КВАЛІФІКАЦІЙНА РОБОТА

Група МІПЗс-22

Бойчук М.Б.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Бойчука Михайла Богдановича

УДК 004.4

Оптимізація моделей та методів хмарної міграції даних в рішеннях на основі віртуальних машин

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації магістра

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____ Бойчук М.Б.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.т.н., доц. Демчина М.М.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «магістр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« 19 » лютого 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бойчуку Михайлу Богдановичу

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Оптимізація моделей та методів хмарної міграції даних в рішеннях на основі віртуальних машин

керівник роботи:

Демчина Микола Миколайович, кандидат технічних наук, доцент

затверджена наказом вищого навчального закладу від « 26 » червня 2023 року № 32/1 с

2. Термін подання студентом роботи 16.02.2024

3. Вихідні дані роботи: Формальні моделі, методи та алгоритми.

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Огляд та аналіз концепцій міграції даних.
2. Дослідження та опис моделей та засобів хмарної міграції даних.
3. Імплементація та оптимізація моделі міграційного сховища даних.
4. Представлення архітектури та процесу системи міграції даних.

5. Дата видачі завдання 29.06.2023

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз концепцій міграції хмарних даних	26.09.2023	Виконано
2.	Дослідження та опис моделей та засобів хмарної міграції даних	20.10.2023	Виконано
3.	Імплементація та оптимізація моделі міграційного сховища даних	15.11.2023	Виконано
4.	Представлення архітектури та робочого процесу системи віртуалізації та міграції даних	30.11.2023	Виконано
5.	Формування висновків	09.12.2023	Виконано
6.	Оформлення пояснювальної записки	22.12.2023	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	11.01.2024	Виконано

Студент _____ Бойчук М.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Демчина М.М.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
16	Система VMware vSphere vMotion	52	Порівняння продуктивності VM I0 за різних швидкостей міграції
23	Алгоритм живої міграції VM	54	Порівняння продуктивності VM I0 із різними розмірами образів віртуальних дисків
25	Основні етапи міграції віртуальної машини які виконуються в техніці попереднього копіювання	60	Структура знімків віртуальної машини та робочий процес для запитів на читання/запис

АНОТАЦІЯ

Кваліфікаційна робота присвячена оптимізації моделей та методів хмарної міграції даних в рішеннях на основі віртуальних машин шляхом застосування результатів цього дослідження, що можуть дати дорожню карту для даних міграції та можуть допомогти тим, хто приймає рішення щодо процесу міграції даних, досягти безпечної та продуктивної міграції до середовища хмарних обчислень.

В першому розділі виконано аналіз концепцій міграції даних та хмарних обчислень, описано предметну область хмарних обчислень. Наведені основні проблеми які виникають під час міграції сховищ та даних, досліджено сутність концепції міграції динамічних станів пам'яті на основі віртуальних машин, проведено опис процесу міграції сховища даних.

В другому розділі проведено дослідження моделей та засоби міграції даних на основі віртуальних машин. описано концепції міграції даних з врахування навантаження, розроблено архітектуру та алгоритм функціонування системи міграції з врахуванням робочого навантаження, проведена оцінка ефективності підходу міграції.

В третьому розділі виконана імплементація та оптимізація моделі міграційного сховища даних, показано процес міграції сховища даних на основі концепції образів системи, представлено архітектуру та робочий процес системи віртуалізації та міграції даних. Запропоновано підхід оптимізації міграції сховища віртуальної машини за допомогою методу послідовного вводу-виводу.

КЛЮЧОВІ СЛОВА: ХМАРНІ СЕРЕДОВИЩЕ, ВІРТУАЛЬНА МАШИНА, МІГРАЦІЯ ДАНИХ, ХМАРНІ ОБЧИСЛЕННЯ, ОБРАЗ СИСТЕМИ, ЖИВЕ СХОВИЩЕ ВІРТУАЛЬНОЇ МАШИНИ.

SUMMARY

The qualification work is devoted to optimizing models and methods of cloud data migration in virtual machine-based solutions by applying the results of this study, which can provide a road map for data migration and can help those who make decisions about the data migration process to achieve a secure and productive migration to the environment cloud computing.

The first chapter analyzes the concepts of data migration and cloud computing, describes the subject area of cloud computing. The main problems that arise during the migration of storage and data are presented, the essence of the concept of migration of dynamic memory states based on virtual machines is investigated, and the process of data storage migration is described.

In the second chapter, a study of models and means of data migration based on virtual machines is carried out. the concept of data migration taking into account the load is described, the architecture and algorithm of the migration system functioning taking into account the workload is developed, the effectiveness of the migration approach is evaluated.

In the third section, the implementation and optimization of the data migration storage model is performed, the data storage migration process based on the concept of system images is shown, the architecture and workflow of the virtualization and data migration system are presented. An approach to optimizing virtual machine storage migration using the sequential input-output method is proposed.

KEY WORDS: CLOUD ENVIRONMENT, VIRTUAL MACHINE, DATA MIGRATION, CLOUD COMPUTING, SYSTEM IMAGE, VIRTUAL MACHINE LIVE STORA

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VM - Virtual Machine

VM IO - Virtual Machine Input-Output

SLA - Service Level Agreement

VMM - Virtual Machine Monitors

DBT - Dirty Block Tracking

WAIO - Workload-Aware IO Outsourcing

NVRAM - Non-Volatile RAM

LBA - Logical Block Address

VMS - Virtualization Management Server

ВСТУП

Актуальність дослідження. Платформи хмарних обчислень, такі як Amazon Web Services, ATT, Intel, IBM і Google Cloud отримали критичну увагу при вирішенні потреб клієнтів, тобто обчислювальну потужність, зменшуючи витрати на інфраструктуру та технічне обслуговування та оптимальний розподіл ресурсів. Ці переваги підтверджуються широким спектром послуг, які надаються загалом, є відкритими та доступними для поступового впровадження та випуску, а також через калькуляцію на основі використання моделі.

Хмарні обчислення є суттєвою віхою в наданні IT-послуг із великими перспективами на ринку. Хмарні обчислення існуючих систем, як основа рішень, має величезний вплив на ефективність IT розробок.

Проте, основною проблемою під час впровадження хмарних платформ є міграція даних у хмарі. Хмарна структура пропонує різні способи та типи послуг. Значну частину таких викликів необхідно вирішувати в кожному рішенні. Значною проблемою для адаптації хмарних обчислень є відсутність визначеності в безпеці платформи. Постачальники хмарних послуг грунтовно підійшли до забезпечення та постійної перевірки життєздатності своїх методологій безпеки.

Під час вибору хмарного провайдера, безпека має бути обов’язковою вимогою, головним приорітетом. Клієнти повинні переконатися, що їхній постачальник може підтримувати їх вимоги щодо безпеки та відповідності технічним специфікаціям. Клієнти можуть пом’якшити ризики хмарних обчислень, розробивши комплексну оцінку ризиків перед вибором постачальника та запуском послуги. Цілісність даних є також проблемою міграції даних у хмару.

Якщо дані клієнта не конфіденційні, єдина проблема полягає в тому, щоб безперешкодно потрапити в хмару. Ця проблема – виклик, який можна вирішити, перемістивши свої дані за допомогою інструменту, який перевіряє цілісність даних.

Дослідження пов’язані із хмарною міграцією, дають широкий аналіз існуючих методів міграції та характеризують їх на всіх рівнях стратегії міграції.

Основна мета полягає в ознайомленні з можливостями та перевагами міграції даних у хмару, а також виділення потенційних викликів міграції даних. Аналіз хмарної міграції як прикладу переходу від клієнтських фреймворків до хмарних фреймворків є постійною темою досліджень в IT, яка вивчалася в літературі, за допомогою яких клієнти приймають рішення щодо стійкості хмарної міграції.

Такі дослідження представляють та затверджують показ моделей факторів впливу (наприклад, збої в IT клієнта та відносна зручність використання) і факторів інтеграції (наприклад, вартість навчання, вартість налаштування та заходи безпеки).

Проте, незалежно від наявності цих елементів, користувачі не можуть повністю мігрувати у хмару через такі проблеми, як зміна витрат і особисті оцінки. Така вразливість проявляється як перевага постачальників хмарних послуг щодо класифікації, цілісності і доступності даних. Ряд досліджень розглядають хмарну міграцію як «чорну скриньку», не звужуючи її основні задачі до ключових операцій, пов'язаних з міграційним циклом.

Мета і завдання дослідження. Метою магістерської роботи є оптимізація продуктивності вводу-виводу віртуальної машини та продуктивності міграції під час процесу міграції живого сховища віртуальної машини шляхом вирішення проблеми введення-виведення.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

- виконати аналіз концепцій міграції даних;

- дослідити концепцію міграції динамічних станів пам’яті на основі віртуальних машин;
- структурувати моделі та засоби міграції даних;
- представити архітектуру та алгоритм функціонування системи міграції з врахуванням робочого навантаження;
- виконати імплементацію та оптимізацію моделі міграційного сховища даних.

Об’єктом дослідження є самі моделі та методи хмарної міграції даних, які використовуються в рішеннях на основі віртуальних машин.

Предметом дослідження є оптимізація моделей та методів хмарної міграції даних в рішеннях, що ґрунтуються на віртуальних машинах.

Методи дослідження базуються на використанні методів алгебри логіки та теорії множин для моделювання сховищ та баз даних та методи оптимізації для оцінки маршрутів міграції.

Наукова новизна одержаних результатів полягає в розробці рекомендацій, щодо різних процедур та моделей хмарної міграції, оцінки та визначення вимог безпеки, вибору хмари провайдера, розрахунок вартості та здійснення важливих організаційних зміни.

Результати цього дослідження можуть дати дорожню карту для даних міграції та можуть допомогти тим, хто приймає рішення щодо процесу міграції даних, досягти безпечної та продуктивної міграції до середовища хмарних обчислень.

Практичне значення одержаних результатів полягає в тому, що шляхом порівняння та аналізу виконано представлення схеми міграції на основі знімків, що дозволяє підвищити ефективність міграції живого сховища віртуальної машини та усунути її вплив на продуктивність додатків користувачів на вихідному сервері шляхом ефективного використання наявних знімків віртуальної машини на резервних серверах.

Апробація результатів дослідження. Матеріали дослідження було

представлено у матеріалах І Всеукраїнської науково-практичної інтернет конференції “ІТ екосистема: цифровізація бізнес-процесів в умовах війни”, у тезах доповіді “Роль віртуальних машин в хмарних дата-центрах”.

Структура. Кількість розділів – 3. Загальний обсяг основної частини – 88 сторінок. Список використаних джерел містить – 51 позицій.

РОЗДІЛ 1. АНАЛІЗ КОНЦЕПЦІЙ МІГРАЦІЇ ДАНИХ ТА ХМАРНИХ ОБЧИСЛЕНЬ

1.1 Опис предметної області хмарних обчислень

Технологія хмарних обчислень революціонізує спосіб ведення бізнесу в IT-відділах підприємства [1]. Підприємства орендують IT-ресурси у хмарних постачальників на основі вимоги та оплати за використання, що забезпечує численні переваги, починаючи від економії коштів і закінчуючи вищим рівнем надійності, доступності та масштабованості.

Netflix, провідна світова компанія з відеопослуг, закрила всі свої центри обробки даних і запустила всі відеосервіси на хмарній платформі Amazon AWS [2]. В останній час п’ять найбільших постачальників хмарних обчислень, включаючи Amazon, Microsoft, IBM, Google і Salesforce, спостерігали різке зростання доходів від хмарних інфраструктурних послуг на 37 - 96%. Враховуючи, що загальний ринок становить лише 16 мільярдів доларів, хмарні обчислення як індустрія все ще вважаються початковими, оскільки 16 мільярдів доларів – це лише невелика частка від майже 4 трильйонів доларів, які витрачають компанії на IT у всьому світі [1].

Завдяки технології віртуалізації та консолідації ресурсів, а також моделі виставлення рахунків на основі використання хмарні обчислення можуть надавати доступ за вимогою до обчислювальних, даних і програмних засобів як послугу, яка не накладає жодних обмежень на кінцевого користувача. фізичне розташування та конфігурації системи [3]. Хмарні обчислення стали однією з найважливіших технологій, яка в найближчому майбутньому може кардинально змінити життя людей та IT-екосистеми.

Очевидний успіх і багатообіцяюча перспектива обчислювальної техніки частково пояснюється її базовою інфраструктурою обчислень і зберігання даних: віртуальною машиною (VM - Virtual Machine).

Як один із найпопулярніших продуктів на ринку хмарних обчислень, віртуальні машини широко розгортаються для запуску різних служб для клієнтів, таких як веб-служби, поштові служби, служби баз даних і служби друку [4]. Основні гіпервізори для віртуалізованого середовища відповідають за керування фізичними пристроями та надання всіх видів віртуальних пристроїв віртуальним машинам. У той же час гіпервізори мають гарантувати ізоляцію та справедливість між різними віртуальними машинами на одному спільному фізичному сервері, одночасно покращуючи та вдосконалюючи загальну продуктивність усіх віртуальних машин із доступними фізичними ресурсами [5].

Онлайн міграція віртуальної машини — це вбудований модуль у сучасні гіпервізори, який може переміщувати працюючу віртуальну машину з одного фізичного сервера на інший або в межах того самого кластера, або в різних центрах обробки даних у всьому світі. Основна мета миттєвої міграції віртуальних машин — задоволення зростаючих потреб у балансуванні навантаження та консолідації серверів, обслуговуванні та оновленні системи, мобільності та керованості віртуальних машин у хмарних центрах обробки даних. Цей процес передбачає переміщення всієї інформації про стан віртуальної машини, що переноситься, що включає синхронізацію станів ЦП, станів пам’яті, мережевих інтерфейсів цільової віртуальної машини між джерелом і місцем призначення міграції, а також віртуальних дискових образів і знімків віртуальної машини (для надійності) і відновлення, поки віртуальна машина все ще виконує своє робоче навантаження. У той же час цей процес є прозорим для додатків, запущених у віртуальній машині, що переносить, та інших запланованих віртуальних машинах на тому самому фізичному сервері. З появою та широким розгортанням інфраструктури

хмарних обчислень на основі віртуальних машин динамічна міграція віртуальних машин як важливий функціональний компонент гіпервізорів, таких як ESX, XEN, QEMU-KVM і HyperV, стає все більш важливою з кількох важливих причин.

По-перше, функція міграції віртуальної машини в гіпервізорах забезпечує швидку та прозору роботу перепланування робочого навантаження між нерівномірно використовуваними фізичними вузлами для підвищення енергоефективності та ефективного використання ресурсів комп'ютерного сервера. Як показало нещодавнє дослідження Gartner Group [6], 61% із 518 респондентів зараз займаються проектами консолідації серверів, тоді як 28% планують консолідацію серверів у найближчому майбутньому. Також відомий факт, що балансування навантаження між сотнями тисяч серверів викликає велике занепокоєння в центрах обробки даних [7, 5]. Завдяки підтримці міграції живого сховища віртуальної машини можна динамічно регулювати зіставлення між віртуальними машинами та фізичними серверами, що розміщують, щоб досягти кращого рівня балансування навантаження та енергоефективності під час виконання.

По-друге, через зростаючі вимоги до обслуговування та оновлення системи, такі як заміна дефектних компонентів, підвищення продуктивності системи та розширення ємності зберігання даних, сервери даних і підсистеми зберігання регулярно зазнають оновлення системи [8, 9]. Жива міграція віртуальної машини може перенести всі запущені віртуальні машини з серверів, які потрібно відремонтувати або оновити.

По-третє, кожна запущена віртуальна машина має власні вимоги до ресурсів під час виконання, наприклад, відбиток пам'яті, пропускна здатність мережі та пропускна здатність зберігання. Якщо фізичний сервер не може надати такі ресурси віртуальній машині, віртуальну машину буде переміщено на інший сервер, який має достатньо доступних ресурсів.

По-четверте, проблема, відома як блокування продавця, змушує клієнтів залежати від постачальників хмарних послуг і не дозволяє їм змінити постачальника без значних витрат на перехід. Гнучкий і портативний підхід до живої міграції віртуальної машини може зіграти важливу роль у вирішенні проблеми прив'язки постачальника.

Нарешті, гібридні хмарні обчислення, де віртуальні машини працюють як на приватних, так і на публічних хмарних сайтах і за запитом переміщуються туди й назад, стають найпопулярнішою інфраструктурою. До кінця 2022 року майже половина великих підприємств розгорне гібридну хмарну інфраструктуру в дата-центрах [10]. VM live migration має вирішальне значення для ширшого визнання та розгортання гібридних хмар.

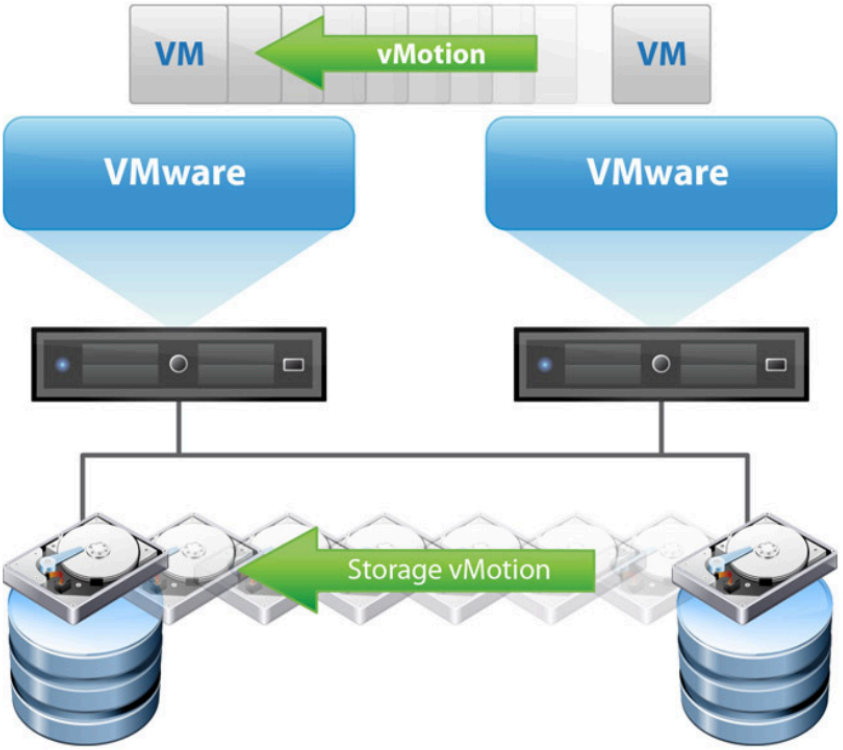


Рисунок 1.1 – Система VMware vSphere vMotion

У типовій хмарній інфраструктурі сховище даних може бути спільним або розподіленим залежно від того, чи зберігаються дані в централізованому середовищі, де всі сервери спільно використовують фізичне сховище, чи

розподіленому середовищі (без спільного використання), де кожен сервер має власний виділене сховище. У середовищі спільного зберігання оперативна міграція віртуальної машини передбачає лише синхронізацію станів ЦП, пам’яті та мережевих інтерфейсів цільової віртуальної машини між джерелом і одержувачем міграції. Образи віртуальних дисків віртуальної машини залишаються в спільному сховищі, доступному як з джерела, так і з місця призначення.

У зв’язку зі зростаючою тенденцією архітектур без спільного доступу в хмарних центрах обробки даних і потребою в живій міграції віртуальної машини між різними хмарами через глобальну мережу стає все більш важливим розглядати можливість оперативної міграції віртуальної машини в розподіленому сховищі або середовищі зберігання без спільного використання, де живе віртуальна машина. міграція сховища також має переносити стан образів віртуальних дисків і знімків віртуальної машини з джерела до місця призначення. Насправді міграція сховища VM стала невід’ємною частиною живої міграції у сучасних гіпервізорах [11, 12, 13].

Саме з цих причин дана робота зосереджена на live міграції віртуальної машини в середовищі розподіленого зберігання, а саме міграції живої пам’яті віртуальної машини. На рисунку 1.1 показаний приклад системи vMotion System від компанії VMware. Коли запущена віртуальна машина мігрує з одного вузла на інший, як стан віртуальної машини в пам’яті, так і образи віртуальних дисків мігрують із вихідного сервера на сервер призначення.

1.2. Основні проблеми які виникають під час міграції сховищ та даних

Враховуючи, що ємність віртуального сховища віртуальної машини, яка включає образи віртуальних дисків і знімки віртуальної машини, набагато більша, ніж інша інформація про стан віртуальної машини, наприклад стан

пам’яті, стан ЦП і стан мережі, надзвичайно важливо покращити продуктивність міграції живого сховища віртуальної машини. Якщо говорити конкретно, то схема міграції живого сховища віртуальної машини повинна мати такі властивості:

• **Короткий час міграції**: у сучасних центрах обробки даних віртуальні машини працюють 7 x 24 години, щоб обслуговувати клієнтів у всьому світі. Часове вікно для обслуговування та оновлення системи є коротким, тому дуже важливо швидко перенести віртуальну машину. Однак розмір образу сховища віртуальної машини зазвичай становить від кількох до десятків ГБ, і для завершення міграції живого сховища віртуальної машини можуть знадобитися хвилини або навіть години, що, ймовірно, обмежить можливості керування системою в хмарних центрах обробки даних. Наприклад, в одному випадку використання Aliyun [14], найбільшого постачальника хмарних послуг Alibaba в Китаї кожна віртуальна машина використовує близько 40 ГБ пам’яті, а кожен сервер містить у середньому 25 віртуальних машин. Припустімо, що кожна віртуальна машина рівномірно розподіляє пропускну здатність мережі 10 Гбіт/с, підключену до одного фізичного сервера. З точки зору мережі, їй все одно потрібно близько 13 хвилин для передачі даних, необхідних для однієї віртуальної машини в реальному часі. Крім того, ці образи віртуальної машини спільно використовують той самий ресурс зберігання, тому кожна віртуальна машина може отримати лише частину загальної пропускної здатності сховища на фізичному сервері. Обмежена пропускна здатність сховища також використовується двома типами потоків вводу-виводу: потоки вводу-виводу віртуальної машини, що обслуговують програму, і потоки вводу-виведення міграції, які здійснюють оперативну міграцію віртуальної машини. Враховуючи, що ці два типи потоків введення-виведення значно заважають один одному, оскільки вони спільно використовують ті самі обмежені ресурси, реальна пропускна здатність для потоку міграції набагато менша, ніж пропускна здатність мережі. Крім того,

нові оновлення, викликані потоком вводу-виводу віртуальної машини під час процесу міграції, також потрібно перенести на цільовий сервер, що додатково подовжить час міграції. Таким чином, для прискорення міграції живого сховища віртуальної машини потрібні ефективні та ефективні підходи до міграції.

