УДК 004.94; 004.4; 004.62 https://doi.org/10.15407/pp2018.04.003

Е.В. Жаріков

МЕТОД УПРАВЛІННЯ ДВОРІВНЕВИМ СХОВИЩЕМ ВІРТУАЛІЗОВАНОГО ЦЕНТРУ ОБРОБКИ ДАНИХ

Вимоги до систем збереження даних з боку сучасних сервісів у системах Інтернету речей та аналізу даних зростають з кожним роком. За умов віртуалізації серверів, мереж і сховищ даних виникає необхідність розробки нових моделей і методів управління системами збереження даних, що покращують продуктивність їх роботи і дозволяють зменшити капітальні та операційні витрати на IT інфраструктуру в цілому. У статті розроблено модель дворівневого сховища і метод управління, що дозволяє без суттєвого збільшення вартості зберігання даних підвищити продуктивність операцій читання/запису даних віртуальними машинами та контейнерами в гіперконвергентних і хмарних ЦОД. Результати дослідження показують, що використання дворівневих сховищ із запропонованим методом управління дає можливість знизити вартість збереження даних за рахунок зменшення об'єму і кількості пристроїв швидкого рівня. Аналіз результатів симуляції роботи дворівневого сховища і запропонованого методу управління показав, що час очікування завершення транзакцій доступу до файлів зменшується, що, у порівнянні з однорівневим сховищем, побудованим на повільних пристроях, призводить до підвищення продуктивності роботи дворівневого сховища при одночасній роботі віртуальних машин і контейнерів.

Ключові слова: система збереження даних, міграція даних, віртуалізація, хмарні обчислення.

Вступ

Цифрова трансформація, що відбувається в поточний час, впливає на зміни в IT-інфраструктурі і вимагає впровадження нових технологій і методів зберігання і обробки даних [1] з метою більш ефективного використання ресурсів та прийняття рішень. Останні тенденції в галузі хмарних обчислень, інтернету речей та машинного навчання спираються на системи збереження даних (сховища даних) як на основний ресурс центру обробки даних (ЦОД). За умов віртуалізації та консолідації ресурсів управління системами збереження даних (СЗД) становить важливу науково-практичну задачу. Для досягнення бажаних показників продуктивності роботи IT-інфраструктури в цілому необхідно забезпечити надійну, безперебійну та досить швидку роботу сховищ даних.

Сучасні застосунки виконуються в таких віртуалізованих середовищах, як віртуальні машини (ВМ) і контейнери. Але СЗД підключені безпосередньо до фізичних серверів (ФС), до мережі ЦОД або до мережі сховищ даних. Тому, з метою управління виділенням необхідних об'ємів сховища із заданими показниками продуктивності необхідно віртуалізувати і об'єднати ресурси сховища в пул.

Кожна ВМ або контейнер використовується різними бізнес-процесами, які потребують різного об'єму даних в СЗД і різної продуктивності при доступі до даних. Особливістю управління СЗД у віртуалізованому середовищі є необхідність забезпечити різні показники продуктивності роботи зі сховищем даних для різних ВМ та контейнерів на одному ФС.

Сучасні СЗД являють собою складні підсистеми ЦОД з різними варіантами підключення (DAS – Direct Attached Storage, NAS – Network Attached Storage, SAN – Storage Area Network), побудовані з використанням пристроїв з різними технологіями запису (флеш-пам'ять, магнітні диски та ін.), і надають доступ до даних, що представлені різними структурами (блоки, файли, об'єкти, документи). Тому, якщо процесорний ресурс має один вимір і може бути розподілений між віртуальними машинами або контейнерами, що виконуються локально, то СЗД, як ресурс, має декілька вимірів (об'єм, пропускна здатність, структури даних та ін.). Також, треба врахувати, що навантаження на ВМ та контейнери змінюється, і немає необхідності підтримувати високу пропускну здатність доступу до даних весь час. Це, в свою чергу, призводить до перевантаження одних

каналів доступу до даних, а з іншого боку до недовантаження інших каналів.

Ще одним важливим фактором, який впливає на застосування тієї чи іншої СЗД в ЦОД, – ціна. СЗД, що можуть обслуговувати динамічні навантаження з високою продуктивністю і низькою затримкою, побудовані з використанням флеш накопичувачів і коштують на порядки дорожче, чим СЗД, побудовані з використанням накопичувачів з магнітними дисками. Також, на ціну дуже впливає спосіб підключення СЗД до споживачів. Найдорожчим варіантом розгортання СЗД є SAN, набагато дешевшим варіантом реалізації СЗД – DAS.

