженням, великим споживанням ОП, інтенсивною роботою із зовнішніми пристроями та, що дуже важливо, активною роботою з накопичувачем на магнітних дисках (як правило, це віртуальні накопичувачі, як використовуються файли хост-системи). При цьому підготовлено сервери, встановлено VirtualBox для сервера, на якому використовується гіпервізорная віртуалізація, а для сервера з контейнерною віртуалізацію - програмне забезпечення Docker [15].

Розглянемо дослідження з однією віртуальною машиною та одним контейнером з однією віртуальною машиною. Дослідження проведено у два етапи:

- на першому етапі проаналізовано навантаження на оперативну пам'ять з одним контейнером (табл. 1) та з однією віртуальною машиною (табл. 2).

Таблиця 1

Час, с	0	100	150	200	400	450	700	750
Навантаження на ОП, %	0,2	0,4	1,4	1,5	1,5	3,7	3,7	0,2

Таблиця 2

Час, с	0	100	150	200	400	450	550
Навантаження на ОП, %	0,5	1,7	1,7	1,7	4,3	4,3	0,5

- на другому етапі проаналізовано навантаження на CPU з одним контейнером (табл. 3) та з однією віртуальною машиною (табл. 4).

Таблиця 3

Час, с	0	100	150	200	400	450	700	750
Навантаження на CPU, %	0	1,5	6,5	6,5	8,5	8,5	8	0,5

Таблиця 4

Час, с	0	100	150	200	400	450	550
Навантаження на CPU, %	0,5	7,5	8,2	8,5	8,5	8,5	0,5

Аналіз показує, що звичайна віртуальна машина працює швидше, ніж аналогічна у контейнері, споживаючи при цьому більше обчислювальних ресурсів. Контейнер показав менше споживання ресурсів, але більш тривалу роботу. Однак помітно, що після фази розпакування файлів з образу навантаження зростає приблизно в 2 рази. Це характеризує процес встановлення операційної системи і є основним піком навантаження на ОП. Таким чином, побудова (складання) системи з двійкових файлів вимагає істотно більше ресурсів, ніж підготовка цих файлів.

Що стосується навантаження на CPU, то помітно, що віртуальна машина більш вимоглива, їй потрібно більше обчислювальної потужності. Але в масштабах сервера це не так значно, якщо аналізується одна віртуальна машина.

Розглянемо узагальнене дослідження при запуску десяти віртуальних машин на одному сервері та десяти контейнерів. У табл. 5 показано навантаження на ОП від часу. У табл. 6 показано навантаження на CPU від часу.

Таблиця 5

Час, с	0	250	2000	2500	4000	4500
Навантаження на ОП, %	0	15	15	37	37	1

Таблиця 6

Час, с	0	250	2000	2500	4000	4500
Навантаження на CPU, %	0	50	17	17	20	5

Зі збільшенням кількості одиниць віртуальних машин і контейнерів потреба в ресурсах зростає однаково, тобто залежність витрати ресурсів відносно один одного зберігається. Час виконання всіх операцій також зростає, але з різною інтенсивністю.

Таким чином, контейнерна віртуалізація є більш ефективною в плані витрати пам'яті, але втрачає у швидкодії через додаткові витрати на обчислення адрес. Контейнери, у свою чергу, навпаки, є віртуалізацією на рівні операційної системи. А це вже має на увазі, що є гостьова операційна система, яка використує те саме ядро, що й хостова ОС. Такий підхід дає контейнерам велику перевагу: вони можуть бути меншими і компактнішими за гіпервізорні гостьові середовища, оскільки у них з хостом набагатого більше загальних процесів, ніж у віртуальних машин.

Дослідження показали, що при використанні контейнерів та програмного забезпечення Docker при організації кластерних обчислень не тільки вирішено проблему ізоляції віртуальних машин і мереж одного віддаленого користувача від іншого, але й отримано виграш у швидкості обчислень й витратах ОП.

Висновки

Показано основні вигоди хмарних АТС при високому рівні технологічності та відносній дешевизні в порівнянні з класичними АТС.

На прикладі моделей віртуалізації проведено аналіз методів віртуалізації, які організовано через різні рівні абстракції. Доведено переваги використання віртуалізації в залежності від витрат на придбання й підтримку обладнання, зменшення кількості серверного обладнання, скорочення штату співробітників, простоти в обслуговуванні, резервуванні, а також від налаштування за різними сценаріями.