• **Висока продуктивність вводу-виводу віртуальної машини**: продуктивність вводу-виводу в запущених віртуальних машинах має постійно відповідати Угоді про рівень обслуговування (SLA - Service Level Agreement). Під час міграції живого сховища віртуальної машини програми всередині віртуальної машини, що мігрує, все ще працюють, і вони повинні не помічати процесу міграції. Однак доступний ресурс зберігання сильно розтягнутий через додаткові потоки міграції, які потребують ресурсів. В одному з експериментів пропускна здатність VM IO (Virtual Machine Input-Output) впала з 94,59 МБ/с до 65,80 МБ/с. Якщо потік міграції агресивно споживає ресурс зберігання, продуктивність вводу-виводу віртуальної машини, що мігрує, суттєво знижується, таким чином порушуючи попередньо визначену угоду про рівень обслуговування. У крайньому випадку віртуальна машина зупиниться, і програми взагалі не зможуть працювати. У той же час спільно розташовані віртуальні машини, які знаходяться на тому самому фізичному сервері та мають таку саму важливість, що й віртуальна машина, що мігрує, також підлягають зменшеному ресурсу зберігання та погіршеній продуктивності вводу-виводу. Таким чином, постачальник хмарних послуг повинен надати однакову гарантію продуктивності для всіх спільно розташованих і одночасно запущених віртуальних машин на фізичному сервері.

• **Можливість одночасного переміщення кількох віртуальних машин:** враховуючи широке розгортання віртуальних машин у центрах обробки даних, зазвичай мігрують кілька віртуальних машин з одного сервера або кілька віртуальних машин на один сервер. Для найкращого

використання ресурсів відображення від віртуальних машин до фізичних серверів має динамічно налаштовуватися відповідно до характеристик робочого навантаження та пріоритетів віртуальної машини [15]. У таких випадках на фізичний сервер буде введено кілька ресурсоємних потоків міграції чий загальний ресурс зберігання залишається незмінним протягом періоду міграції. Це призводить до різкого скорочення доступного ресурсу зберігання для однієї віртуальної машини. Тому набагато складніше досягти прийнятної продуктивності вводу-виводу віртуальної машини для всіх запущених віртуальних машин, переміщуючи кілька віртуальних машин до місць призначення з розумною швидкістю міграції.

- **Можливість перенесення кількох знімків віртуальної машини**: як зазначалося раніше, знімки віртуальної машини широко використовуються для відновлення віртуальної машини після збою системи та втрати даних, але це має певну ціну. Крім образів віртуальних дисків та іншої інформації про стан віртуальної машини під час міграції живого сховища віртуальної машини також потрібно перенести знімки віртуальної машини до місця призначення. Що ще важливіше, розмір кожного знімка віртуальної машини не є незначним і значно змінюється, і він значною мірою залежить від трафіку запису до запущеної віртуальної машини. Візьмемо приклад використання кластера Aliyun [14], розмір кожної віртуальної машини становить 40 ГБ, а кожен знімок віртуальної машини становить у середньому 8 ГБ, що означає, що 20% образів віртуальних дисків було змінено з моменту останнього знімка. Таким чином, знімки віртуальної машини ще більше погіршуватимуть продуктивність міграції живого сховища віртуальної машини.

Було запропоновано ряд підходів для покращення міграції живого сховища, включаючи оптимізацію послідовності передачі блоків даних [12] і робочого процесу міграції [11], зменшення надлишкової передачі даних [16] і використання гетерогенних пристроїв зберігання [17]. Після поглибленого

вивчення ми виявили, що всі ці підходи не в змозі вирішити життєво важливу проблему інтерференції вводу-виводу між процесом вводу-виводу віртуальної машини та процесом вводу-виводу міграції, оскільки обидва типи процесів вводу-виводу використовують той самий критичний шлях вводу-виводу, читаючи з/записуючи в той самий спільний пристрій зберігання даних. Через суперечку ресурсів вводу-виводу та взаємодію запитів між двома різними типами вводу-виводу не тільки подовжуватиметься черга запитів вводу-виводу на диску, але й частішати будуть трудомісткі операції пошуку диска. У результаті продуктивність процесу введення-виведення віртуальної машини буде серйозно знижена. Наші експериментальні результати показують, що пропускна здатність VM IO зменшується до 6 разів.

1.3 Опис концепції міграції динамічних станів пам’яті на основі віртуальних машин

Міграція віртуальної машини (VM), в принципі — це передача стану ЦП, стану пам’яті, стану пристрою, стану мережі, стану сховища та інших станів VM з одного фізичного вузла на інший через LAN або WAN [19, 20]. Одним із простих способів виконати операцію міграції віртуальної машини є призупинення віртуальної машини в джерелі, потім передача станів віртуальної машини та, нарешті, відновлення роботи віртуальної машини в місці призначення. Перевагою цього підходу є простота і легкість реалізації. Однак йому потрібно перервати запущену ОС і програми у віртуальній машині, чий тривалий час простою нестерпний для програм у безперервних службах. Таким чином, динамічна міграція віртуальної машини, яка переносить віртуальні машини на льоту, стає обов’язковою функцією моніторів віртуальних машин (VMM - Virtual Machine Monitors), і більшість, якщо не всі, сучасні VMM підтримують оперативну міграцію віртуальної машини. Серед передач різних станів віртуальної машини стан пам’яті

віртуальної машини зазвичай займає більшу частину часу міграції, і було запропоновано ряд підходів для прискорення міграції стану живої пам’яті [21, 22].

Загалом підходи до міграції пам’яті можна розділити на дві категорії:

- міграція пам’яті до копіювання;
- міграція пам’яті після копіювання.

Переміщення пам’яті перед копіюванням копіює всі сторінки пам’яті від джерела до місця призначення. Оскільки багато сторінок пам’яті можуть бути змінені ОС і програмами під час передачі, потрібні повторні передачі змінених (брудних) сторінок, доки швидкість повторної передачі не буде рівною або вищою за швидкість модифікації. Після досягнення цієї точки VMM призупинить роботу віртуальної машини на вихідному вузлі, передасть решту брудних сторінок і відновить роботу віртуальної машини на вузлі призначення. Час простою коливається від мілісекунд до секунд залежно від кількості брудних сторінок пам’яті. Під час міграції пам’яті після копіювання, навпаки, VMM призупиняє роботу віртуальної машини на вихідному вузлі, передає мінімальну підмножину станів віртуальної машини (наприклад, стан ЦП, стан пристрою та стан мережі), відновлює роботу віртуальної машини на вузлі призначення хоча більша частина стану пам'яті все ще знаходиться на вихідному вузлі. Потім ініціюється фоновий процес копіювання, щоб перенести решту сторінок пам’яті від джерела до місця призначення. Коли віртуальна машина намагається отримати доступ до сторінок, які не було передано, помилки сторінки будуть створені та перехоплені VMM на цільовому вузлі та перенаправлені на вихідний вузол через мережу. Порівняно з міграцією пам’яті до копіювання, міграція пам’яті після копіювання має набагато менший час простою, водночас значно погіршуючи продуктивність програм користувача під час міграції. Міграція пам’яті до + після копіювання спрямована на досягнення балансу між продуктивністю користувача та простоєм. Під час міграції пам’яті перед копіюванням після

завершення передачі початкового стану пам’яті VMM призупиняє віртуальну машину на джерелі та відновлює її на місці призначення [23, 12].

Оскільки затримка доступу постійних систем зберігання все ще на кілька порядків повільніша, ніж у чіпів енергозалежної пам’яті, сучасні операційні системи агресивно кешують дані із системи зберігання в пам’яті, щоб приховати тривалу затримку доступу. Таким чином, велика частина даних кешується в пам’яті за допомогою, а дубльована копія в системі зберігання. В [24] вказано, що обсяг дублювання даних між пам’яттю та пристроєм зберігання перевищує 55% або 2.1 Гб даних на сервері Linux. Коли мова заходить про живу міграцію віртуальної машини, перенесення цих дубльованих сторінок пам’яті з вихідного сервера на цільовий сервер займає не тільки багато часу, але й не потрібно. На основі цього спостереження вони пропонують відстежувати дубльовані сторінки пам’яті на вихідному сервері під час виконання. Під час міграції замість переміщення цих дублікатів сторінок через з’єднання з обмеженою швидкістю до пункту призначення цільовий сервер безпосередньо отримує ці сторінки зі спільного сервера зберігання. Таким чином, загальна передача даних значно зменшується, а продуктивність живої міграції також покращується [24, 25].

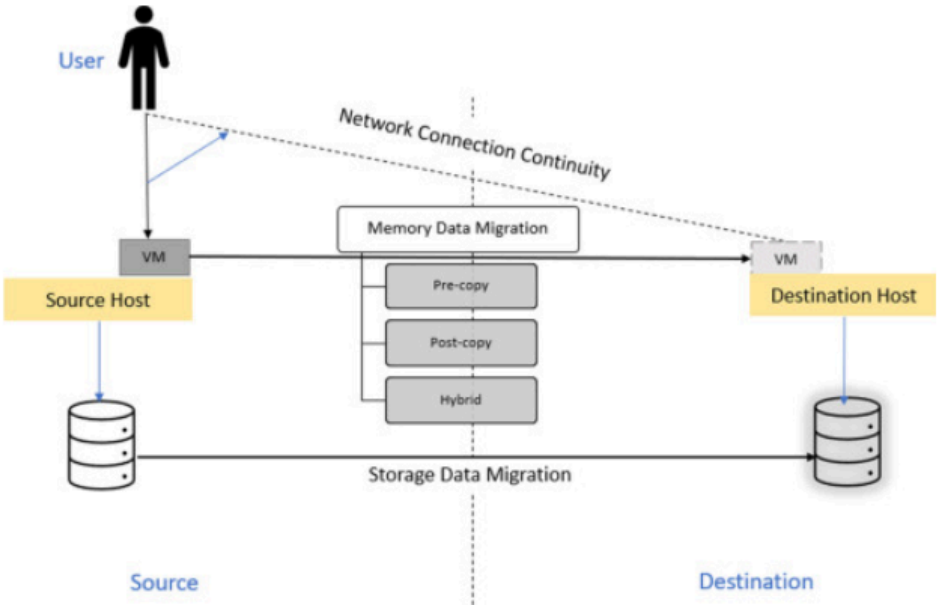


Рисунок 1.2 – Алгоритм живої міграції VM

В роботі [26] пропонують класифікувати сторінки пам’яті на кілька типів відповідно до різних характеристик, таких як висока подібність слів, низька подібність слів, велика кількість нульових байтів, а потім застосовувати різні алгоритми стиснення для стиснення сторінок пам’яті з різними властивостями. У результаті можна досягти кращого компромісу між обчисленням і ступенем стиснення. Через менший обсяг передачі даних і низькі накладні витрати на стиснення під час періоду живої міграції загальний час міграції та час простою значно скорочуються.

В дослідженні [27] пропонують використовувати алгоритм дельта-стиснення для покращення продуктивності живої міграції віртуальної машини, наприклад, скорочення загального часу міграції та простою віртуальної машини. У цьому підході дельта-сторінка пам’яті обчислюється для кожної брудної сторінки, а потім дельта-сторінка стискається та переноситься на сервер призначення, а не відповідна необроблена сторінка пам’яті. На цільовому сервері сторінку необробленої пам’яті можна відмінити від дельта-сторінки та попередньої копії. Оскільки в дельта-сторінці багато нульових бітів, стискати дельта-сторінки набагато легше й ефективніше, ніж сторінки необробленої пам’яті. Таким чином, загальна передача даних під час живого процесу VM значно зменшується [27].

В роботі [28] розробили нову схему міграції віртуальної машини в режимі реального часу на основі методів перевірки вказівки/відновлення та трасування/відтворення. Файли трасування виконання генеруються у вихідному вузлі, який містить достатньо інформації, щоб відтворити тривале виконання віртуальної машини інструкція за інструкцією. Через ітераційну передачу файлів трасування з вихідного вузла на вузол призначення вся інформація про стан віртуальної машини буде синхронізована ¬на сервері призначення. Завдяки меншому розміру файлів трасування порівняно з

брудними сторінками пам’яті час простою міграції віртуальної машини та споживання пропускної здатності мережі значно зменшуються.

Дослідники в [29] розробити та реалізувати схему живої міграції віртуальної машини на основі посткопіювання в гіпервізорі XEN. У їх реалізації застосовуються адаптивне попереднє розповсюдження сторінок і динамічне саморозповсюдження, щоб зменшити загальний час міграції та час простою віртуальної машини під час живої міграції віртуальної машини.

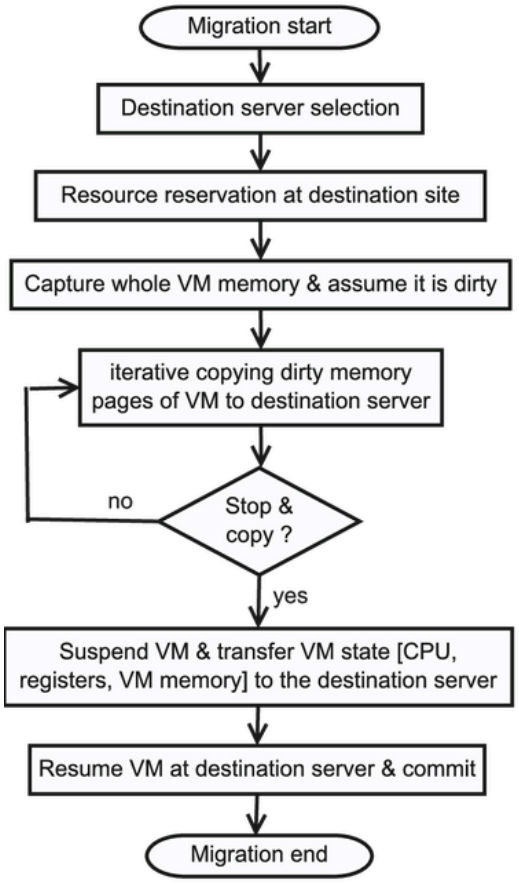


Рисунок 1.3 – Основні етапи міграції віртуальної машини які виконуються в техніці попереднього копіювання

За допомогою адаптивного попереднього підкачки робочий набір пам’яті віртуальних машин можна проаналізувати за послідовністю попередніх запитів на сторінку пам’яті, щоб сторінки пам’яті могли проактивно надсилатися на цільовий сервер до того, як віртуальні машини надсилатимуть

запити на доступ до них. За схемою динамічного самороздування віртуальна машина може повернути вільні сторінки пам’яті до основного гіпервізора до того, як почнеться оперативна міграція віртуальної машини. Таким чином, загальний обсяг пам’яті віртуальної машини, що мігрує, значно зменшується.

В роботі [30] стверджується, що живу міграцію віртуальної машини можна покращити використовуючи помічник програми. Наприклад, багато Java-додатків працюють у віртуальних машинах і є багато брудних сторінок, які чекають повернення сміття JVM. Видалення цих брудних сторінок із процесу живої міграції віртуальної машини не тільки прискорить процес поточної міграції віртуальної машини, але й не вплине на самі програми. Експериментальні результати показують, що час завершення, мережевий трафік передачі сторінок пам’яті та час простою програми покращилися на 90% порівняно зі звичайною схемою живої міграції віртуальної машини.

У віртуалізованому середовищі сторінки фізичної пам'яті знаходяться під контролем гіпервізори, які потім надають окремим віртуальним машинам сторінки віртуальної пам’яті за запитом. Усі сторінки віртуальної пам’яті, які виділені віртуальним машинам і не використовуються ними, знаходяться в пулі вільної пам’яті. Вміст цих сторінок вільної пам’яті не має відношення до віртуальної машини, тому їх можна розглядати як нульові сторінки. Ефективність живої міграції віртуальної машини буде значно покращена, якщо ці безкоштовні сторінки в межах віртуальної машини, що переносяться, можна буде виявити та видалити з процесу поточної міграції. В [31] розробили нову схему самоаналізу, яка може ефективно ідентифікувати пул вільної пам’яті без порівняння байтів сторінок пам’яті. За допомогою такої схеми інтроспекції віртуальної машини можна суттєво покращити як оперативну міграцію віртуальної машини, так і дедуплікацію пам’яті віртуальної машини .

PMigrate [32] — це структура, яка спрямована на розпаралелювання живої міграції віртуальної машини, оскільки зростаючий обсяг ресурсів,

виділених окремим віртуальним машинам, також пропонує можливості використовувати такі ресурси для розпаралелювання живої міграції віртуальної машини. Для перенесення інформації про стан віртуальної машини можна використовувати не лише десятки vCPU, але й десятки портів мережевих адаптерів. Крім того, вони розробляють динамічну дрібнозернисту абстракцію блокування, блокування діапазону, щоб збільшити паралелізм одночасних змін у спільному адресному просторі пам’яті.

В [23] аналізують чотири популярні методи оптимізації для живої міграції віртуальної машини, включаючи дельта-стиск, пропуск сторінки, дедуплікацію підсторінки та стиснення даних. Вони демонструють, що приріст продуктивності будь-якої техніки оптимізації тісно пов’язаний із характеристиками програми, шляхом проведення експериментів з живої міграції віртуальної машини з різними методами оптимізації та різними програмами. Крім того, було надано кілька вказівок щодо вибору відповідної техніки оптимізації та можливої комбінації різних схем оптимізації.

Enlighted Post-Copy [33] — це схема оптимізації для живої міграції віртуальної машини, що використовує інформацію про час виконання віртуальної машини як просвітлення під час процесу живої міграції віртуальної машини. Перед миттєвою міграцією віртуальної машини інформація про освітлення, яка містить інформацію про час виконання віртуальної машини, наприклад робочий набір віртуальної машини та вільні сторінки, передається гіпервізору. Після відновлення роботи віртуальної машини на сервері призначення сторінки пам’яті в робочому наборі віртуальної машини спочатку передаються на сервер призначення. У той же час сторінки пам’яті, які належать до вільного пулу віртуальної машини, відкидаються на вихідному сервері. Таким чином, загальний час міграції значно скорочується, а продуктивність запущеної віртуальної машини покращується, оскільки на цільовому сервері виникає менше помилок сторінки.

SRVM [34] — це механізм підтримки гіпервізора для живої міграції віртуальної машини з пропуском через мережеві пристрої SR-IOV. Передача SR-IOV може забезпечити набагато кращу продуктивність віртуальної машини порівняно зі схемами паравіртуалізації. Однак це створює проблеми для живої міграції віртуальної машини, оскільки гіпервізори не можуть вільно зберігати/відновлювати перехід через такі пристрої, як пристрої паравіртуалізації. SRVM вирішує ці проблеми, надаючи підтримку гіпервізорів для відстеження брудних сторінок пам’яті та надання VF після живої міграції віртуальної машини. У той же час він не вимагає жодних змін у гостьовій ОС чи драйвері. Завдяки схемі SRVM система віртуалізації може отримати як високу продуктивність віртуальної машини (за допомогою пристрою SR-IOV), так і гнучку можливість живої міграції віртуальної машини (з підтримкою SRVM) одночасно.

Усі наведені вище дослідження покращують міграцію віртуальної машини для різних сценаріїв. Наприклад, підхід [30] може бути застосований лише до віртуальних машин, у яких запущені програми Java. Водночас вони відрізняються від наших підходів до міграції (WAIO, SnapMig та IOFollow), оскільки зосереджені на рівні основної пам’яті живої міграції віртуальної машини зі спільним сховищем, тоді як наші підходи в основному покращують міграцію оперативної пам’яті віртуальної машини на рівні зберігання без спільне зберігання. Однак вони дають нам загальну картину найсучасніших досліджень щодо розробки рішення для покращення продуктивності міграції живого сховища віртуальної машини.

1.4 Опис процесу міграції сховища даних

У типовому процесі живої міграції віртуальної машини в неспільному середовищі зберігання передача стану пам’яті та образів віртуальних дисків займає більшу частину часу міграції та споживаної пропускної здатності

мережі. Оскільки міграція образів віртуальних дисків зазвичай займає набагато більше часу, ніж міграція стану пам’яті, дослідницька проблема того, як забезпечити ефективну та результативну міграцію живого сховища віртуальної машини, привернула велику увагу наукових кіл та промисловості [35, 36, 37].

У ранньому поколінні VMware vSphere vMotion [11] схема моментальних знімків використовує знімки віртуальної машини та ітеративно об’єднує серію знімків від джерела до місця призначення. Кожен знімок не лише зберігає стан живлення віртуальної машини, як-от увімкнено, вимкнено чи призупинено, але також містить усі файли, які складають цю віртуальну машину, включаючи віртуальні диски, обсяг пам’яті, інтерфейс віртуальної мережі та інші пристрої [38]. Щойно розмір останнього знімка перевищить попередньо визначений поріг, ітерація припиняється; віртуальна машина призупиняється в джерелі, а потім відновлюється в місці призначення. Через проблему продуктивності та узгодженості ця схема рідко використовується в поточних системах. На відміну від Snapshot, vMotion DBT (Dirty Block Tracking) [11] виконує оновлення на місці, замість того, щоб реєструвати записи у файлі знімка, і використовує брудну таблицю блоків для відстеження записів під час останньої ітерації міграції сховища. І Snapshot і DBT мають спільну головну проблему: час простою між призупиненням і відновленням віртуальної машини може бути відносно довгим під час інтенсивного запису. Щоб вирішити цю проблему, пропонується дзеркальне відображення IO [11] для віддзеркалення кожного запиту на запис як до вузлів джерела, так і до вузлів призначення під час живої міграції. У фоновому режимі зображення віртуальних дисків віртуальної машини (крім данb[в нових запитах на запис) переносяться до місця призначення. Після завершення цієї міграції образи віртуальних дисків у вихідному вузлі автоматично збігаються з образами у вузлі призначення , оскільки всі дані в нових запитах на запис уже віддзеркалено до пункту призначення. Таким чином, коли віртуальну машину

призупинено в джерелі, лише стан пам’яті віртуальної машини та інші стани потрібно передати до пункту призначення перед відновленням роботи віртуальної машини на вузлі призначення. IO Mirroring значно скорочує час простою за рахунок збільшення мережевого трафіку (віддзеркалення кожного запиту на запис).