На разі, незалежно від технології організації доступу до сховища, необхідно застосувати той чи інший принцип читання/запису даних. Наявні в ЦОД пристрої, що реалізують старі, нові або найновіші принципи запису даних на носії, можливо розділити за критерієм швидкості на дві групи: швидкісні (найсучасніші) і повільні (пристрої і інтерфейси попередніх поколінь). Зазвичай, впровадження і використання швидкісних пристроїв коштує дорожче (іноді набагато дорожче), чим використання поширених пристроїв зберігання даних попередніх поколінь. Таким чином, завжди виникає задача досягнення високої продуктивності читання/запису даних при зменшенні витрат на придбання і експлуатацію пристроїв і систем збереження даних.

При цьому, ефективне використання ресурсів СЗД полягає у забезпеченні заданої якості обслуговування запитів доступу до даних при мінімізації капітальних та операційних витрат на утримання СЗД.

З появою швидких пристроїв і інтерфейсів у системах збереження даних стало можливим об’єднувати в рамках сховища пристрої з різною продуктивністю, створюючи так звані багаторівневі сховища (storage tiering) [2–5].

Одним із шляхів зменшення вартості придбання і експлуатації СЗД без суттєвої втрати об’єму є міграція даних між швидкодіючими пристроями одного рівня та повільними пристроями іншого рівня сховища. При цьому, рівень швидкодіючих пристроїв має набагато менший об’єм, чим об’єм пристроїв повільного рівня. Відповідно, вартість придбання і експлуатації СЗД зменшується завдяки зменшенню кількості коштовних швидких пристроїв.

Для вирішення задачі управління СЗД ЦОД в статті пропонується метод управління, що базується на моделі дворівневого сховища і алгоритмах міграції даних між швидким та повільним рівнями сховища.

1. Аналіз публікацій

Системам збереження даних в ЦОД приділяється все більше уваги у зв’язку із зміною вимог до швидкісних і об’ємних показників при роботі з даними з боку сучасних сервісів. Останнім часом набули розвитку нові технології роботи з даними, такі як сховища на основі флеш пам’яті (Flash storage), енергонезалежна пам'ять (Non-volatile memory express) [6], фабрики сховищ на основі Ethernet (Ethernet storage fabric) [7], пам'ять класу сховища (Storage class memory) [8].

При цьому, адаптація і впровадження нових, більш продуктивних, технологій збереження даних не витісняє існуючі, більш повільні, а доповнює їх. Крім того, з одного боку, сучасні сервіси вимагають швидкого доступу до даних і високої продуктивності, з іншого – великих об’ємів для довготривалого зберігання і резервного копіювання.

Розробці нових методів і фреймворків для підвищення ефективності роботи СЗД приділяється багато уваги з боку науковців та інженерів [3–5, 9–12]. В роботі [9] запропонований менеджер розміщення даних “AutoTiering”, який працює з багаторівневим сховищем, що складається тільки з SSD пристроїв. Розподіл пристроїв між рівнями сховища відбувається в залежності від продуктивності накопичувачів та їх вартості. Стан кожного рівня сховища оцінюється з використанням метрик: продуктивність операцій читання/запису випадкових даних (IOPS), швидкість читання/запису послідовних даних (MBPS) та розмір сховища. Навантаження для оцінки роботи запропонованого рішення згенеровані з використанням IOMeter [13] і

FIO [14]. Основна мета, яку автори [9] при роботі з багаторівневим сховищем намагаються досягти – збільшення прибутків при збережені даних. Максимізація критерію досягається шляхом міграції даних між пристроями рівнів у залежності від вартості збереження даних та вартості їх міжрівневої міграції. Однак, додавання чергового рівня до сховища призводить до необхідності додавати ще один рівень управління і ще один рівень міграції даних, ускладнюючи загальну схему роботи з даними.