На прикладі безкоштовного програмного продукту Asterisk, перевага якого в розширеному функціоналі, розглянуто рішення для побудови хмарної АТС. Показано процес запуску створення контейнеру для Asterisk. Надано модель реалізації середовища віртуалізації PROXMOX для віртуальної АТС Asterisk.

Проведено дослідження для оцінки контейнерної та гіпервізорної віртуалізацій. Дослідження показало, що хоча споживання обчислювальних ресурсів збільшується практично однаково зі збільшенням кількості використовуваних контейнерів та віртуальних машин, контейнери споживають менше ресурсів. Тривалість виконання обчислень зростає приблизно однаково, і тут виграють контейнери, тому що відпрацьовують швидше звичайних віртуальних машин. Для проєктування кластерної системи цей критерій є дуже важливим. Оскільки використовувані всередині контейнерів віртуальні машини безпечно ізольовані, то це дає можливість працювати окремо і ніяк не перетинатися в процесі виконання завдань та процесів.

Список літератури:

1. А. Гаврилов Платформы виртуализации. Обзор [Электронный ресурс] // Режим доступа: http://dx.doi.org/10.1109/ICC.2010.5502484.
2. Романов О.І., Нестеренко М.М., Фесьоха Н.О. Аналіз сучасних технологій віртуалізації для побудови інформаційно-телекомунікаційних систем // Збірник наукових праць ВІТІ. 2019. № 1. С. 82-90.
3. Новый плакат - VMware [Электронный ресурс] // Cloud on AWS Logical Design Poster for Workload Mobility. Режим доступа: https://www.vmgu.ru/search/AWS.
4. Івченко Ю. М., Івченко В. Г., Гондар О. М. Впровадження технології віртуалізації для підвищення надійності та безпеки інформаційних систем на залізничному транспорті // Електромагнітна сумісність та безпека на залізничному транспорті. 2014. № 7. С. 82-86.
5. Чепцов В.Ю., Хорошилов А.В. Эмуляция ввода-вывода оборудования с отображением в ОЗУ внутри ядер операционных систем // Труды Института системного программирования РАН. 2018. № 30(3). С. 121-134.
6. Yang Ye etc. S2H: Hypervisor as a setter within Virtualized Network I/O for VM isolation on cloud platform // Computer Networks. 2021. V.201. P. 156-163.
7. Ekane Brice, Ngoc Tu Dinh, Teabe Boris, Hagimont Daniel, Noel De Palma Adaptive network device services in a virtualized environment // Future Generation Computer Systems. 2021. V. 127. P. 14-22.
8. Wessel Sascha, Huber Manuel, Stumpf Frederic, Eckert Claudia Improving mobile device security with operating system-level virtualization // Computers & Security. 2015. V. 52. P. 207-220.
9. PROXMOX VE ADMINISTRATION GUIDE RELEASE [Электронный ресурс] // Proxmox Server Solutions Gmbh. Режим доступа: https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf.
10. Gordeev A. V., Gorelik D. V. Comparative Testing of Container and Hypervisor Virtualizations // Information and Control Systems. 2018. no. 2. P. 60–66.
11. Евстратов В. В. Контейнеризация как современный способ виртуализации // Молодой ученый. 2020. № 49 (339). С. 7-9.
12. Jim Van Meggelen, Russell Bryant и Leif Madsen Asterisk: The Definitive Guide. O'Reilly Media, Inc., Gravenstein Highway North, 2013. - 311 p.
13. ТОП-7 сервисов виртуальных АТС для IP телефонии в Украине [Электронный ресурс] // Режим доступа: https://seoukraine.com.ua/.
14. Конфігуратор виділених серверів [Электронный ресурс] // Режим доступа: https://www.ukraine.com.ua/uk/dedicated/.
15. Jessie Frazelle Docker Containers on the Desktop [Электронный ресурс] // Режим доступа: https://blog.jessfraz.com/post/docker-containers-on-the-desktop/.

Надійшла до редколегії 15.01.2022

Відомості про авторів:

Токар Любов Олександрівна – канд. техн. наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри інфокомунікаційної інженерії ім. В.В. Поповського (ІКІ); Україна; e-mail: liubov.tokar@nure.ua; ORCID: https://orcid.org/0000-0002-7780-1928