В [39] пропонують трифазний алгоритм міграції, який забезпечує мінімальний час простою та підтримує послідовність системи.

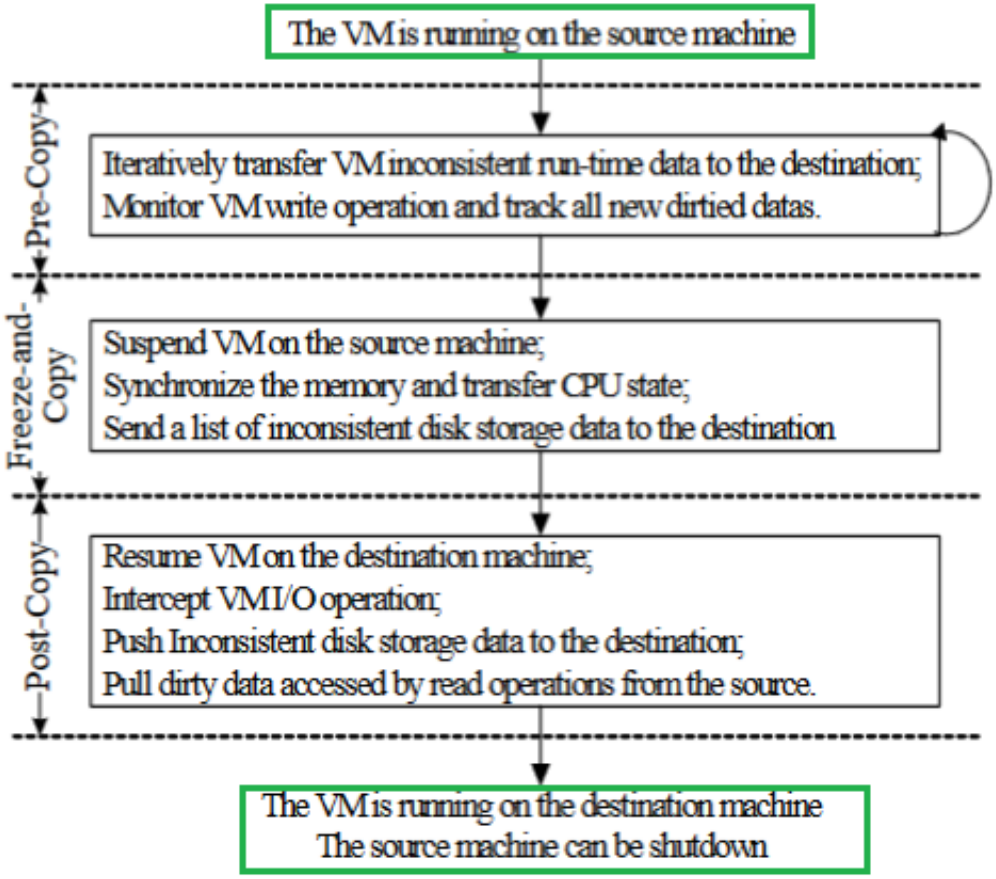


Рисунок 1.4 – Трифазна жива міграція всієї системи

Крім того, вони представляють схему інкрементної міграції для полегшення міграції назад до вихідного вузла. Результати оцінки показують, що час простою цього алгоритму становить приблизно 100 мілісекунд, що близько до часу міграції в середовищі шардового зберігання. Час міграції значно скорочується завдяки уникненню непотрібної передачі даних. Однак

скорочення часу міграції можна досягти лише тоді, коли віртуальна машина збирається перейти назад на сервер, на якому зберігаються попередні образи віртуальних дисків віртуальної машини.

Щоб уникнути непотрібної повторної передачі часто оновлюваних блоків під час ітерацій брудних передач блоків, в [12] запропоновано схему, яка відрізняє часто оновлювані області від рідко оновлюваних областей і передає рідко оновлювані блоки даних перед часто оновлюваними блоками даних. Таким чином зменшується ймовірність того, що переміщені блоки даних стануть брудними та вимагатимуть повторної передачі до завершення живої міграції віртуальної машини. Таким чином, як загальний обсяг переданих даних, так і час міграції значно зменшуються. Однак ця схема не може покращити продуктивність VM IO під час живої міграції. Процес введення-виведення віртуальної машини отримує доступ до популярної області образів віртуальних дисків; тоді як процес міграції вводу-виводу отримує доступ до непопулярної області образів віртуальних дисків. Тому диск має рухатися туди-сюди, що призводить до значного зниження продуктивності введення-виведення під час міграції.

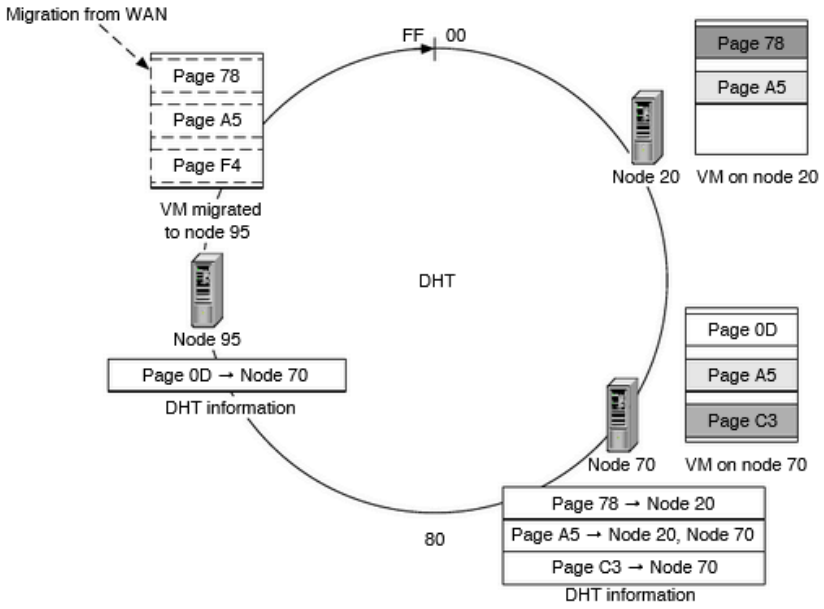


Рисунок 1.5 – Розподілена хеш-таблиця Shrinker

Shrinker [16] — це розподілена система, яка здатна мігрувати віртуальний кластер через WAN. Він має дві вбудовані служби: Coordination Service (на вихідному сайті) та Indexing Service (на цільовому сайті).

Служба координації відстежує хеш-значення сторінок пам’яті та блоків віртуального диска, які вже були передані на сайт призначення, щоб гіпервізори на стороні джерела могли виконувати дедуплікацію даних шляхом заміни дубльованої передачі сторінок пам’яті та блоків диска їхніми хеш-значеннями. Служба індексування записує хеш-значення та інформацію про розташування сторінок пам’яті та дискових блоків на стороні призначення. Гіпервізори в місці призначення можуть реконструювати пам’ять віртуальної машини та образи віртуальних дисків за допомогою зв’язку між службою індексування та іншими гіпервізорами, які зберігають реальні дані. У результаті загальний час передачі даних і міграції значно скорочується. Ця схема може зменшити надлишкову передачу даних, а не кількість даних, що зчитуються з диска. Кожен блок даних у образах віртуальних дисків зчитується в пам’ять, а потім обчислюється відбиток. На основі відбитка пальця Shrinker може вирішити, передати блок даних чи його відбиток. Таким чином, перешкоди вводу-виводу між запитами вводу-виводу VM і запитами введення-виведення міграції все ще існують на диску. Крім того, ця схема збільшує накладні витрати на обчислення (генерування відбитків пальців для кожного блоку даних).

В дослідженні [17] враховують різницю в швидкості між HDD і SSD, а також проблему зносу SSD, щоб оптимізувати міграцію живого сховища. Вони пропонують три варіанти оптимізації:

- міграція сховища з низьким резервуванням, призначена для зменшення загальної передачі даних;
- підвищення продуктивності віртуальної машини вводу-виводу під час міграції віртуальної машини з твердотільних накопичувачів на жорсткі

диски шляхом використання вищої продуктивності твердотільних накопичувачів і міграції сховища з низьким надлишком на основі джерела;

- асинхронний IO механізм віддзеркалення для значного скорочення часу відповіді вводу-виводу для кожного запиту в запущеній віртуальній машині під час міграції.

У середовищі глобальної мережі продуктивність віртуальної машини вводу-виводу погіршиться, оскільки віртуальній машині потрібно отримувати останні оновлення з цільового сервера (у глобальній мережі), що зазвичай набагато повільніше, ніж у середовищі локальної мережі.

В роботі [40] побудували систему міграції живого сховища віртуальної машини, яка називається XvMotion, щоб перенести віртуальну машину на великі відстані між гетерогенними системами, з продуктивністю, подібною до міграції віртуальної машини в локальній мережі. Щоб досягти цієї мети, вони запропонували кілька методів і оптимізацій, включаючи структуру транспортування потоків, асинхронне віддзеркалення вводу-виводу, координацію пам’яті та диска, оглушення під час надсилання сторінки тощо.

Зрештою, продуктивність системи зберігання у віртуалізованому середовищі відіграє життєво важливу роль для продуктивності міграції живого сховища віртуальної машини. Багато інших дослідницьких робіт також опосередковано покращують продуктивність міграції живого сховища ВМ завдяки внеску в планувальник вводу-виводу [41, 42], файлові системи [43, 44, 45], твердотільні накопичувачі [46, 47, 48] і методи дедуплікації даних [49, 50].

Під час живої міграції віртуальної машини віртуальна машина виконує програми більшу частину часу (за винятком вікна простою), тому продуктивність вводу-виводу запущеної віртуальної машини має вирішальне значення в хмарному середовищі. Усі згадані вище схеми в основному зосереджені на скороченні загального обсягу передачі даних, часу міграції та простою віртуальної машини, але не покращують продуктивність

вводу-виводу віртуальної машини. Метою даної роботи є покращення продуктивності вводу-виводу віртуальної машини та продуктивності міграції під час міграції живого сховища віртуальної машини шляхом вирішення проблеми введення-виведення.

В принципі, наші схеми мають ту саму мету, що й попередні дослідницькі роботи, а саме пришвидшити продуктивність міграції живого сховища віртуальної машини та забезпечити розумну продуктивність вводу-виводу для міграційної віртуальної машини.

Висновки до розділу 1

В даному розділі проведено аналіз основних відомостей про системи та алгоритми міграції живих сховищ віртуальних машин, наведені переважаючі схемами міграції віртуальної машини, що стосуються різних проблем, деякі з яких представляють сучасний рівень, а також їхні ключові особливості.

РОЗДІЛ 2. МОДЕЛІ ТА ЗАСОБИ МІГРАЦІЇ ДАНИХ НА ОСНОВІ ВІРТУАЛЬНИХ МАШИН

2.1 Опис концепції міграції даних з врахування навантаження

Як було зазначено в розділі 1, існує багато підходів до оптимізації міграції живого сховища віртуальної машини [11, 38, 11, 11, 39, 12, 16, 17, 40]. Після поглибленого вивчення ми виявили, що всі ці підходи не в змозі вирішити життєво важливу проблему інтерференції вводу-виводу між процесом вводу-виводу віртуальної машини та процесом вводу-виводу міграції, оскільки обидва типи процесів вводу-виводу використовують той самий критичний шлях вводу-виводу, читаючи і записуючи в той самий спільний пристрій зберігання даних. Через суперечку ресурсів вводу-виводу та взаємодію запитів між двома різними типами вводу-виводу не тільки подовжуватиметься черга запитів вводу-виводу на диску, але й частішати будуть трудомісткі операції пошуку диска. У результаті продуктивність процесу введення-виведення віртуальної машини буде помітно знижена. Експериментальні результати показують, що пропускна здатність VM IO зменшується до 6 раз.

У цій роботі ми пропонуємо структуру вводу-виводу з урахуванням робочого навантаження (WAIO - Workload-Aware IO Outsourcing), щоб покращити як продуктивність віртуальної машини, так і продуктивність міграції під час міграції живого сховища. Основна ідея полягає в тому, щоб тимчасово захопити робочий набір цільової віртуальної машини та передати дані цього робочого набору на сурогатний пристрій протягом періоду міграції. Завдяки цьому процес вводу-виводу віртуальної машини може отримати доступ до сурогатного пристрою під час міграції, тоді як процес введення-виведення міграції отримує доступ до вихідного диска. У результаті

можна значно зменшити взаємодію вводу-виводу між обома типами процесів вводу-виводу та покращити загальну продуктивність міграції живого сховища. Сурогатним пристроєм може бути запасний SSD, запасний жорсткий диск (HDD) або вільне місце в іншому вузлі зберігання. Ця структура ортогональна до існуючих підходів до оптимізації, включаючи підходи DBT [12] і IO Mirroring [11] і це рівень підвищення продуктивності для більшості, якщо не всіх, схем живої міграції віртуальних машин. Крім того, цю структуру також можна використовувати для покращення продуктивності інших завдань віртуальної машини, таких як реплікація віртуальної машини [18], оскільки ці завдання зіткнуться з тією ж проблемою втручання введення-виведення, що й міграція живого сховища віртуальної машини. Емпірична оцінка прототипної системи показує, що структура WAIO може підвищити продуктивність віртуальної машини вводу-виводу до 11 раз порівняно з підходом DBT. З іншого боку, наша система може перенести віртуальну машину з вищою швидкістю, не жертвуючи суттєвою продуктивністю вводу-виводу віртуальної машини.

Підхід WAIO і робота [12] використовують характеристики доступу до вводу/виводу програми, але вони різними способами покращують продуктивність живої міграції віртуальної машини. Схема в [12] має на меті значно зменшити загальний обсяг переданих даних, використовуючи локальність робочого навантаження віртуальної машини.

Завдяки аналізу локальності робочого навантаження рідко оновлювані блоки даних відрізняються від часто оновлюваних блоків даних у віртуальних образах дисків. Рідко оновлювані блоки даних передаються перед часто оновлюваними блоками даних під час міграції, так що повторна передача блоків даних мінімізується, таким чином зменшуючи загальний обсяг передачі даних. Хоча WAIO також використовує локальність робочого навантаження, його методологія повністю відрізняється від методології Zheng. WAIO використовує локальність робочого навантаження для

захоплення та передачі робочого набору віртуальної машини сурогатному пристрою під час міграції, що не впливає на послідовність передачі блоків даних. Важливо, що WAIO є ортогональним і доповнює вищезазначені підходи та може ще більше вдосконалювати ці методи.

В таблиці 2.1 коротко порівнюється WAIO з наведеними вище підходами.

Таблиця 2.1

Порівняння між WAIO та найсучаснішими схемами

Features	DBT	IO Mirroring	Zheng	WAIO
Migration Time Reduction	✓	✓	✓	✓
VM IO Performance				✓
Workload Locality			✓	✓

Проблема перешкод вводу-виводу. Як зазначалося вище, ми припускаємо, що перешкоди вводу-виводу між IO міграції та VM IO є основною причиною низької швидкості міграції та зниження продуктивності VM IO. З одного боку, процес міграції віртуальної машини зчитує блоки даних із образів віртуальних дисків у джерелі та записує їх у образи віртуальних дисків у місці призначення. З іншого боку, операції вводу-виводу віртуальної машини обслуговуються гіпервізорами та спрямовуються до образів віртуальних дисків у джерелі.

Два одночасних, але незалежних потоку вводу-виводу призводять до конкуренції за головку жорсткого диска в джерелі, оскільки лише одна головка диска може виконувати операцію вводу-виводу в будь-який момент часу. Операції пошуку головки диска не тільки стають частішими, але й черга запитів на введення-виведення також стає довшою. Таким чином, важко

досягти кращої продуктивності віртуальної машини вводу-виводу та скоротити час міграції одночасно під час міграції живого сховища віртуальної машини. Ця проблема введення вводу-виводу явно залишається в існуючих підходах через співіснування IO VM та міграційних IO.

Щоб підтвердити нашу гіпотезу, були проведені експерименти на системі QEMU-KVM у середовищі локальної мережі.

Працююча віртуальна машина переноситься туди-сюди між двома серверами через мережу Ethernet 1 Гбіт/с. Запити на введення-виведення генеруються IOMeter [47] із запитами на читання/запис 60%/40% у віртуальній машині Windows 15 ГБ. Розмір кожного послідовного запиту становить 32 КБ, а кожного випадкового запиту — 4 КБ. Швидкість міграції становить 20,8 МБ/с. У таблиці 2.2 показано зниження продуктивності VM IO під час міграції живого сховища.

Таблиця 2.2

Продуктивність VM IO під час міграції

Throughput (MB/s)	Sequential IOs	Random IOs
No Migration	58.44	5.46
Live Migration	19.43	0.85

Як зазначено в цій таблиці, пропускна здатність зменшується на коефіцієнт 3,01 для послідовних запитів і 6,42 для випадкових запитів під час живої міграції порівняно зі сценаріями без міграції. Таке значне зниження продуктивності введення-виведення потенційно може спричинити призупинення запущених програм усередині віртуальної машини, що порушить угоду про рівень обслуговування.

Таблиця 2.3 показує ефективність міграції за різних швидкостей міграції. У цьому експерименті ми оцінюємо продуктивність послідовних запитів введення-виведення за різних швидкостей міграції.

Таблиця 2.3

Продуктивність VM IO за різної швидкості міграції

Migration Speed (MB/s)	10.26	40.14
VM IO Throughput (MB/s)	42.25	9.29
Migration time (s)	1498	382

Швидкість міграції встановлюється через систему QEMU-KVM. Як показано в цій таблиці, установивши більшу швидкість міграції 40,14 МБ/с, загальний час міграції скорочується до 25,5% від нижчої швидкості 10,26 МБ/с, а продуктивність віртуальної машини знижується в 4,55 рази. Очевидно, що враховуючи обмежені ресурси (наприклад, пропускну здатність сховища) і перешкоди вводу-виводу, існує компроміс між продуктивністю міграції та віртуальної машини, і той чи інший повинен поступатися.

Техніко-економічний аналіз аутсорсингу IO (вводу-виводу). Щоб пом’якшити проблему перешкод вводу-виводу, одним із можливих рішень є вихідні IO віртуальної машини, щоб IO міграції могли займати головку диска та виконувати послідовні операції читання, якщо це можливо. Цей момент також узгоджується з відомим фактом, що підхід до міграції сховища в автономному режимі (без зовнішніх запитів вводу-виводу) є набагато швидшим, ніж його реальний аналог (з одночасними зовнішніми та внутрішніми запитами вводу-виводу). Перш ніж ми зможемо використовувати аутсорсинг IO для оптимізації міграції живого сховища віртуальної машини, потрібно відповісти на два ключові запитання: які IO передати аутсорсингу та де?

Щоб відповісти на перше запитання, нам потрібно взяти до уваги характеристики робочого навантаження [48, 49]. По-перше, читання має бути синхронним, а запис може бути асинхронним. Це означає, що якщо записи можуть виконуватися сурогатним пристроєм зберігання для аутсорсингу

вводу-виводу, завершення запису може бути негайно повернено користувачеві без будь-яких перешкод для образів віртуальних дисків. По-друге, читання може отримати доступ до носія інформації образу віртуального диска та спричинити труднощі з аутсорсингом введення-виведення. Попередні дослідження вказують на те, що локальність доступу є однією з ключових характеристик веб-робочого навантаження і спостерігають, що 10% файлів, доступ до яких здійснюється на веб-сервері, відповідають приблизно 90% запитів і 90% переданих байтів [50].

У віртуалізованому середовищі запити на введення-виведення віртуальних дисків також мають сильну часову та просторову локальність. Наприклад, інше дослідження [12] показує, що на файловому сервері 72% блоків, які зчитуються під час процесу міграції, також були прочитані до початку міграції. Серед цих блоків 96% зчитуються більше трьох разів під час міграції. Сильна просторова локальність також підтверджується в цьому дослідженні. Зважаючи на сильну локальність запитів вводу-виводу, очікується, що робочий набір переміщеної віртуальної машини буде досить малим, щоб було життєздатним передати популярні запити на читання та всі записи на сурогатний пристрій.

Щоб відповісти на друге запитання, ми визнаємо та використовуємо всюдисущі резервні/безкоштовні ресурси зберігання в центрах обробки даних. У хмарному середовищі тимчасовий віртуальний диск може служити для зберігання робочого набору віртуальної машини, доки цей тимчасовий віртуальний диск не конкурує за пропускну здатність зберігання з оригінальними віртуальними дисками. Розміщення тимчасового віртуального диска досить гнучке, включаючи жорсткі диски, SSD або RAID. Якщо резервного пристрою немає, вільний простір для зберігання в системах зберігання при невеликих навантаженнях також може містити тимчасовий віртуальний диск.

2.2 Архітектура та алгоритм функціонування системи міграції з врахуванням робочого навантаження

У цьому розділі спочатку представимо огляд архітектури системи WAIO, а потім докладно представляємо алгоритм WAIO.

Архітектура WAIO. На рисунку 2.1 показано огляд архітектури системи WAIO. WAIO — це розширений модуль на рівні стеку вводу-виводу гіпервізорів, який включає п’ять функціональних модулів:

- ідентифікація популярних даних;
- диспетчер сурогатного простору;
- перенаправлення вводу-виведення;
- засіб відновлення простору;
- інтерфейс адміністратора.

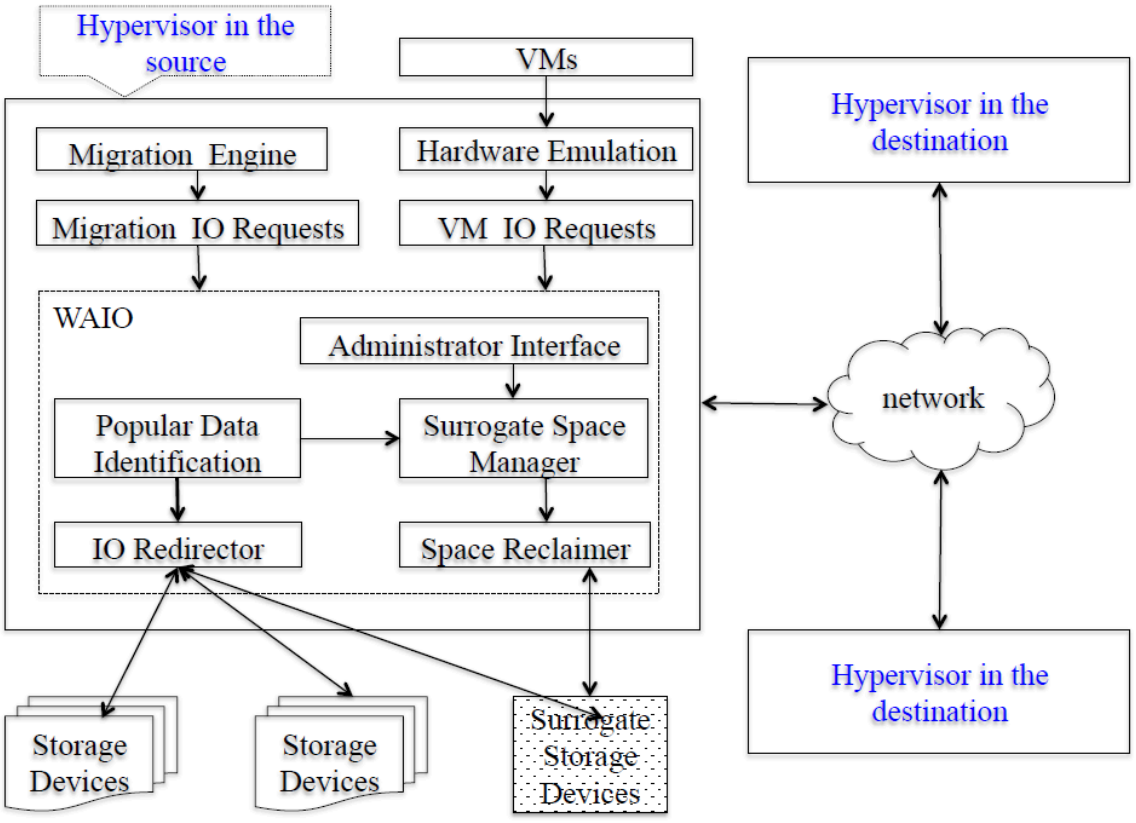


Рисунок 2.1 – Архітектура системи WAIO

Усі вони знаходяться у вихідному вузлі міграції живого сховища віртуальної машини. Обов'язки п'яти модулів детально розроблені таким чином. Ідентифікація популярних даних відстежує популярність запитів на читання від самої віртуальної машини в образах віртуальних дисків. Сурогатному пристрою передаються лише популярні блоки даних, які будуть читатися. Оскільки сурогатний пристрій обслуговує всі запити на запис, немає необхідності відстежувати популярність запитів на запис. Кожен образ віртуального диска запущеної віртуальної машини ділиться на частини фіксованого розміру та модуль ідентифікації даних який записує частоту доступу для кожного блоку. Якщо частота доступу до певного блоку перевищує попередньо визначене порогове значення, весь фрагмент буде передано на зовнішній пристрій для заміни. Усі наступні звернення до цього блоку будуть обслуговуватися сурогатним пристроєм, який усуває їх перешкоди процесу міграції.

Модуль міграції зазвичай сканує цілі віртуальні образи дисків, надсилаючи запити лише для читання. Більшість із цих запитів надсилаються лише один раз, за винятком запитів, які читають брудні блоки даних. Таким чином, на популярність блоків даних не впливають запити від модуля міграції.

IO Redirector перенаправляє відповідні запити IO із самої запущеної віртуальної машини на сурогатний пристрій. Він перенаправляє всі запити на запис на сурогатному пристрої, щоб зменшити трафік вводу-виводу на оригінальний пристрій зберігання. Тим часом усі популярні запити на читання, визначені модулем ідентифікації популярних даних, перенаправляються на сурогатний пристрій за допомогою перенаправлення вводу-виведення. Якщо сурогатний пристрій має лише часткові дані для запиту, IO Redirector надсилатиме запити на читання до оригінального пристрою зберігання та об’єднуватиме дані з оригінального пристрою з даними на сурогатному пристрої. Непопулярні запити на читання

спрямовуватимуться на оригінальний пристрій зберігання даних, а фрагмент даних буде передано на зовнішній пристрій зберігання даних, лише якщо частота доступу перевищує попередньо визначений поріг.

Запити на читання з модуля міграції можна перенаправляти або на оригінальний пристрій зберігання, або на сурогатний пристрій. У той час як оригінальний пристрій зберігання даних забезпечує основну частину образів віртуальних дисків, сурогатний пристрій забезпечує оновлені фрагменти даних. Більшість запитів буде перенаправлено на оригінальний пристрій зберігання через локальність робочого навантаження віртуальної машини.

Менеджер сурогатного простору відповідає за керування сурогатним пристроєм і збирання сміття на сурогатному пристрої. Перед тим, як дані будуть збережені на сурогатному пристрої, IO Redirector має запитати вільний простір для зберігання в диспетчера сурогатного простору.

Space Reclaimer використовується для відновлення всіх блоків у сурогатному пристрої після завершення міграції, щоб сурогатний пристрій був доступний для інших служб зберігання. Він також відповідає за звільнення структур даних у пам'яті.

Інтерфейс адміністратора надає системним адміністраторам інтерфейс для налаштування параметрів дизайну. Зокрема, WAIO збирає інформацію про сурогатний пристрій зберігання через цей інтерфейс.

Конструкція WAIO досить гнучка. По-перше, модуль ідентифікації даних можна реалізувати за допомогою різних алгоритмів, які фіксують локальність запитів на читання для різних програм. Ми використовуємо частоту доступу, щоб оцінити популярність фрагментів даних і існує багато інших алгоритмів для тієї ж мети [12]. WAIO може легко реалізувати будь-яку їх комбінацію для ефективного захоплення популярних запитів на читання для різних робочих навантажень. Можливо, краще зафіксувати та проаналізувати шаблони доступу до вводу-виводу перед початком міграції, а потім вибрати найбільш підходящий алгоритм у WAIO під час поточної

міграції. По-друге, сурогатним пристроєм зберігання може бути резервний SSD, HDD, RAID або вільний простір для зберігання на інших вузлах, якщо є вільна порівнянна пропускна здатність зберігання. Нарешті, WAIO можна включити в різні підходи до міграції (наприклад, DBT і IO Mirroring) та інші функції віртуальної машини (наприклад, реплікацію віртуальної машини). У цьому документі ми зосереджуємось на сценарії міграції живого сховища віртуальної машини.

Алгоритм WAIO. WAIO використовує робочий набір віртуальної машини та передає його сурогатному накопичувачу під час періоду міграції. Для цього WAIO має відстежувати таку інформацію в пам'яті, як показано на рисунку 2.2.

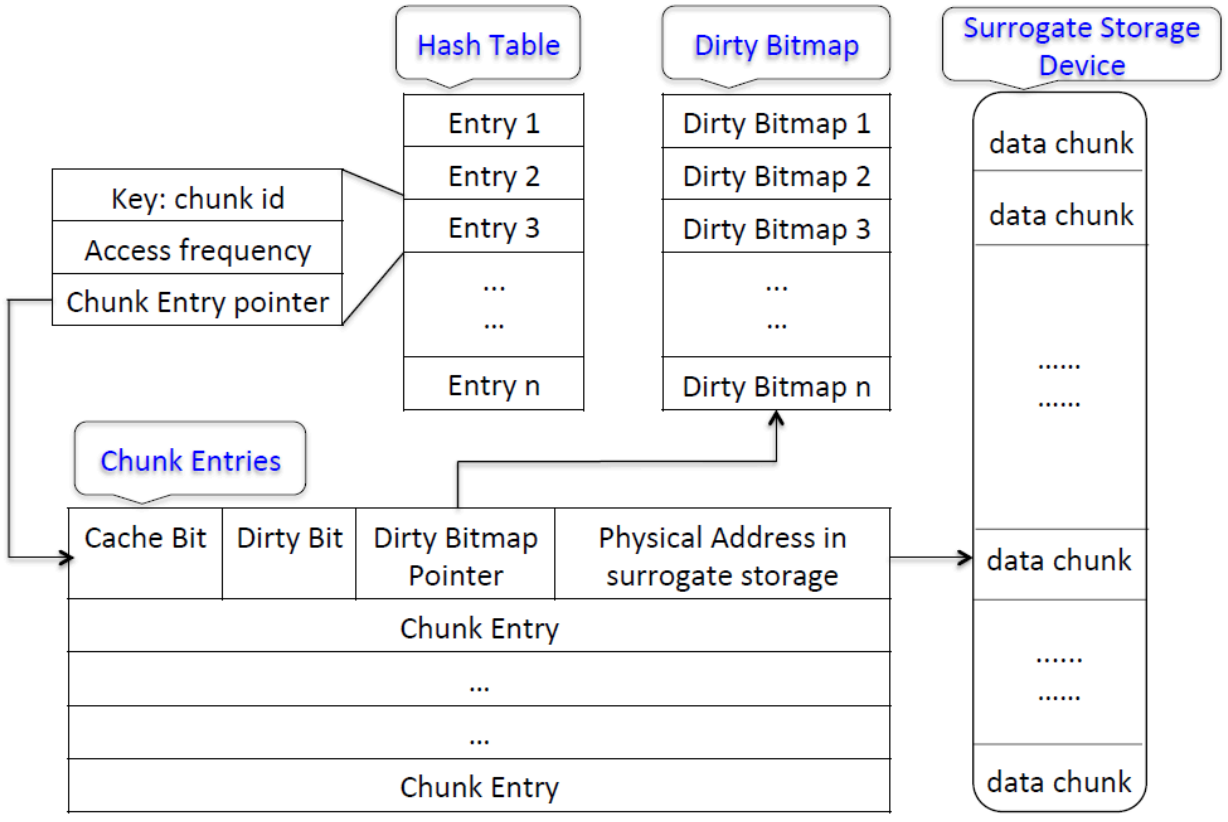


Рисунок 2.2 – Структура даних системи WAIO

Популярність кожної ділянки у віртуальних образах дисків: WAIO використовує хеш-таблицю для відстеження блоків даних, до яких було отримано доступ під час міграції. Коли доступ до блоку даних здійснюється

вперше, WAIO додає запис у хеш-таблицю для цього блоку даних. Ключ — це логічний ідентифікатор блоку даних, а значення містить кількість доступу (частоту) і вказівник на запис блоку. Для подальшого доступу нам потрібно постійно оновлювати відповідні записи в хеш-таблиці.

Статус кожного фрагмента: під час міграції фрагменти даних образів віртуальних дисків мають різні стани. Щоб відстежувати таку інформацію про стан, WAIO використовує кілька полів у записі послідовності, зокрема Cache Bit, Dirty Bit, растровий покажчик і фізичну адреса. Біт кешу записує, чи відповідний фрагмент уже кешується в сурогатному сховищі (біт кешу встановлено на 1) чи ні (біт кешу встановлено на 0). Брудний біт (Dirty Bit) вказує, чи були записані нові дані в цей блок під час періоду міграції. Якщо так, брудний біт має значення 1; в іншому випадку встановлено значення 0. Якщо встановлено Dirty Bit, Dirty Bitmap потрібне для детального відстеження оновлень цього блоку (наприклад, які байти в цьому фрагменті було перезаписано), а його адреса зберігається в полі Dirty Bitmap Pointer запис фрагмента. Нарешті, початкова адреса блоку даних у сурогатному сховищі записується в поле Physical Address у полі Chunk Entry.

Під час міграції живого сховища віртуальної машини запущена віртуальна машина та модуль міграції в гіпервізорі надсилають запити вводу-виводу до образів віртуальних дисків цієї віртуальної машини. Якщо запиту введення-виведення потрібен доступ до кількох послідовних блоків даних, запит буде розділено на кілька підзапитів, які отримають доступ до відповідних блоків даних одночасно. Початковий запит не буде виконано, доки всі підзапити не завершать свої операції введення-виведення. Якщо запиту введення-виведення потрібен лише доступ до однієї частини даних, тоді сам запит також розглядається як підзапит. Усі підзапити обслуговуватимуться оригінальним пристроєм зберігання та сурогатним пристроєм зберігання.

Підзапити IO від запущеної віртуальної машини можуть бути запитами на читання та запис. Для підзапитів на читання WAIO спочатку шукає в хеш-таблиці відповідний фрагмент даних. Якщо такого запису немає (фрагмент даних не міститься в сурогатному пристрої), WAIO обслуговує цей підзапит, зчитуючи фрагмент даних з оригінального диска. Крім того, весь фрагмент даних передається на зовнішній пристрій сурогатного зберігання, а в хеш-таблицю вставляється новий запис, який вказує, що цей фрагмент даних знаходиться в сурогатному сховищі. Якщо запис існує, WAIO перевірить запис фрагмента, щоб визначити статус цього фрагмента даних. Коли встановлено біт кешу (весь фрагмент кешується в сурогатному сховищі), цей підзапит буде обслуговуватися безпосередньо сурогатний запам'ятовуючий пристрій.

В іншому випадку WAIO виконає такі операції:

- прочитає весь фрагмент із оригінального диска;
- об’єднає їх із брудними даними в сурогатному сховищі;
- поверне дані до підзапиту;
- оновить інформацію про запис фрагмента.

Для підзапитів на запис WAIO потрібно лише записати дані на сурогатний пристрій зберігання та оновити відповідний запис фрагмента в хеш-таблиці.

Проблеми узгодженості даних. Узгодженість даних є ключовим питанням для проектування систем від одного вузла до великомасштабних розподілених систем. У системі WAIO присутні два аспекти:

- усі структури даних у пам’яті мають безпечно зберігатися;
- дані, передані на сурогатний пристрій зберігання, мають надійно зберігатися, доки їх не витребує WAIO.

По-перше, щоб запобігти неочікуваній втраті даних структур даних у пам’яті, включаючи хеш-таблицю, записи фрагментів і брудні растрові зображення, WAIO зберігає їх в енергонезалежній ОЗУ (NVRAM -

non-volatile RAM). Загальний розмір цих структур даних у пам’яті дуже малий, тому це не спричинить значних додаткових витрат на обладнання. Наприклад, враховуючи образ віртуального диска розміром 1 ТБ із розміром фрагмента 1 МБ, споживання простору в гіршому випадку становитиме лише 25 МБ, коли кожен фрагмент передається на сторонній пристрій для зберігання даних. Крім того, NVRAM вже широко розгорнуто на серверах зберігання в хмарних центрах обробки даних з метою підвищення надійності системи та підвищення продуктивності запису. Таким чином, можливо використовувати існуючу NVRAM для зберігання цих структур даних у пам’яті.

По-друге, ми можемо покластися на вже вбудований механізм надійності (наприклад, RAID, ECC, репліки) сурогатного пристрою зберігання для захисту даних, переданих WAIO. Крім того, ці ресурси необхідні лише в період міграції. Після завершення живої міграції віртуальної машини вони будуть витребувані та доступні для інших служб.

2.3 Оцінка ефективності підходу міграції

Ми реалізуємо дослідимо прототип WAIO в системі QEMU-KVM. Оскільки WAIO є ортогональним і доповнює існуючі підходи до міграції живого сховища віртуальної машини, ми оцінюємо його ефективність, проводячи порівняння продуктивності між DBT, репрезентативним найсучаснішим підходом і DBT, розширеним за допомогою WAIO, у різних сценаріях міграції. Оцінки керуються трьома трасування блоку, які представляють різні характеристики робочого навантаження.

Реалізація прототипу. У системі QEMU-KVM функціональність вводу/виводу забезпечується системою QEMU, яка є емулятором віртуальної машини з відкритим кодом. QEMU може емулювати багато віртуальних пристроїв для віртуальних машин, включаючи віртуальні диски та віртуальні

мережеві інтерфейсні карти. Драйвер QEMU фіксує всі запити введення-виведення від віртуальних машин, а потім передає їх модулю ядра KVM. Модуль ядра KVM надішле ці запити до відповідної програми QEMU та поверне результати до віртуальної машини. Програма QEMU обробляє запити вводу-виводу від імені віртуальної машини. WAIO реалізовано в додатку QEMU, тому він може отримувати робочий набір запущеної віртуальної машини.

Коли гіпервізор отримує команду живої міграції віртуальної машини, ініціюється WAIO та створюється хеш-таблиця в пам’яті. WAIO перенаправляє запити вводу-виводу як з апаратного рівня емуляції (VM IO), так і з модуля міграції (Migration IO) у функціях bdrv_co_readv_em і bdrv-co-writev-em. Популярні прочитані дані та нещодавно записані дані передаються на сурогатний пристрій (налаштований інтерфейсом адміністратора WAIO), а хеш-таблиця в пам’яті та записи фрагментів оновлюються одночасно. Після завершення процесу міграції хеш-таблиця в пам’яті та простір для зберігання на сурогатному пристрої повертаються засобом відновлення простору WAIO. WAIO реалізовано в спрощений спосіб, який вимагає лише 638 рядків коду, доданих або змінених у програмі QEMU.

Імітаційна установка для експерименту. Експериментальна платформа складається з двох серверів як джерела та пункту призначення для міграції живого сховища віртуальної машини. Кожен сервер налаштовано з конфігурацією 8 ГБ пам'яті DDR і два жорсткі диски по 1 ТБ, система Ubuntu. Сервер на стороні джерела міграції має SSD на 80 ГБ як сурогатний пристрій. Ці два сервери з’єднані мережею Ethernet 1 Гбіт/с.

Щоб виміряти продуктивність WAIO, ми переносимо працюючу віртуальну машину між двома серверами. Віртуальна машина налаштована з 1 віртуальним ЦП, 2 ГБ пам’яті, 1 віртуальним диском, 1 віртуальною мережевою інтерфейсною картою та системою Ubuntu. Під час міграції ми відтворюємо трасування на рівні блоку та збираємо продуктивність

вводу-виводу у віртуальній машині. Storage Performance Council [46] опублікував декілька трасувань на рівні блоків для дослідницьких цілей і ці трасування широко використовувалися для оцінки продуктивності системи зберігання. Трасування Financial1 і Financial2 збираються з програм OLTP у великій фінансовій установі, а трасування WebSearch1 збирається з веб-пошукової системи. Ключові характеристики узагальнено в таблиці 2.4.

Таблиця 2.4

Характеристики об’єктів трасувань

Trace Name	Read Ratio	IOPS	Avg. Req. Size (KB)
Financial1	32.8%	69	6.2
Financial2	82.4%	125	2.2
WebSearch1	100%	113	15.1

Під час імітаційного експерименту ми реалізуємо інструмент відтворення трасування, який читатиме файл трасування та надсилатиме запити вводу-виводу на віртуальний диск віртуальної машини. Пропускна здатність вводу-виводу, як показники продуктивності вводу-виводу віртуальної машини, повідомляється інструментом відтворення трасування під час міграції живого сховища. У типовому віртуалізованому середовищі на одному сервері працює кілька віртуальних машин. У наших експериментах на кожному сервері є ще дві запущені віртуальні машини. На одній віртуальній машині працює тест файлового сервера, а на іншій — веб-сервер.

Оцінки на основі трасування. Як зазначалося раніше, DBT є одним із фундаментальних підходів до міграції живого сховища віртуальної машини. Ми інтегруємо WAIO у схему DBT і проводимо експерименти, щоб оцінити покращення продуктивності. У цьому експерименті віртуальна машина налаштована на 2 ГБ пам’яті, один образ віртуального диска на 15 ГБ з драйвером virtio. Режим кешу встановлюється за замовчуванням (наскрізний запис) у програмі QEMU, що вмикає кеш сторінки хоста та вимикає кеш

запису на диск віртуальної машини. Один запасний SSD на вихідному сервері використовується як сурогатний пристрій, і ми встановили швидкість міграції запущеної віртуальної машини на рівні 40 МБ/с. Ми відтворюємо три траси під час міграції живого сховища віртуальної машини та встановлюємо кількість незавершених операцій вводу-виводу як одну. Пропускна здатність вводу/виводу у віртуальній машині повідомляється після завершення міграції.

На рисунку 2.3 показано, що порівняно з DBT WAIO збільшує пропускну здатність до 1,30, 2,61 і 11,10 для трасування Financial1, Financial2 і WebSearch1 відповідно.

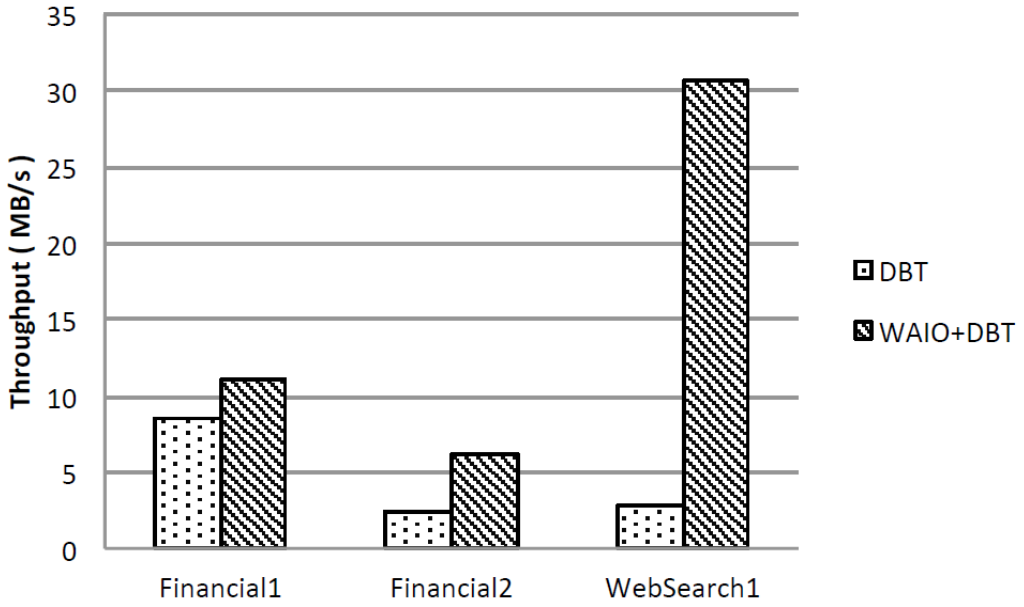


Рисунок 2.3 – Продуктивність VM I0 під час міграції

Причини, чому WAIO досягає значного покращення пропускної здатності вводу-виводу, три. По-перше, більшість запитів вводу-виводу (понад 90% для всіх трьох трасувань) передаються на сторонній сурогатний SSD, тому на них не впливають запити введення-виведення міграції. Середній розмір запиту становить кілька КБ (див. таблицю 2.4) , тоді як розмір блоку становить 1 МБ. Завдяки аутсорсингу одного фрагмента сурогатний SSD може обслуговувати багато вхідних запитів вводу-виводу з високою ймовірністю. Ось чому WAIO може передати таку кількість запитів

вводу-виведення сурогатному SSD. По-друге, сурогатний SSD має кращу продуктивність вводу-виводу, ніж оригінальний жорсткий диск. Навіть якщо ми використовуємо жорсткий диск як сурогатний пристрій, WAIO все ще може певною мірою покращити пропускну здатність. Ми покажемо цей результат у дослідженні чутливості. По-третє, оскільки багато 10 запитів передаються на зовнішній сурогатний SSD, черга 10 на оригінальному пристрої відповідно скорочується, таким чином збільшуючи пропускну здатність решти 10 запитів, які обслуговуються оригінальним пристроєм.

Крім того, WAIO використовує лише 3,99%, 5,57% і 10,01% свого віртуального дискового простору на сурогатному пристрої для трасування Financial 1, Financial і WebSearchl відповідно. Такі накладні витрати на простір для зберігання в сурогатному пристрої зберігання незначні та будуть вивільнені негайно після завершення живої міграції.

На рисунку 2.4 показано порівняння середнього часу відповіді користувача DBT і WAIO кожні 10 секунд.

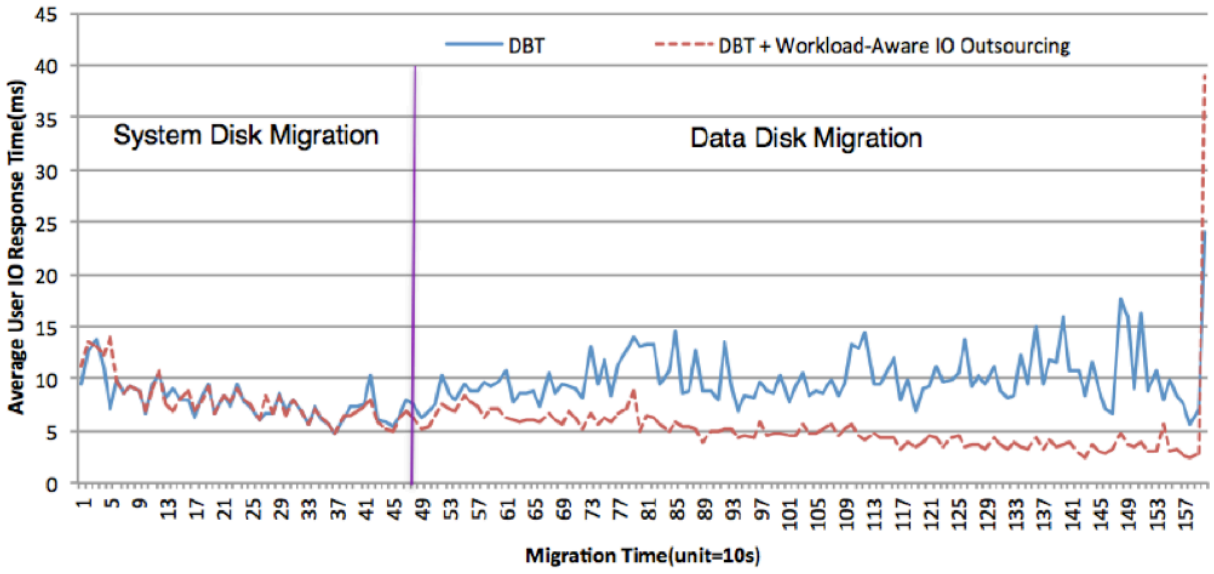


Рисунок 2.4 – Продуктивність віртуальної машини вводу-виводу під час міграції живого сховища віртуальної машини з двома образами віртуальних дисків

У цьому тесті віртуальна машина для міграції має два образи віртуальних дисків: образ системного диска та образ диска даних і ці два образи віртуального диска знаходяться в різні запам'ятовуючі пристрої. Працююча програма у віртуальній машині зберігає дані читання/запису з образів дисків із даними. З малюнка ми бачимо, що продуктивність часу відгуку користувача DBT і WAIO під час міграції системного диска майже однакова. Під час перенесення диска з даними стало зрозуміло, що WAIO значно покращує час відгуку користувача базового DBT, у середньому на 91%.

Швидкість міграції: щоб перевірити чутливість WAIO до швидкості міграції віртуальної машини, ми проводимо експерименти з міграції віртуальної машини на різних швидкостях міграції та вимірюваємо продуктивність віртуальної машини. Пропускна здатність VM I0 представлена на рисунку 2.5.

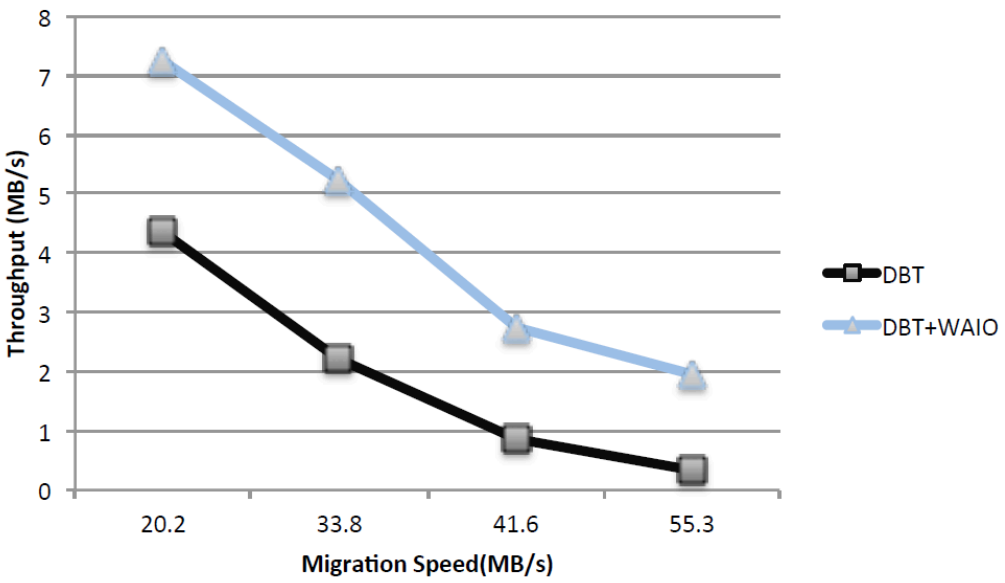


Рисунок 2.5 – Порівняння продуктивності VM I0 за різних швидкостей міграції

Ми робимо три ключові спостереження. По-перше, пропускна здатність VM I0 значно падає зі збільшенням швидкості міграції. Це пов’язано з тим,

що більше запитів на міграцію очікують у черзі вихідного пристрою зберігання даних із вищою швидкістю міграції, забираючи більше пропускної здатності сховища, доступної для запитів VM I0. По-друге, WAIO покращує продуктивність VM I0 у 6,09 разів при різних швидкостях міграції порівняно з DBT. У WAIO, як тільки робочий набір віртуальної машини передається сурогатному пристрою, запити VM I0 в основному обслуговуються сурогатним пристроєм.

Як результат, на їх пропускну здатність менше впливає збільшення запитів на міграцію 10. По-третє, WAIO забезпечує більшу гнучкість інфраструктури хмарних обчислень. Порівняно з DBT, WAIO може або досягти аналогічної пропускної здатності VM 10 (5,25 МБ/с) із вищою швидкістю міграції (41,6 МБ/с над 33,8 МБ/с), або досягти вищої пропускної здатності VM 10 (5,25 МБ над 2,24 МБ/с).) із подібною швидкістю міграції (33,8 МБ/с). Системні адміністратори можуть визначати різні політики для міграції оперативного сховища на основі угоди про рівень обслуговування та вимог до керування системою.

Розмір віртуального дискового образу: щоб оцінити WAIO з різними розмірами віртуального дискового образу, ми створюємо чотири віртуальні машини з різними віртуальними дисковими образами (тобто 15 ГБ, 20 ГБ, 25 ГБ і 30 ГБ), а потім вимірюємо продуктивність VM 10 під час міграції. Крім того, ми масштабуємо адресний простір файлу трасування, щоб охопити адресний простір образу віртуального диска в нашому експерименті. На рисунку 2.6 показано, що пропускна здатність VM 10 як у WAIO, так і в DBT зменшується зі збільшенням розміру образу віртуального диска через збільшення часу на операції пошуку диска. У той же час WAIO перевершує DBT на коефіцієнт до 2,10 з точки зору пропускної здатності VM IO під час міграції. Таке покращення головним чином пояснюється зменшенням перешкод вводу-виводу в оригінальному пристрої. Крім того, оскільки фрагменти даних, переданих стороннім виконавцям, послідовно

розподіляються в сурогатному пристрої, WAIO менш чутливий до розмірів віртуального диска, ніж механізм DBT.

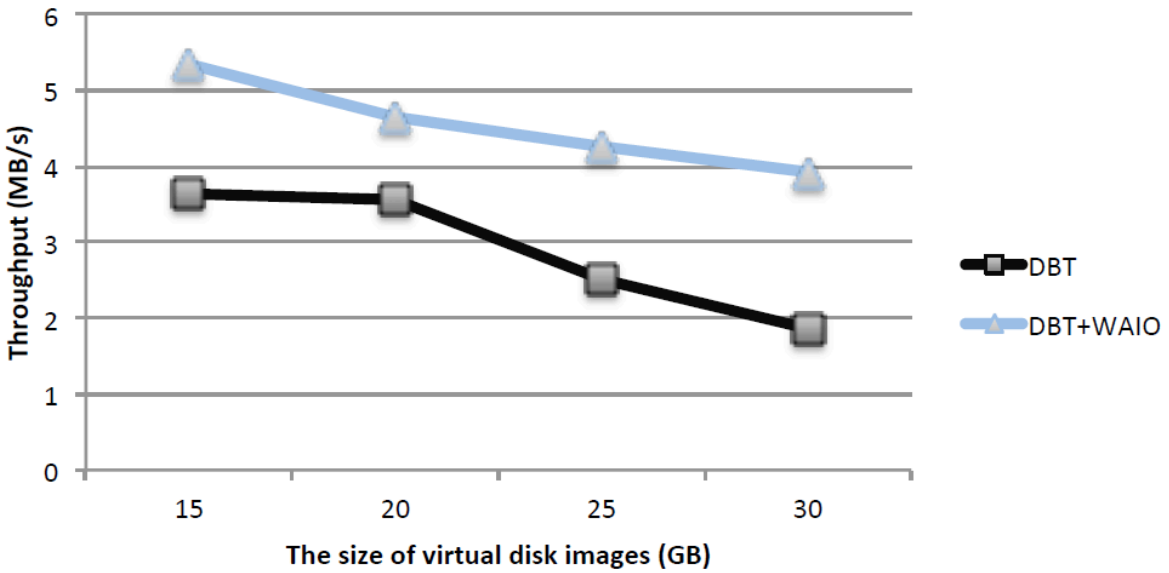


Рисунок 2.6 – Порівняння продуктивності VM I0 із різними розмірами образів віртуальних дисків

Отже, WAIO стає більш ефективним, ніж DBT, у міграції живого сховища віртуальної машини з більшими образами віртуальних дисків.

Висновки до розділу 2

В даному розділі запропоновано підхід та структуру системи міграції урахуванням робочого навантаження (WAIO) для покращення як продуктивності віртуального вводу-виводу так і продуктивності міграції під час міграції живого сховища. Основна ідея полягає в тому, щоб тимчасово захопити робочий набір цільової віртуальної машини та передати дані цього робочого набору на сурогатний пристрій протягом періоду міграції. Завдяки цьому процес введення-виведення віртуальної машини може отримати доступ до сурогатного пристрою під час міграції, тоді як процес введення-виведення міграції більшу частину часу отримує доступ до оригінального диска. У результаті можна значно зменшити взаємодію вводу-виводу між обома

типами процесів вводу-виводу та покращити загальну продуктивність міграції живого сховища. Ця структура ортогональна до існуючих підходів до оптимізації, включаючи підходи DBT [12] і IO Mirroring [11] і цей рівень дозволяє підвищувати продуктивність схем живої міграції віртуальних машин.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ ТА ОПТИМІЗАЦІЯ МОДЕЛІ МІГРАЦІЙНОГО СХОВИЩА ДАНИХ

3.1 Міграція сховища даних на основі концепції образів системи

В цьому розділі ми надаємо необхідну інформацію для дослідження концепції SnapMig, зокрема знімки віртуальної машини, оперативну міграцію образів (знімків) віртуальної машини та резервне копіювання знімків віртуальної машини, що потім допомагає мотивувати це дослідження.

У робочому середовищі віртуальні машини можуть зіткнутися з двома типами збоїв: системним збоєм і втратою даних. Знімки віртуальної машини, які зберігають стан віртуальної машини з попередньою міткою часу, можна використовувати для відкоту віртуальної машини до попереднього узгодженого стану до збою системи. Резервне копіювання миттєвих знімків, яке передає попередні миттєві знімки віртуальної машини на сервери резервного копіювання, регулярно використовується для відновлення станів віртуальної машини, коли гіпервізори стикаються з помилками втрати даних. Обидві ці дві схеми широко використовуються в продуктах хмарної індустрії.

Як зазначено в попередніх розділах, міграція живого сховища віртуальної машини є нетривіальною роботою. Усю інформацію про стан віртуальних машин, які мігрують, включаючи обсяг пам’яті, стани віртуальної мережі, образи віртуальних дисків і інформацію про знімки, необхідно перенести на сервери призначення. Загальний розмір такої інформації зазвичай становить десятки ГБ, найбільшу частину з яких становлять образи віртуальних дисків і знімки, на які припадає приблизно від 80% до 95%. Крім того, оскільки віртуальні машини, що переходять, запускають програми для клієнтів протягом періоду міграції і всі ці оновлення також потрібно перенести на сервери призначення. Що ще

важливіше, коли кілька віртуальних машин мігрують одночасно, це складніше швидко перенести віртуальні машини та забезпечити розумну продуктивність вводу-виводу для всіх віртуальних машин одночасно [51, 13]. Під час міграції живого сховища віртуальної машини на вихідних серверах вводяться додаткові потоки міграції, які споживають значну кількість ресурсів зберігання. Однак загальний ресурс системи зовсім не збільшується. Потоки міграції та потоки VM заважають один одному, а не співпрацюють, що призводить до значного погіршення продуктивності всіх цих потоків. Існує кілька схем міграції віртуальних машин, запропонованих у літературі та галузевих продуктах, таких як Dirty Block Tracking [11], IOMirroring [11], підходи з урахуванням робочого навантаження [12] і підходи до зменшення кількості надлишкових даних [47]. Однак усі вони ігнорують той факт, що базові образи віртуальних машин для міграції та попередні знімки вже знаходяться на серверах резервного копіювання (через звичайні операції резервного копіювання), а сервери резервного копіювання можуть допомогти перенести цю потребу в ресурсах (тобто пропускну здатність мережі та сховища) інформацію на сервери призначення від імені вихідних серверів. У цьому документі ми стверджуємо, що за допомогою резервних знімків віртуальної машини в процесі міграції віртуальної машини можна значно підвищити загальну ефективність міграції.

Дослідження процесів резервного копіювання миттєвих знімків віртуальної машини та його результуючих миттєвих знімків інформації про стан віртуальної машини, доступної на серверах резервного копіювання, ми пропонуємо схему міграції живого сховища віртуальної машини під назвою SnapMig, щоб одночасно покращити продуктивність віртуальної машини та продуктивність міграції. Використовуючи сервери резервного копіювання для передачі базових образів і попередніх знімків віртуальних машин для міграції, вихідним серверам потрібно лише перенести останні зміни стану віртуальної машини на сервери призначення. Отже, серйозні перешкоди між

трафіком вводу-виводу, створеним програмами користувача у віртуальних машинах (включно з віртуальними машинами, що мігрують), у серверів і трафік вводу-виводу, спричинений процесом міграції віртуальної машини, значно зменшується, що призводить до суттєвого підвищення продуктивності як потоків віртуальної машини, так і потоків міграції. Крім того, після міграції останні знімки віртуальної машини, доступні на цільових серверах резервними серверами, дозволяють користувачам вільно повертати свої віртуальні машини до попередніх станів. Крім того, SnapMig ортогональний до існуючих підходів до міграції та розглядається як рівень оптимізації продуктивності для існуючих підходів до міграції. Нарешті, ми спостерігаємо, що переваги продуктивності SnapMig стають більш виразними з одночасним переміщенням живих сховищ кількох віртуальних машин.

Образ (знімок) VM. З метою забезпечення надійності та доступності віртуальних машин, знімки віртуальних машин широко використовуються для відновлення віртуальних машин для клієнтів після збою системи або втрати даних. Зараз більшість сучасних платформ віртуалізації, включаючи VMware, HyperV і KVM, підтримують знімки у своїх продуктах.

Існує два типи знімків: знімок диска та знімок системи. Знімок диска зберігає стан відповідного образу віртуального диска віртуальної машини в певну позначку часу. Отримавши знімок диска, користувач може вільно повернути віртуальну машину до попереднього узгодженого стану, але потрібно перезавантажити віртуальну машину та перезапустити програми.

Процес створення знімка диска складається з двох етапів:

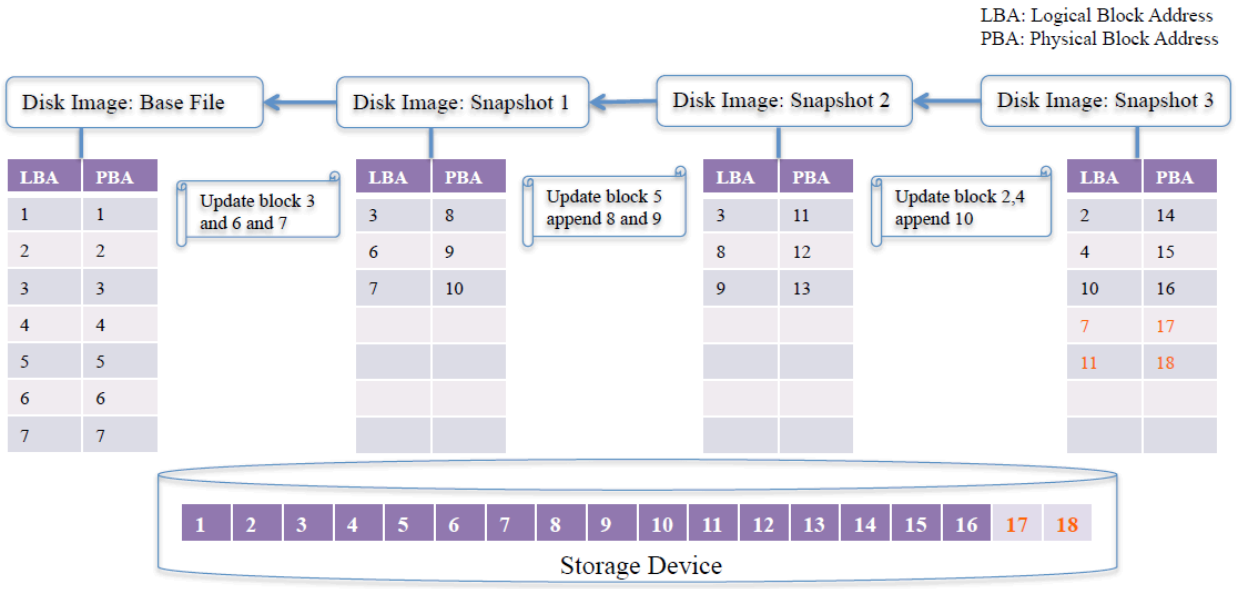
- очищення даних буферного кешу в пам’яті на віртуальні диски;
- створення знімка для кожного віртуального диска.

Таким чином, є незначні витрати на продуктивність для запуску програм у віртуальній машині. Знімок диска в основному використовується для резервного копіювання та аварійного відновлення віртуальних машин.

Знімок системи містить інформацію про стан оперативної пам’яті та інших віртуальних пристроїв віртуальної машини, окрім образів віртуальних дисків. Завдяки підтримці системних знімків користувачі можуть повернути віртуальну машину до попереднього стану роботи. Програми буде відновлено в останній точці виконання, тому немає необхідності перезавантажувати віртуальну машину під час періоду відновлення. Однак для створення знімка системи потрібно більше часу, ніж для знімка диска, оскільки на знімку буде записано стан пам’яті та всіх віртуальних пристроїв. Крім того, запущена віртуальна машина буде приголомшена на етапі створення, що призведе до значного зниження продуктивності запущених програм усередині віртуальної машини [38]. Знімки системи в основному використовуються для виконання ризикованих операцій у середовищі тестування. Ми зосередимося лише на знімках диска, оскільки вони широко використовуються в середовищі віртуальних машин. Крім того, схему, засновану на знімках диска, також легко розширити.

З метою ефективності зберігання кожен знімок віртуальної машини записує лише зміни стану образу віртуальної машини, зроблені з часу останнього знімка в реалізації, припускаючи, що він має доступ до всіх попередніх знімків і базового зображення. Як показано на рисунку 3.1, кожен знімок і базове зображення зберігають таблицю відображення з адреси логічного блоку (LBA - Logical Block Address) на адресу фізичного блоку (PBA - Physical Block Address) як свої метадані. Кожна таблиця (окрім таблиці в базовому зображенні) записує оновлення та нові записи після останнього знімка. Для запитів на оновлення (блок 7) / запис (блок 11) нові записи додаються до таблиці зіставлення знімка 3 (поточний знімок). Для запитів на читання (блок 1 і 6) запитувалися старіші знімки (моментальні знімки 1, 2 і 3) або базове зображення, щоб отримати останню версію блоків даних, до яких здійснюється доступ. Зазвичай робочі сервери зберігають кілька знімків для кожної віртуальної машини, щоб віртуальну машину

можна було швидко відновити до будь-якого зі збережених попередніх знімків у разі збою системи або пошкодження даних.



Requests	Step_1	Step_2	Step_3	Step_4	Step_5	Step_6
Read block 1	Search Snapshot 3	Search Snapshot 2	Search Snapshot 1	Search Base File	Get PBA 1	Read PBA 1 from Storage Device
Read block 6	Search Snapshot 3	Search Snapshot 2	Search Snapshot 1	Get PBA 9	Read PBA 9 from Storage Device	
Update block 7	Allocate block 17 in Storage Device	Write data to physical block 17	Add an entry to Snapshot 3(7->17)			
Append block 11	Allocate block 18 In Storage Device	Write data to physical block 18	Add an entry to Snapshot 3(11->18)			

Рисунок 3.1 – Структура знімків віртуальної машини та робочий процес для запитів на читання/запис

Окрім підтримки знімків у сучасних платформах віртуалізації, основні постачальники сховищ, такі як EMC, також забезпечують оптимізацію сховища для знімків віртуальних машин.

Міграція VM Snapshot. Як зазначено в попередньому підрозділі, існує кілька наявних знімків для кожної віртуальної машини, створених користувачем або системою автоматично, і кожен знімок містить зміни образу віртуальної машини з моменту останнього знімка або базового зображення (якщо це перший знімок). Розмір кожного знімка віртуальної машини суттєво різниться і значною мірою залежить від трафіку запису до запущеної віртуальної машини. Візьмемо приклад використання кластера Aliyun, розмір

кожної віртуальної машини становить 40 ГБ, а кожен знімок віртуальної машини становить у середньому 8 ГБ, що означає, що 20% образів віртуальних дисків було змінено з моменту останнього знімка. Коли мова заходить про живу міграцію віртуальної машини, ці знімки буде або переміщено до місця призначення сервера (під назвою FullMig) або відкинуто на вихідному сервері (під назвою SelectiveMig). Якщо ми перенесемо ці знімки на сервер призначення (FullMig), користувач зможе використовувати ці знімки для відновлення віртуальної машини на сервері призначення, як вони робили до міграції [38]. Однак загальний обсяг переданих даних збільшується, і довший час міграції неминучий. Якщо ми відкинемо знімки та просто перенесемо поточний стан віртуальної машини на сервер призначення (SelectiveMig), загальний обсяг передачі даних буде набагато меншим, ніж у варіанті FullMig. Однак користувачі не можуть повернутися до жодного з попередніх знімків на сервері призначення.

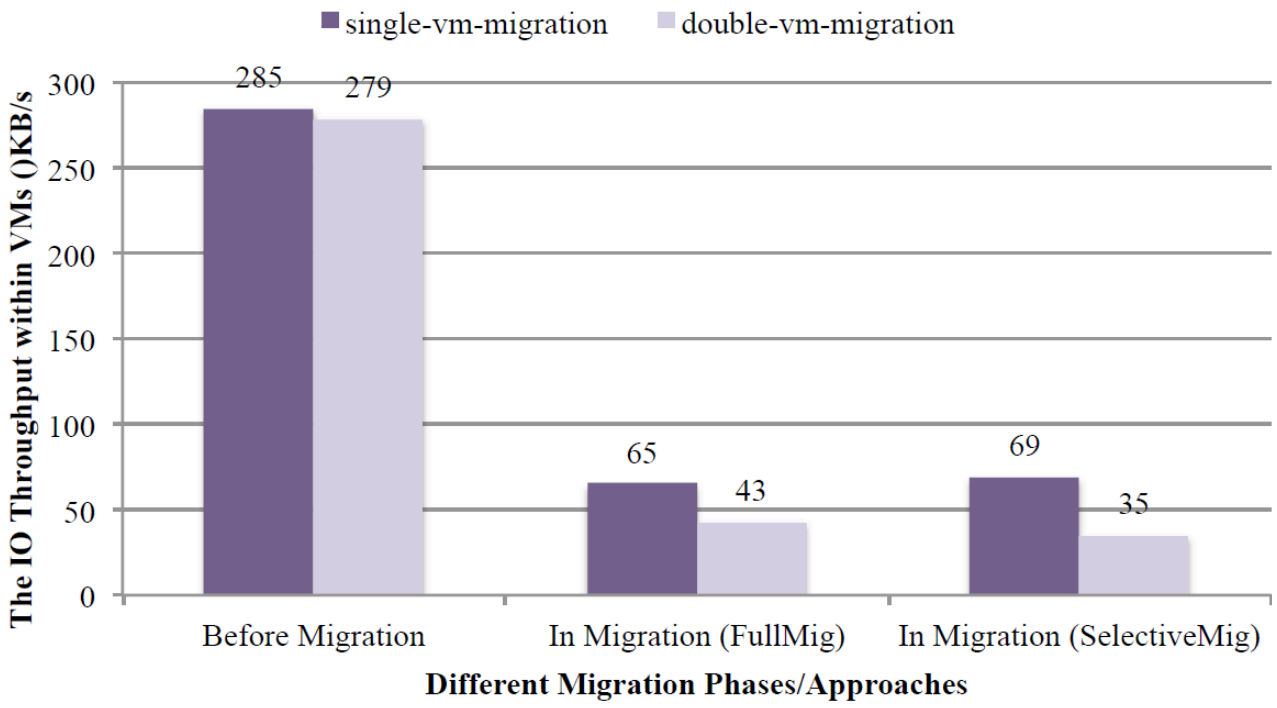
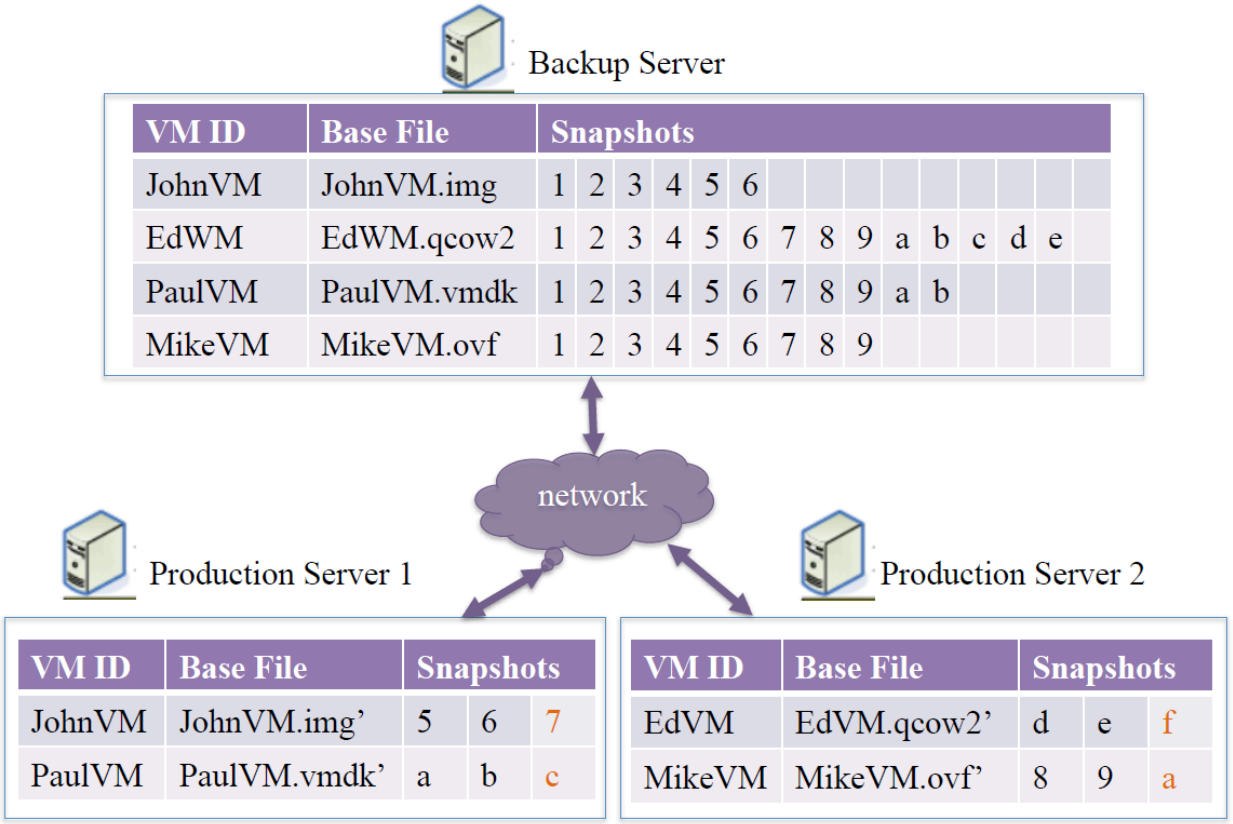


Рисунок 3.2 – Порівняння продуктивності VM IO під час міграції живого сховища

Результати експериментальних досліджень показують, що продуктивність VM падає з 285 = 4,38x (зменшення) до 279 = 7,97x (зменшення) від міграції однієї машини до міграції 2 VM, як показано на рисунку 3.2. У той же час час міграції збільшується більш ніж у 3 рази як для схеми FullMig, так і для схеми SelectiveMig, якщо ми порівнюємо одночасну міграцію 2 віртуальних машин з міграцією однієї віртуальної машини. В ідеалі нам хотілося б рішення, яке могло б забезпечити швидшу міграцію живого сховища віртуальної машини та зберегти всі попередні знімки одночасно. Що ще важливіше, ми хотіли б зменшити трафік для вихідного сервера, оскільки зниження продуктивності мігруючих/розміщених віртуальних машин на вихідному сервері в період міграції має вирішальне значення для загальної продуктивності системи.

Резервне копіювання знімків віртуальної машини. Для цілей надійності та доступності віртуальних машин існує багато механізмів, які використовуються для захисту віртуальних машин, таких як моментальні знімки сховищ даних через системи зберігання, реплікація томів зберігання/LUN, знімки віртуальних машин і реплікація у віртуалізованих програмах. Серед усіх цих механізмів найпоширенішим є резервне копіювання знімків віртуальної машини. Під час виконання створюється серія знімків віртуальної машини, а потім ці знімки регулярно передаються на резервні сервери у вікні резервного копіювання. На серверах резервного копіювання ці знімки будуть додатково оброблені для зменшення споживання місця для зберігання. Наприклад, за допомогою методів дедуплікації надлишкові блоки даних будуть ідентифіковані та видалені. Як показано на рисунку 3.3, для конкретної віртуальної машини (JohnVM) робочий сервер зберігає кілька останніх знімків (знімки 5, 6 і 7) для однієї віртуальної машини, тоді як сервер резервного копіювання зберігає всі попередні знімки віртуальної машини (знімки 1-6) і базовий образ (JohnVM.img) на момент останньої операції резервного копіювання.



VM ID	Base File	Snapshots
JohnVM	JohnVM.img	1 2 3 4 5 6
EdWM	EdWM.qcow2	1 2 3 4 5 6 7 8 9 a b c d e
PaulVM	PaulVM.vmdk	1 2 3 4 5 6 7 8 9 a b
MikeVM	MikeVM.ovf	1 2 3 4 5 6 7 8 9

VM ID	Base File	Snapshots
JohnVM	JohnVM.img’	5 6 7
PaulVM	PaulVM.vmdk’	a b c

VM ID	Base File	Snapshots
EdVM	EdVM.qcow2’	d e f
MikeVM	MikeVM.ovf’	8 9 a

Рисунок 3.3 – Розподіл знімків віртуальної машини між робочими серверами та резервним сервером

Щойно створений знімок (знімок 7) після останньої операції резервного копіювання зберігатиметься лише на робочому сервері. Попередні знімки (1-4) об’єднуються в базове зображення (JohnVM.img') на робочому сервері. У цьому сценарії, якщо робочий сервер стикається з будь-якою проблемою з втратою даних або відключенням живлення, на сервері резервного копіювання все ще є додаткова копія віртуальної машини та її знімки. Інший робочий сервер може відновити віртуальну машину за підтримки резервного сервера. Якщо на робочому сервері не відбувається пошкодження даних, знімки резервного сервера залишатимуться неактивними більшу частину часу.

Під час міграції живого сховища віртуальної машини вихідний сервер буде досить зайнятий, наприклад, виконуючи заплановане завдання технічного обслуговування, запускаючи багато спільно розташованих

віртуальних машин або чекаючи незабаром завершення роботи. Загалом, щоразу, коли запускається міграція живого сховища віртуальної машини, до вихідного сервера буде введено I0-потоки інтенсивної міграції. Щоб досягти задовільної продуктивності VM IO для всіх віртуальних машин, які переміщуються/розміщуються на вихідному сервері, і швидко перенести ВМ на сервери призначення, важливо усунути непотрібний трафік на вихідному сервері. На основі аналізу вище неактивних знімків на сервері резервного копіювання, ми пропонуємо схему SnapMig, яка може досягти цих цілей шляхом використання цих неактивних знімків на серверах резервного копіювання. У схемі SnapMig сервери резервного копіювання передають базові зображення віртуальних машин для міграції та попередні знімки на сервер призначення, тоді як вихідний сервер переносить лише останні зміни до віртуальних машин для міграції.

Перенесення віртуальних машин буде відновлено після успішної реконструкції їхніх станів на сервері призначення. SnapMig має чотири переваги:

1. Краща продуктивність вводу-виводу віртуальної машини під час міграції, оскільки значно зменшений, якщо не повністю виключений трафік міграції, задіяний на вихідному сервері, дозволяє віртуальним машинам, які мігрують/розміщені разом, досягти значно кращої продуктивності вводу-виводу, оскільки якщо більшу частину часу взагалі не було міграції;

2. Коротший час міграції, оскільки сервер(и) резервного копіювання, який не працює більшу частину часу, може передавати зображення віртуальної машини на сервер призначення з набагато вищою швидкістю, ніж будь-який вихідний сервер, який має дуже велике навантаження;

3. Усі попередні знімки доступні на цільових серверах, щоб користувачі могли вільно повертати віртуальні машини до будь-якого з попередніх станів;

4. Покращення продуктивності буде набагато помітнішим у сценаріях, коли кілька віртуальних машин переносяться одночасно.

3.2. Представлення архітектури та робочого процесу системи віртуалізації та міграції даних

В цьому підрозділі ми представляємо дизайн системи SnapMig, представляючи архітектуру SnapMig її ключові функціональні модулі та робочий процес.

Архітектура SnapMig. На рисунку 3.4 показано огляд архітектури розподіленої системи віртуалізації, яка включає три частини: клієнти керування, виробничі сервери та резервні сервери з'єднані високошвидкісною мережею.

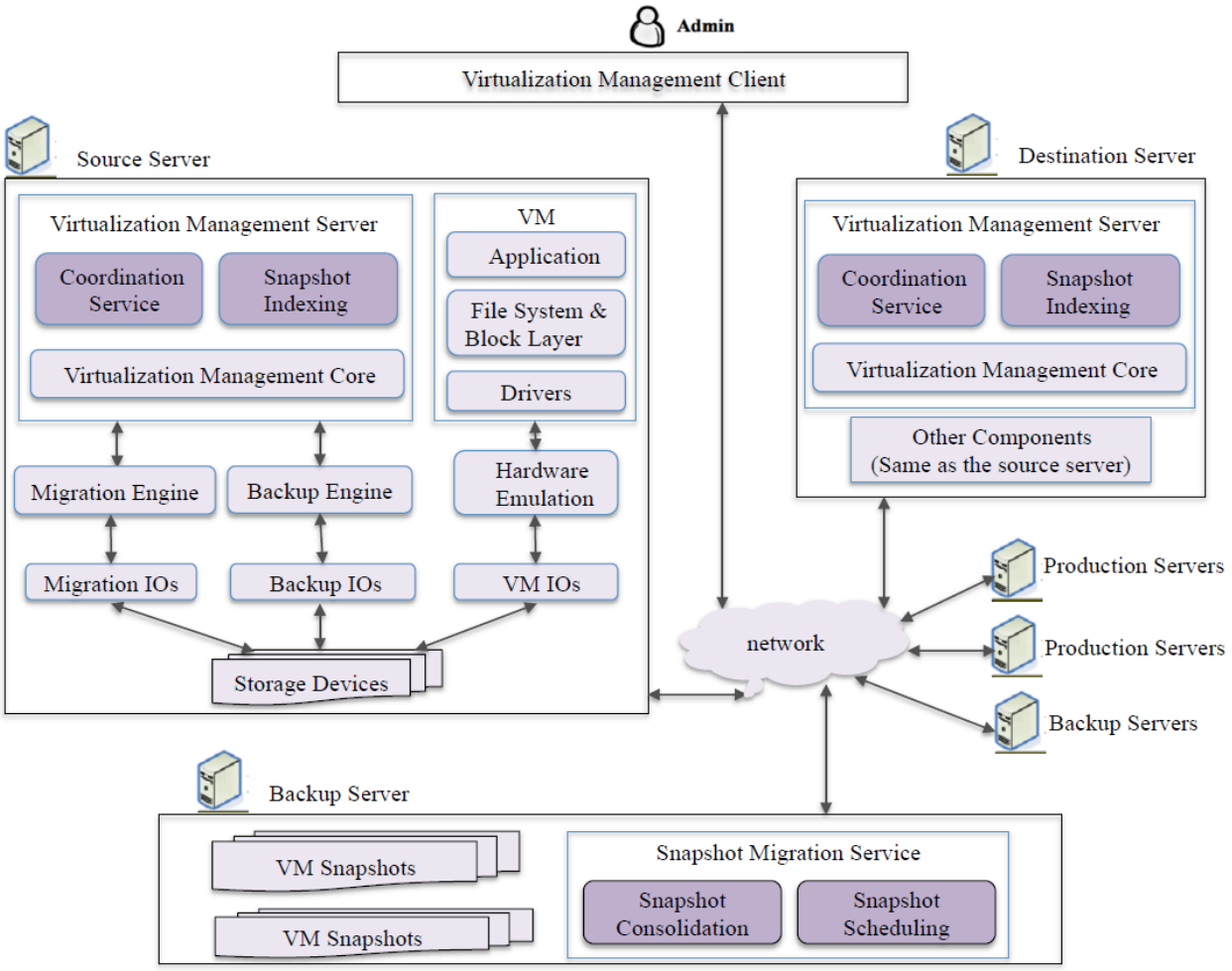


Рисунок 3.4 – Архітектура системи віртуалізації та міграції SnapMig

Клієнти управління надають консоль для системних адміністраторів, щоб виконувати різні завдання керування, наприклад створення віртуальної машини, резервне копіювання знімка віртуальної машини та оперативну міграцію віртуальної машини. Вони можуть перебувати будь-де, якщо доступне мережеве з'єднання з іншими серверами. Виробничі сервери розміщують багато живих віртуальних машин і виконують завдання за запитом клієнтів керування. У робочих серверах є два рівні: сервер керування віртуалізацією (VMS) і модуль гіпервізора. VMS прослуховує вхідні запити від -клієнтів керування та виконує завдання , викликаючи відповідні функції в модулі гіпервізора. За дизайном VMS, наприклад libvirt з відкритим кодом, може підтримувати більшість сучасних гіпервізорів. Сервери резервного копіювання зберігають знімки віртуальної машини з виробничих серверів за допомогою регулярних операцій резервного копіювання, і вони зазвичай оснащені різними функціональними можливостями з метою надійності та ефективності простору, такими як видалення надлишкових даних і стиснення даних.

Схема SnapMig інтегрована в цей кластер і складається з чотирьох функціональних модулів. Сервіс Snapshot Indexing and Coordination Service знаходиться в компоненті VMS (virtualization management server) на робочих серверах, тоді як Snapshot Consolidation і Snapshot Scheduling знаходяться на резервних серверах, як показано на рисунку 3.4. Завдання цих чотирьох модулів детально описані нижче.

Індексування знімків відстежує розподіл і розміщення знімків віртуальної машини серед резервних серверів. Для підвищення надійності може бути кілька копій одного знімка віртуальної машини на кількох резервних серверах, особливо для віртуальних машин із вищими пріоритетами. Крім того, знімки віртуальної машини можна перенести між серверами резервного копіювання з метою балансування навантаження та надійності. Щойно почнеться оперативна міграція віртуальної машини, цей

модуль надсилатиме запит серверам резервного копіювання щодо розповсюдження та розміщення поточних знімків віртуальної машини, що переміщується, і передаватиме їх до модуля Coordination Service.

Coordination Service контролює робочий процес живої міграції віртуальної машини. Завдяки інформації про розповсюдження та розміщення знімків із Snapshot Indexing Coordinate Service вказує відповідним серверам резервного копіювання перенести базовий образ віртуальної машини та попередні знімки на цільовий сервер. Щойно ці сервери резервного копіювання завершать передачу, цей модуль викличе власний механізм міграції на вихідному сервері, який у реальному часі перенесе останні знімки та стан пам’яті на цільовий сервер. Тим часом цей модуль повторно налаштує віртуальну машину та її знімки на сервері призначення. Зрештою, він реєструватиме загальний прогрес міграції, тому міграцію можна буде продовжити у разі збою міграції.

Snapshot Consolidation розроблено для усунення непотрібної передачі даних із серверів резервного копіювання на сервери призначення. Він зливає деякі старші миттєві знімки до базового зображення або окремого миттєвого знімка, перш ніж перенести їх на цільовий сервер. Як показано в таблиці 3.1 , знімки 1-4 непотрібні для цільового сервера, краще було б об’єднати їх із базовим зображенням перед міграцією знімка.

Snapshot Scheduling – це розширена функція, розроблена для подальшого скорочення загального часу міграції. За допомогою аналізу робочого набору віртуальної машини, що переміщується, можна запланувати послідовність блоків у базовому образі та знімках, щоб «гарячі» блоки (тобто часто доступні), які знаходяться в робочому наборі віртуальної машини, переносилися раніше інших. Коли ці гарячі блоки готові на сервері призначення, вихідний сервер починає оперативну міграцію останніх змін стану, а потім віртуальну машину можна перезапустити на сервері призначення, перш ніж усі блоки буде перенесено на сервер призначення.

Конструкція SnapMig досить гнучка. По-перше, він підтримує всі типи сучасних гіпервізорів, оскільки SnapMig спілкується з гіпервізорами через стандартний API управління віртуалізації. По-друге, схема SnapMig є ортогональною до більшості сучасних підходів до міграції живого сховища віртуальної машини, включених у механізм міграції на рисунку 4.4 і вона може доповнювати ці існуючі підходи для подальшого покращення продуктивності міграції живого сховища віртуальної машини. Нарешті, схему SnapMig можна масштабувати відповідно до архітектури кластера. Сервери резервного копіювання та робочі сервери можна вільно додавати в кластер або видаляти з нього під час виконання.

Робочий процес SnapMig. На рисунку 3.5 показано робочий процес міграції живого сховища віртуальної машини в Snap Mig. У цьому прикладі віртуальна машина, що мігрує має один образ диска (з назвою JohnVM.img) і загалом 7 знімків диска (1-7), як зазначено в таблиці 3.1.

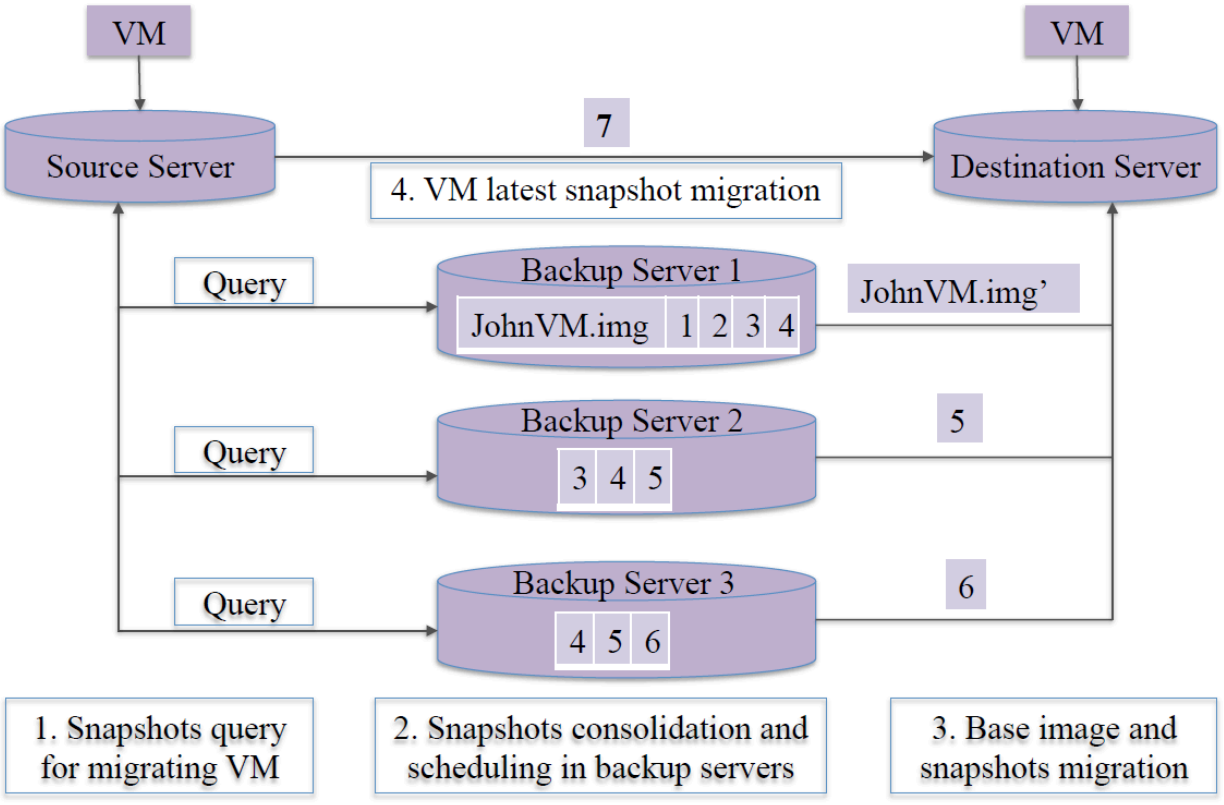


Рисунок 3.5 – Робочий процес схеми SnapMig

Таблиця 3.1

Приклад розподілу та розміщення знімків віртуальної машини під час процесу міграції

Server Name	VM ID	Base File	Snapshots					
Source Server	JohnVM	JohnVM.img'	5	6	7			
Destination Server	JohnVM	JohnVM.img'	5	6	7			
Backup Servers	JohnVM	JohnVM.img	1	2	3	4	5	6

Знімок 7 створюється після останньої щоденної операції резервного копіювання, тому він знаходиться лише на вихідному сервері. Базове зображення та всі інші знімки, 1-6, уже зберігаються на серверах резервного копіювання за допомогою регулярних операцій резервного копіювання. Є кілька копій для базового образу та деяких важливих знімків на серверах резервного копіювання з метою надійності. Щоб заощадити місце для зберігання на вихідному сервері та дозволити користувачам повернутися до останніх знімків, старіші знімки (1-4) об'єднуються з базовим зображенням. Лише нові знімки (5-7) зберігаються на вихідному сервері.

Як вже було зазначено, FullMig і SelectiveMig є найсучаснішими підходами до міграції. У підході FullMig базовий образ віртуальної машини та знімки 5-7 спочатку переносяться на сервер призначення, а потім для міграції викликається звичайний підхід міграції в механізмі міграції, наприклад Dirty Block Tracking або IOMirroring [11] стан у пам'яті та останні зміни стану віртуальної машини. На відміну від підходу FullMig, SelectiveMig переносить лише останній образ віртуального диска на цільовий сервер і відкидає всі наявні знімки, наприклад знімки 5-7. Наприклад, якщо певний блок даних оновлено як на знімку 5 так і на знімку 7, SelectiveMig переносить лише останню версію цього блоку даних (зі знімка 7).

Робочий процес схеми SnapMig наступний. На початку міграції Служба координації запитує індексування знімків для останнього розподілу та розміщення знімків віртуальної машини, що мігрує. Тоді він повідомить

відповідні сервери резервного копіювання, щоб розпочати міграцію базового зображення та знімків. Ці сервери резервного копіювання спочатку об’єднають усі непотрібні знімки та розпочнуть міграцію на сервер призначення. Коли базове зображення та знімки стануть доступними на цільовому сервері, вихідний сервер перенесе останні зміни стану та стан у пам’яті на цільовий. На кінцевому сервері віртуальну машину буде відновлено.

Порівняно з FullMig і SelectiveMig, SnapMig забезпечує незначний міграційний трафік на вихідному сервері, включаючи лише запити на читання системи зберігання та мережеву передачу. Наприклад, якщо певний блок даних оновлено в базове зображення та знімки 5-6, SnapMig не потрібно переносити цей блок даних до місця призначення, оскільки цей блок даних буде переміщено резервними серверами. Таке значне усунення трафіку на вихідному сервері покращить загальну продуктивність системи в кількох аспектах: скорочення часу міграції, краща продуктивність віртуальної машини та краща міграція кількох віртуальних машин.

3.3. Підхід оптимізації міграції сховища віртуальної машини за допомогою методу послідовного вводу-виводу

В цьому підрозділі ми надаємо необхідну інформацію для дослідження IOFollow, включаючи властивість послідовного вводу-виводу та модель потоків.

Властивість послідовного введення-виведення. Властивість послідовного вводу-виводу була однією з найбільш фундаментальних концепцій у галузі системних досліджень, головним чином через невідповідність продуктивності між послідовним і випадковим вводом-виводом у системах зберігання даних. За останні кілька десятиліть пропускна здатність передачі даних значно зросла через більшу щільність

бітів на поверхні дисковода. Однак витрати на пошук і затримку обертання зменшуються повільно, оскільки набагато складніше прискорити механічні рухи головки диска та швидкість обертання пластин. Таким чином, продуктивність послідовного вводу-виводу набагато краща, ніж у випадкового вводу-виводу з частими пошуками та обертаннями диска.

Властивість послідовності IO визначається багатьма метриками як просторового, так і тимчасового вимірів.

Просторовий вимір:

- Послідовні адреси: різниця між адресами логічних блоків (LBA) послідовних запитів введення-виведення знаходиться в межах попередньо визначеного порогу. Далі його можна класифікувати як суворо послідовний (відсутність розриву між LBA послідовних запитів введення-виведення) та поступовий доступ (обмежений розрив між LBA послідовних запитів введення).
- Послідовні доступні байти: середній розмір даних для читання або запису в запиті введення-виведення.

Тимчасовий вимір:

- Interleaved Streams: об’єднання запитів введення-виведення від кількох потоків, програм або віртуальних машин. На властивість послідовності (Sequential) IO для окремих потоків IO може впливати чергування з іншими потоками IO. Наприклад, два послідовних запити введення-виведення (запит 1 і 2) із послідовними адресами з потоку A можуть мати низьку продуктивність введення-виведення, оскільки бере участь довгий рух головки диска, під час якого диск обслуговує інший запит вводу-виводу (запит 3) із потоку B і LBA. запиту 3 знаходиться далеко від LBA запиту 1 або 2.
- Inter-arrival Time: часовий інтервал між послідовними запитами IO. Якщо між двома запитами вводу-виводу є тривалий час очікування, між

ними можуть надходити інші фонові запити вводу-виводу, що вплине на властивість послідовного введення-виведення вихідного потоку вводу-виводу.

Чергування кількох потоків вводу-виводу не лише вплине на властивість послідовного вводу-виводу, але й значно знизить продуктивність вводу-виводу для кожного потоку-учасника. Експериментальні результати в [9] вказують на те, що потік вводу-виводу з випадковим записом є руйнівним для всіх інших видів потоків вводу-виводу з чергуванням, таких як послідовний потік читання / запису. Продуктивність самого потоку вводу-виводу з випадковим записом також значно погіршиться. Більше випадків використання деградації продуктивності для кількох перемежовуваних потоків вводу-виводу представлено в роботі [9].

Враховуючи характер характеристик доступу до диска та властивість послідовного введення-виведення між вихідними потоками вводу-виводу, сучасні файлові системи, як локальні так і розподілені файлові системи, покращують продуктивність вводу-виводу агресивно надсилаючи послідовні запити вводу-виводу на основні диски. Наприклад, у файловій системі на основі журналу достатня кількість оновлень буферизується в пам’яті перед тим, як вони надсилаються на диски як запит на великий послідовний сегмент, так що пропускна здатність диска значно покращується порівняно з окремими невеликими випадковими записами. запити. Подібним чином обслуговуються запити на читання, які зчитують увесь сегмент із дисків одночасно. У файловій системі Google (GFS) мінімальний розмір кожного запиту за замовчуванням становить 64 КБ, тому висока пропускна здатність дисків може бути досягнута за допомогою послідовних запитів вводу-виводу. На жаль, файлові системи можуть лише впливати, але не визначати властивість послідовного вводу-виводу потоків, створених самими програмами користувача. Є багато методів оптимізації, щоб покращити властивість послідовності IO шляхом використання семантичних підказок.

Модель потоків у віртуалізованих системах. У віртуалізованому середовищі потік вводу-виводу на вході системи зберігання є багаторівневим чергуванням окремих потоків вводу-виводу: спочатку створюється потік вводу-виводу на рівні програми з чергуванням кількох потоків вводу-виводу на рівні потоку в одній програмі.

По-друге, для кожної запущеної віртуальної машини створюється один потік вводу-виводу на рівні віртуальної машини з чергуванням кількох потоків вводу-виводу на рівні програми та потоку вводу-виводу операційної системи віртуальної машини. Нарешті, чергування всіх потоків вводу-виводу на рівні віртуальної машини та потоку вводу-виводу гіпервізора стане остаточним потоком вводу-виводу для системи зберігання.

Враховуючи усі показники, пов’язані з властивістю Sequential I0, VM-level I0 Streams (називаються VM IO) здебільшого визначаються програмами користувача та гостьовими операційними системами. На рисунку 3.6 показано IO-потокову модель систем віртуалізації.

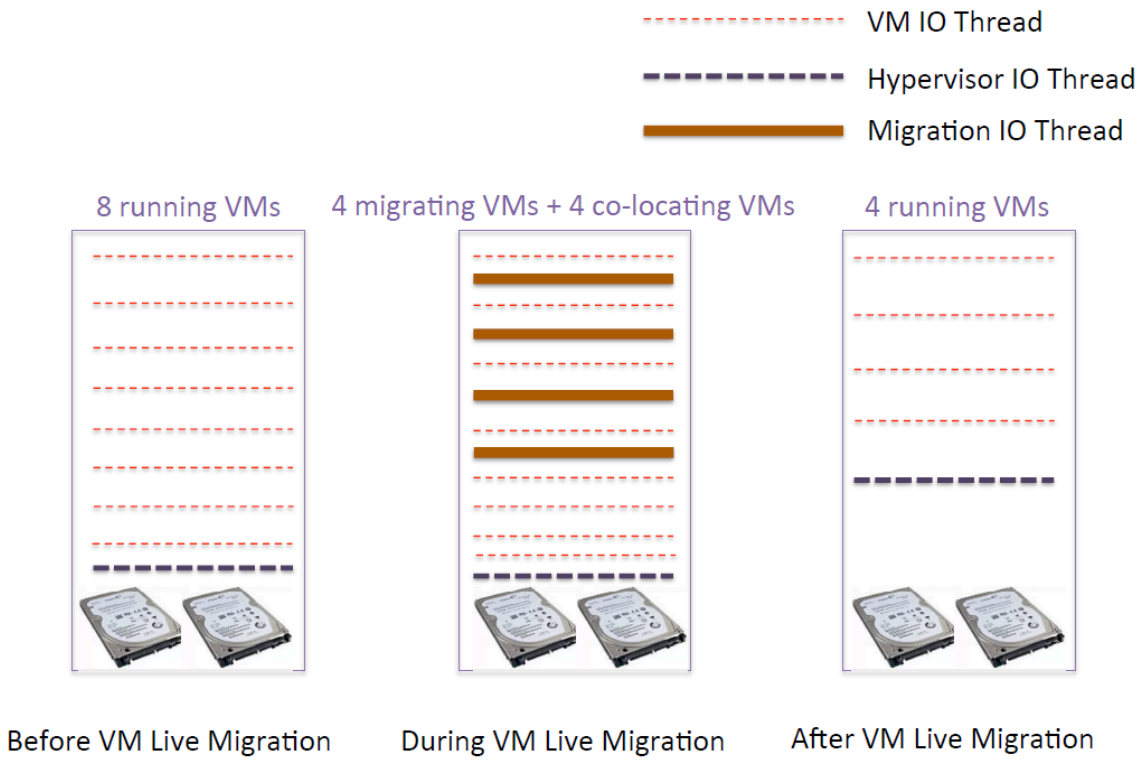


Рисунок 3.6 – IO-потокова модель віртуалізованої системи на різних етапах міграції живого сховища віртуальної машини

Коли справа доходить до міграції живого сховища віртуальної машини, один потік міграції буде призначено для кожної міграційної віртуальної машини, і він перенесе всю інформацію про стан міграційної віртуальної машини з вихідного сервера на сервер призначення. У більшості випадків потік міграції зчитує дані образів віртуальних дисків віртуальної машини від початку до кінця, а також синхронізує оновлені дані з сервером призначення.

З точки зору властивості Sequential IO, потік міграції є ідеальним послідовним робочим навантаженням, оскільки він читатиме дані в просторі образів віртуальних дисків виключно послідовно та заслуговує на високу продуктивність. На жаль, це не так.

Саме чергування з іншими потоками вводу-виводу, включаючи вводи-виводи VM та інші введення-виводи міграції, підриває властивість послідовного вводу-виводу потоків міграції. Головка диска має здійснювати пошук і повертатися в інші місця, щоб обслуговувати інші запити введення-виведення між двома послідовними запитами на читання для потоку міграції. Після обслуговування інших запитів вводу-виводу головка диска має шукати та повертатися назад до сусіда попереднього розташування, щоб обслуговувати наступний запит на читання для потоку міграції. Таким чином, продуктивність вводу-виводу всіх залучених потоків вводу-виводу значно погіршується, тому неможливо уникнути більш тривалого часу міграції віртуальної машини та низької продуктивності віртуальної машини.
Під час міграції живого сховища віртуальної машини гіпервізори вже перевантажені через додаткові потоки міграції, які потребують пропускної здатності. Подальше руйнування властивості Sequential IO потоків міграції призведе до збільшення ваги механізму міграції, тому буде набагато складніше швидко перенести віртуальні машини та забезпечити SLA для продуктивності вводу-виводу всіх залучених віртуальних машин.

Проаналізувавши дослідницькі роботи та внутрішній механізм робочого процесу міграції, ми помітили, що:

1. Окремі запити вводу-виводу від віртуальної машини, що мігрує, генеруються програмами, тому ми маємо обмежені можливості маніпулювати потоком вводу-виводу віртуальної машини. щоб покращити продуктивність VM IO.

2. Лише загальний час міграції та точність передачі стану віртуальної машини мають значення для потоків міграції, тоді як розмір індивідуального запиту, початкова адреса кожного запиту або послідовність запитів міграції введення-виведення не мають значення.

Таким чином, механізм міграції має повну гнучкість для створення різних типів запитів міграції вводу-виведення, доки вся інформація про стан віртуальної машини, що мігрує, належним чином надходить до пункту призначення протягом розумного вікна часу міграції.

На основі цього пропонується схему міграції живого сховища віртуальної машини під назвою IOFollow, яка покращить як продуктивність вводу-виводу віртуальної машини, так і ефективність міграції шляхом створення та планування послідовності міграції відповідно до потоку вводу-виводу запитів віртуальної машини.

По суті, ми виберемо наступного кандидата на міграцію блоку на основі двох критеріїв:

- цей блок даних не було перенесено до місця призначення;
- адреса цього блоку даних близька до поточної області доступу або положення головки диска.

Таким чином ми очікуємо позбутися непотрібних рухів голівки диска, так що потік запитів вводу-виведення з чергуванням на воротах системи зберігання стає більш послідовним. Продуктивність усіх потоків вводу-виводу віртуальної машини, потоку вводу-виводу гіпервізора та потоків вводу-виводу міграції значно зростає під час процесу міграції живого сховища віртуальної машини. Крім того, ми можемо вибірково кешувати в пам’яті блоки даних, зчитані потоками міграції із системи зберігання, і

використовувати ці дані для обслуговування прогнозованих вхідних запитів VM IO, так що цим запитам VM IO взагалі не потрібно буде торкатися системи зберігання. Таким чином, можна значно покращити як продуктивність вводу-виводу віртуальної машини, так і продуктивність міграції.

На перший погляд, роботи IOFollow, WAIO та Zheng [12] покращують продуктивність міграції живого сховища віртуальної машини, використовуючи характеристики робочого навантаження всередині віртуальних машин, але їхні фундаментальні ідеї відрізняються одна від одної. У системі WAIO робочий набір віртуальної машини ідентифікується та тимчасово передається на інший сурогатний пристрій зберігання, так що потік віртуальної машини обслуговується сурогатним пристроєм, а ¬потік міграції отримує доступ до вихідного пристрою зберігання більшу частину часу. Перешкоди вводу-виводу між цими двома типами потоків усуваються шляхом ізоляції цих потоків від різних пристроїв зберігання. У роботі [12] мета полягає в тому, щоб зменшити кількість повторних передач часто оновлюваних блоків даних шляхом міграції рідко оновлюваних блоків даних.

Іншими словами, потоки віртуальної машини матимуть доступ до гарячих зон диска (робочий набір віртуальної машини), тоді як потоки міграції отримають доступ до холодних зон диска (блоки даних, до яких рідко звертаються). Ця схема зменшить загальну передачу даних ціною посилення перешкод вводу-виводу між потоками віртуальної машини та потоками міграції, оскільки пристрої зберігання повинні шукати туди-сюди, щоб обслуговувати чергувані запити вводу-виведення як у гарячих зонах, так і в холодних зонах. Нарешті, IOFollow прагне покращити продуктивність міграції віртуальної машини, дозволяючи потокам міграції та потокам віртуальної машини співпрацювати один з одним. Крім того, IOFollow покращує коефіцієнт звернень до кешу для потоків віртуальних машин за рахунок інтелектуального кешування блоків даних у пам’яті.

Образи віртуальних дисків на основі файлів широко застосовуються у віртуалізованому середовищі. З точки зору віртуальної машини, образ віртуального диска такий самий, як і фізичний диск, який підтримує всі типи API рівня блоків, наприклад команди ISCSI. Він сумісний із основними гостьовими операційними системами, такими як Windows і Linux. З точки зору системи зберігання, образи віртуальних дисків — це не що інше, як звичайні великі файли, які можуть зберігатися в більшості файлових систем. Таким чином, усі запити на введення-виведення від користувальницьких програм запущеної віртуальної машини стають запитами на введення-виведення для основного великого файлу, який містить образ віртуального диска. Так само потік міграції зчитує цей великий файл, щоб перенести інформацію про стан сховища віртуальної машини. Продуктивність образів віртуальних дисків має вирішальне значення для продуктивності працюючої віртуальної машини та гнучкості міграції. Щоб підвищити продуктивність образів віртуальних дисків, було винайдено декілька спеціалізованих файлових систем/систем зберігання лише для образів віртуальних дисків, таких як VMWare VMFS і Tintri VMStore.

На рисунку 3.7 показано огляд архітектури системи IOFollow. IOFollow — це простий модуль, який можна інтегрувати в будь-який сучасний гіпервізор, а його параметри, такі як розмір фрагмента міграції, алгоритм заміни кешу блоків, можна налаштувати під різні навантаження програми. Для завдань міграції живого сховища віртуальної машини лише сервер на стороні джерела має включити модуль IOFollow, тоді як сервер на стороні призначення залишається недоторканим. IOFollow — це рівень підвищення продуктивності, який можна поєднати зі звичайним підходом до живої міграції, включаючи Dirty Block Tracking і IO Mirroring, для подальшого покращення продуктивності вводу-виводу віртуальної машини і продуктивність міграції. IOFollow можна застосовувати для живої міграції

віртуальних машин між серверами в одному кластері або в різних центрах обробки даних у всьому світі.

У поточних віртуалізованих системах для кожної запущеної віртуальної машини є кеш-пам'ять двох рівнів: кеш запису на гостьовому диску (у межах віртуальної машини) і кеш-пам'ять сторінок хоста (у гіпервізору).

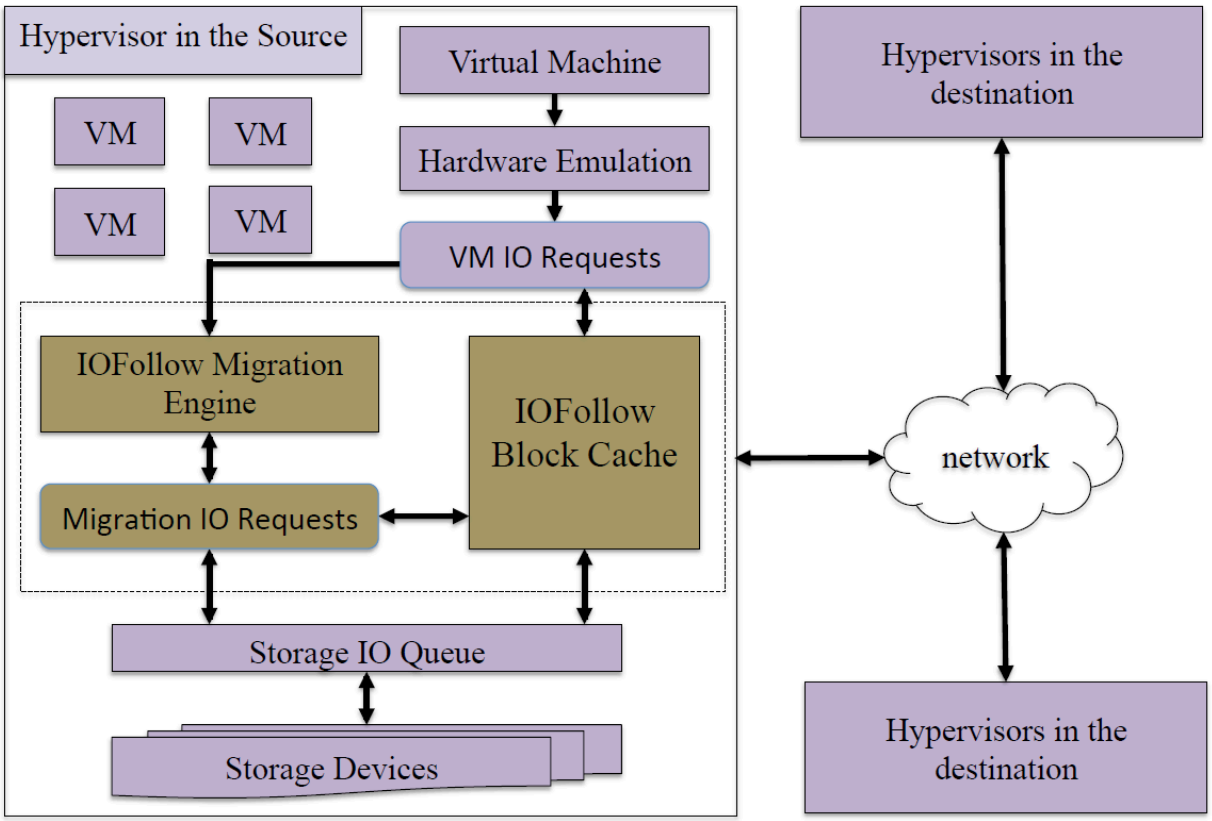


Рисунок 3.7 – Архітектура системи IOFollow

Відповідно до характеристик різних програм користувачі можуть увімкнути або вимкнути один із цих двох рівнів кешу або обидва перед створенням віртуальної машини або протягом життєвого циклу запущеної віртуальної машини. Під час нормального періоду виконання лише потік вводу-виводу віртуальної машини із запущених програм і гостьової операційної системи відвідує ці дворівневі кеш-системи, що може значно заощадити доступ до сховища для програм або гостьової операційної системи. Коли починається міграція живого сховища, надходить додатковий

потік запитів вводу-виводу з потоку міграції, який займатиме значну частину виділеного кешу сторінки хоста для віртуальної машини, що мігрує. Оскільки традиційний кеш не може підвищити продуктивність для потокових запитів вводу/виводу (наприклад, онлайн-програми для потокового відео), кращий дизайн системи кешу необхідний для загальної продуктивності системи.

IOFollow містить два основні компоненти: Migration Blocks Scheduler і Migration-Aware Block Cache Manager. Планувальник блоків міграції проаналізує трафік запитів на введення-виведення віртуальної машини, визначить поточну зону доступу до віртуальної машини, передбачить пізнішу область доступу введення-виведення, а потім вибере правильний фрагмент даних для міграції. Після того, як фрагмент даних буде переміщено на цільовий сервер, він буде переданий диспетчеру кешу блоків з підтримкою міграції.

Вибір фрагментів даних міграції базується на двох частинах:

- коротший час пошуку системи зберігання для запиту міграції IO;
- цей блок даних, отриманий потоком міграції, може обслуговувати пізніші запити вводу-виводу віртуальної машини з більшою ймовірністю.

Менеджер кешу блоків із підтримкою міграції керує ресурсом пам’яті та інтелектуально кешує блоки даних для наступних запитів вводу-виводу віртуальної машини. Оскільки потік міграції сканує образи віртуальних дисків лише один раз, він не матиме доступу до тих самих блоків даних більше одного разу, за винятком оновлених блоків даних. Однак доступ до цих блоків даних може здійснюватися за допомогою запитів VM IO. Таким чином, кешуючи блоки даних гарячої міграції в пам’яті, багато вхідних запитів VM IO можуть обслуговуватися диспетчером кешу блоків з підтримкою міграції безпосередньо в пам’яті. Коли кеш блоків заповниться, алгоритм заміни кешу вживе заходів і холодні блоки даних буде видалено.

Системні збої або збій міграції можуть бути спричинені багатьма факторами, такими як апаратні/програмні помилки, збої живлення,

неправильні операції або атаки ззовні. Система IOFollow охоплює ці збої за допомогою проактивного дизайну для узгодженості та надійності системи. Зокрема, IOFollow зберігає ключові структури даних в енергонезалежній RAM (NVRAM), щоб запобігти раптовій втраті живлення або збою системи. Оскільки розмір цих структур даних, як правило, дуже малий, це не становить значних витрат для системи. Дані в кеші блоків ми можемо зберігати безпосередньо в DRAM, оскільки в системі зберігання завжди є ще одна копія.

Висновки до розділу 3

Отже, в цьому розділі запропоновано концепцію міграції сховища даних на основі концепції образів системи SnapMig. На основі досліджень процесу резервного копіювання миттєвих знімків віртуальної машини та його отриманих миттєвих знімків інформації про стан віртуальної машини, доступної на серверах резервного копіювання, запропоновано схему міграції живого сховища віртуальної машини під назвою SnapMig, щоб покращити як продуктивність віртуальної машини, так і ефективність міграції. Також представлено систему IOFollow, розроблену для підвищення продуктивності міграції живого сховища віртуальної машини. На основі фактів, що порядок послідовний або випадковий блоків, які переносяться, не впливає на продуктивність, оскільки віртуальні диски віртуальної машини, що переноситься, не можна відновити, доки всі блоки даних не будуть доступні на сервері призначення, тоді як порядок запитів VM IO працюють, оскільки кожен запит IO споживається програмами під час виконання. Для цього система IOFollow планує послідовність міграції блоків відповідно до запитів VM IO, щоб можна було значно зменшити трудомісткі переміщення головки диска.

ВИСНОВКИ

В кваліфікаційній роботі виконано оптимізацію моделей та методів хмарної міграції даних в рішеннях на основі віртуальних машин Для цього було зосереджено дослідження на продуктивності міграції живого сховища віртуальної машини та визначено фундаментальну та наукову проблему перешкод вводу-виводу.

В роботі представлено дослідження технології WAIO для оптимізації міграції живого сховища з урахуванням робочого навантаження. Звичайні підходи до міграції, такі як Dirty Block Tracking (DBT) і IO Mirroring, не вирішують проблеми взаємодії вводу-виводу між запитами віртуальної машини та запитами введення-виведення міграції під час періоду міграції, що погіршує як продуктивність введення-виведення віртуальної машини, так і продуктивність міграції. В роботі пропонується схему вводу-виводу з урахуванням робочого навантаження (WAIO), щоб підвищити ефективність міграції живого сховища віртуальної машини. WAIO фактично передає робочий набір віртуальної машини сурогатному пристрою під час міграції та створює окремий шлях вводу-виводу для обслуговування запитів вводу-виведення віртуальної машини. Завдяки аутсорсингу запитів VM IO з оригінального сховища на сурогатний пристрій процес міграції VM Live Storage можна виконати на оригінальному сховищі без перешкод, тоді як зовнішні запити VM IO обслуговуються окремо і, отже, набагато швидше.

Запропоновано підхід SnapMig - міграція живого сховища віртуальної машини на основі знімків. Більшість існуючих підходів до міграції віртуальної машини не можуть вирішити проблему перешкод вводу-виводу, оскільки вони спричиняють значне додаткове сховище та мережевий трафік до вихідного сервера, який уже сильно завантажений або запланований для оновлення чи ремонту. У результаті як продуктивність віртуальної машини,

яку сприймає програма/користувач, так і продуктивність міграції значно погіршуються. Схема SnapMig дозволяє підвищити ефективність міграції живого сховища віртуальної машини та усунути її вплив на продуктивність додатків користувачів на вихідному сервері шляхом ефективного використання наявних знімків віртуальної машини на резервних серверах. Делегуючи сервери резервного копіювання для передачі базового образу віртуальної машини та знімків на цільовий сервер, вихідному серверу потрібно лише перенести останні зміни стану на цільовий сервер, що водночас покращує продуктивність віртуальної машини, скорочує час міграції та ефективніше переносить кілька віртуальних машин.

Представлений в роботі підхід IO Following дозволяє виконувати оптимізацію міграції живого сховища віртуальної машини. Поточні підходи до міграції живого сховища віртуальної машини ігнорують властивість послідовного введення-виведення потоків вводу-виведення, що чергуються, на вході сервера зберігання, просто послідовно переміщуючи образи віртуальних дисків віртуальної машини, незалежно від одночасних потоків вводу-виводу віртуальної машини та інших потоків міграції. Таким чином, вводиться багато непотрібних операцій пошуку головки диска та обертання, що забирає багато часу, що погіршує як продуктивність вводу-виводу віртуальної машини, так і продуктивність міграції. Система IOFollow може покращити як продуктивність вводу-виводу віртуальної машини, так і ефективність міграції шляхом створення та планування послідовності міграції відповідно до потоку вводу-виводу запитів віртуальної машини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. Angel, H. Ballani, T. Karagiannis, G. O'Shea, and E. Thereska, "End-toend performance isolation through virtual datacenters," in Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation. USENIX Association, , pp. 233–248. 2014.

2. P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu et al., "Ananta: Cloud scale load balancing," in ACM SIGCOMM Computer Communication Review, vol. 43, no. 4. ACM, , pp. 207–218, 2013.

3. R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers," in Autonomic Computing, ICAC'07. Fourth International Conference on. IEEE, 2007, pp. 5–5. 2007.

4. X. Lin, Y. Mao, F. Li, and R. Ricci, "Towards fair sharing of block storage in a multi-tenant cloud," in Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing. USENIX Association, , pp. 15–15. 2012.

5. J. Zheng, T. S. E. Ng, and K. Sripanidkulchai, "Workload-aware live storage migration for clouds," in ACM Sigplan Notices, vol. 46, no. 7. ACM, , pp. 133–144, 2011.

6. J. Zheng, T. S. E. Ng, K. Sripanidkulchai, and Z. Liu, "Comma: Coordinating the migration of multi-tier applications," in ACM SIGPLAN Notices, vol. 49, no. 7. ACM, , pp. 153–164, 2014.

7. W. Zhang, H. Tang, H. Jiang, T. Yang, X. Li, and Y. Zeng, "Multi-level selective deduplication for vm snapshots in cloud storage," in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, , pp. 550– 557, 2012.

8. T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, “Black-box and gray-box strategies for virtual machine migration.” in NSDI, vol. 7, , pp. 17–17, 2007.

9. P. Riteau, C. Morin, and T. Priol, “Shrinker: Improving live migration of virtual clusters over wans with distributed data deduplication and content-based addressing,” in Euro-Par 2011 Parallel Processing. Springer, , pp. 431– 442, 2011.

10. R. Zhou, F. Liu, C. Li, and T. Li, “Optimizing virtual machine live storage migration in heterogeneous storage environment,” in ACM SIGPLAN Notices, vol. 48, no. 7. ACM, pp. 73–84. 2013.

11. S. Kannan, A. Gavrilovska, and K. Schwan, “pvm: persistent virtual memory for efficient capacity scaling and object storage,” in Proceedings of the Eleventh European Conference on Computer Systems. ACM, p. 13. 2016.

12. N. Li, H. Jiang, D. Feng, and Z. Shi, “Pslo: enforcing the x th percentile latency and throughput slos for consolidated vm storage,” in Proceedings of the Eleventh European Conference on Computer Systems. ACM, p. 28. 2016.

13. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live migration of virtual machines,” in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, pp. 273–286, 2005.

14. M. Nelson, B.-H. Lim, G. Hutchins et al., “Fast transparent migration for virtual machines.” in USENIX Annual technical conference, general track, pp. 391–394. 2005.

15. S. Nathan, U. Bellur, and P. Kulkarni, “On selecting the right optimizations for virtual machine migration,” in Proceedings of the12th ACM SIGPLAN/SIGOPS

16. International Conference on Virtual Execution Environments. ACM, pp. 37–49. 2016.

17. C. Jo, E. Gustafsson, J. Son, and B. Egger, “Efficient live migration of virtual machines using shared storage,” in ACM Sigplan Notices, vol. 48, no. 7. ACM, pp. 41–50. 2013.

18. R. Birke, M. Bjoerkqvist, L. Y. Chen, E. Smirni, and T. Engbersen, “(big) data in a virtualized world: volume, velocity, and variety in cloud datacenters,” in

19. Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14), pp. 177–189, 2014.

20. H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, “Live virtual machine migration with adaptive, memory compression,” in Cluster Computing and Workshops, 2009. CLUSTER’09. IEEE International Conference on. IEEE, pp. 1–10, 2009.

21. P. Sva¨rd, B. Hudzia, J. Tordsson, and E. Elmroth, “Evaluation of delta compression techniques for efficient live migration of large virtual machines,” ACM Sigplan Notices, vol. 46, no. 7, pp. 111–120, 2011.

22. H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, “Live migration of virtual machine based on full system trace and replay,” in Proceedings of the 18th ACM international symposium on High performance distributed computing. ACM, pp. 101–110. 2009.

23. M. R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning,” in Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, pp. 51–60, 2009.

24. K.-Y. Hou, K. G. Shin, and J.-L. Sung, “Application-assisted live migration of virtual machines with java applications,” in Proceedings of the Tenth European Conference on Computer Systems. ACM, p. 15, 2015.

25. J.-H. Chiang, H.-L. Li, and T.-c. Chiueh, “Introspection-based memory deduplication and migration,” in ACM SIGPLAN Notices, vol. 48, no. 7. ACM, pp. 51–62. 2013.

26. X. Song, J. Shi, R. Liu, J. Yang, and H. Chen, “Parallelizing live migration of virtual machines,” in ACM SIGPLAN Notices, vol. 48, no. 7. ACM, pp. 85–96. 2013.

27. Y. Abe, R. Geambasu, K. Joshi, and M. Satyanarayanan, “Urgent virtual machine eviction with enlightened post-copy,” 2015.

28. X. Xu and B. Davda, “Srvm: Hypervisor support for live migration with passthrough sr-iov network devices,” in Proceedings of the12th ACM SIG-PLAN/SIGOPS International Conference on Virtual Execution Environments. ACM, pp. 65–77? 2016.

29. C. Tang, “Fvd: A high-performance virtual machine image format for cloud.” in USENIX Annual Technical Conference, 2011.

30. Ahmad, A. Gulati, and A. Mashtizadeh, “vic: Interrupt coalescing for virtual machine storage device io,” in 2011 USENIX Annual Technical Conference (USENIX ATC’11), p. 45. 2011.

31. Kang, B. Zhang, T. Wo, C. Hu, and J. Huai, “Multilanes: providing virtualized storage for os-level virtualization on many cores,” in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14), pp. 317–329, 2014.

32. Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, “Live and in-cremental whole-system migration of virtual machines using block-bitmap,” in Cluster Computing, 2008 IEEE International Conference on. IEEE, pp. 99–106. 2008.

33. J. Mashtizadeh, M. Cai, G. Tarasuk-Levin, R. Koller, T. Garfinkel, and S. Setty, “Xvmotion: unified virtual machine migration over long distance,” in Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference. USENIX Association, pp. 97–108, 2014.

34. S. Yang, T. Harter, N. Agrawal, S. S. Kowsalya, A. Krishnamurthy, S. Al-Kiswany, R. T. Kaushik, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau,

“Split-level i/o scheduling,” in Proceedings of the 25th Symposium on Operating Systems Principles. ACM, pp. 474–489. 2015.

35. J.-P. Lozi, B. Lepers, J. Funston, F. Gaud, V. Qu´ema, and A. Fedorova, “The linux scheduler: a decade of wasted cores,” in EuroSys 2016, 2016.

36. Lu, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and S. Lu, “A study of linux file system evolution,” ACM Transactions on Storage (TOS), vol. 10, no. 1, p. 3, 2014.

37. Sivathanu, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and S. Jha, “A logic of file systems.” in FAST, vol. 5, pp. 1–1, 2005.

38. D. Le, H. Huang, and H. Wang, “Understanding performance implications of nested file systems in a virtualized environment.” in FAST, 2012.

39. T. Y. Kim, D. H. Kang, D. Lee, and Y. I. Eom, “Improving performance by bridging the semantic gap between multi-queue ssd and i/o virtualization framework,” in Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on. IEEE, 2015.

40. P. Sehgal, S. Basu, K. Srinivasan, and K. Voruganti, “An empirical study of file systems on nvm,” in Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on. IEEE, 2015.

41. Y. Lu, J. Shu, and W. Zheng, “Extending the lifetime of flash-based storage through reducing write amplification from file systems,” in Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13), 2013.

42. Fu, D. Feng, Y. Hua, X. He, Z. Chen, W. Xia, Y. Zhang, and Y. Tan,“Design tradeoffs for data deduplication performance in backup workloads,” in 13th USENIX Conference on File and Storage Technologies (FAST 15), 2015.

43. K. Srinivasan, T. Bisson, G. R. Goodson, and K. Voruganti, “idedup: latencyaware, inline data deduplication for primary storage.” in FAST, vol. 12, 2012.

44. L. Cui, J. Li, B. Li, J. Huai, C. Hu, T. Wo, H. Al-Aqrabi, and L. Liu, “Vmscatter: migrate virtual machines to many hosts,” in ACM SIGPLAN Notices, vol. 48, no. 7. ACM, 2013.

45. U. Deshpande, X. Wang, and K. Gopalan, “Live gang migration of virtual machines,” in Proceedings of the 20th international symposium on High performance distributed computing. ACM, 2011.

46. D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, “Difference engine: Harnessing memory redundancy in virtual machines,” Communications of the ACM, vol. 53, no. 10, pp. 85–93, 2010.

47. J. Zheng, T. Ng, K. Sripanidkulchai, and Z. Liu, “Pacer: A progress management system for live virtual machine migration in cloud computing,” Network and Service Management, IEEE Transactions on, vol. 10, no. 4, pp. 369–382, 2013.

48. K. Veeraraghavan, J. Flinn, E. B. Nightingale, and B. Noble, “qufiles: The right file at the right time,” ACM Transactions on Storage (TOS), vol. 6, no. 3, 12, 2010.

49. J. Wires, S. Ingram, Z. Drudi, N. J. Harvey, and A. Warfield, “Characterizing storage workloads with counter stacks,” in 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pp. 335–349, 2014.

50. C.-C. Tsai, Y. Zhan, J. Reddy, Y. Jiao, T. Zhang, and D. E. Porter, “How to get more value from your file system directory cache,” in Proceedings of the 25th Symposium on Operating Systems Principles. ACM, pp. 441–456, 2015.

51. H. Lu, K. Veeraraghavan, P. Ajoux, J. Hunt, Y. J. Song, W. Tobagus, S. Kumar, and W. Lloyd, “Existential consistency: measuring and understanding consistency at facebook,” in Proceedings of the 25th Symposium on Operating Systems Principles. ACM, pp. 295–310.

метадані

Заголовок

Оптимізація моделей та методів хмарної міграції даних в рішеннях на основі віртуальних машин

Автор

Бойчук М.Б. Науковий керівник / Експерт

підрозділ

King Danylo University

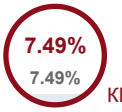
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

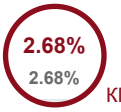
Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	70

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

18281
Кількість слів

134555
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/395/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%D0%A1%D1%82%D0%B5%D0%BF%D0%B0%D0%BD%D1%8E%D0%BA.pdf?sequence=1	88	0.48 %
2	http://www.mecs-press.org/ijitcs/ijitcs-v7-n10/v7n10-2.html	45	0.25 %
3	http://www.mecs-press.org/ijitcs/ijitcs-v7-n10/v7n10-2.html	40	0.22 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/395/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%D0%A1%D1%82%D0%B5%D0%BF%D0%B0%D0%BD%D1%8E%D0%BA.pdf?sequence=1	36	0.20 %