У роботі [10] представлено математичні моделі для задач кластеризації та управління ресурсами систем збереження даних з метою мінімізації витрат на користування існуючими сховищами, покращення якості обслуговування споживачів даних та рівномірного розподілення файлів між рівнями збереження даних. Для розробки моделей систем збереження даних запропоновано використовувати методи математичного програмування з використанням критеріїв мінімізації витрат і рівномірного заповнення рівнів та пристроїв з урахуванням ресурсних, технологічних і часових обмежень. Але розроблені алгоритми показали різну продуктивність роботи з файлами різного розміру. Крім того, не обґрунтовано використання трьох рівнів сховища і не вказано, для якого навантаження проводились дослідження роботи запропонованих алгоритмів.

В роботі [11] запропоновано схему збереження даних з використанням дворівневого сховища. На рівні файлової системи створюються логічні гібридні диски з відповідними позначками розміщення блоків даних, що належать програмам при їх запуску і функціонуванні. Це дозволяє усунути затримки при пошуку блоків даних у відповідних таблицях відображення і розмістити часто використовувані блоки на пристроях з високою продуктивністю. В результаті застосування представленої схеми прискорився запуск програм та підвищилась продуктивність роботи зі сховищем. Запропонована схема підтримує міграцію блоків між твердотілими накопичувачами (solid-state drive, SSD) та пристроями на магнітних дисках (hard disk drive, HDD), та навпаки. Представлений графічний розподіл запитів блоків різної довжини при використанні програм LibreOffice та Eclipse показав, що більш частіше відбувається зчитування коротших послідовностей блоків даних. Однак в запропонованій схемі виконується міграція тільки певних блоків, а не всіх, що належать до виконуваної програми. Також, зворотна міграція блоків з рівня SSD на рівень HDD виконується тільки для блоків, які довго не використовуються, незалежно від їх розміру.

Розподілена файлова система, що враховує показники продуктивності різних рівнів системи збереження даних, представлена в роботі [12]. Запропонована файлова система розроблена на базі HDFS [15] і дозволяє застосовувати політики автоматизації управління даними як в межах рівнів сховища на одному вузлі, так і в розподіленому варіанті. Із застосуванням багатокритеріальної оптимізації автори запропонували схеми забезпечення відмовостійкості, балансування навантаження та збільшення продуктивності. Однак запропонована система працює на файловому рівні і орієнтована на вузьке коло застосування, наприклад Hadoop [16] і Spark [17].

2. Постановка задачі

Організація роботи сховища у віртуалізованому середовищі полягає у вирішенні задачі забезпечення заданої продуктивності доступу до даних з боку віртуальних машин та контейнерів.

Один із способів підвищення продуктивності роботи сховища без суттєвого підвищення вартості зберігання даних є розбиття сховища на два рівні: перший рівень є найпродуктивнішим, меншого об’єму і дорожчим при використанні, другий рівень є повільним, більшого об’єму і дешевшим. При організації роботи сховища шляхом розбиття його на рівні виникає необхідність управління процесом міграції даних між рівнями. При цьому, ВМ та контейнери обробляють дані, що знаходяться на пристроях швидкого рівня. В сучасних СЗД швидкий рівень складається з твердотілих накопичувачів. Ці пристрої мають найвищу продуктивність роботи з даними

в рамках сховища. Якщо необхідних даних на швидкому рівні немає, відбувається міграція даних з менш швидкісного (повільного) рівня. В системах збереження даних повільний рівень складається з пристроїв на магнітних дисках (HDD). Міграція з повільного рівня на швидкий виконується з метою зменшення затримки при роботі з даними в сховищі. Одночасно з цим відбувається міграція даних зі швидкого рівня на повільний рівень з метою звільнення місця на пристроях швидкого рівня для подальшої обробки та міграції «гарячих» даних.

Таким чином, необхідно розробити модель дворівневого сховища і метод управління, що дозволяє без суттєвого збільшення вартості зберігання даних підвищити продуктивність операцій читання/запису даних віртуальними машинами та контейнерами в гіперконвергентних і хмарних ЦОД.

3. Розробка моделі дворівневого сховища

Основними елементами моделі дворівневого сховища є дані (файли та блоки), пристрої швидкого рівня, пристрої повільного рівня та фізичні сервери. До кожного сервера підключено декілька пристроїв, які розміщені в загальному випадку на декількох рівнях, в залежності від їх продуктивності. В розробленій моделі пропонується використовувати два рівні: швидкий і повільний.

Позначимо n кількість файлів, які зберігаються віртуальною машиною в локальному сховищі поточного ФС. В загальному випадку кількість файлів є змінною. Збереження файлів у сховищі відбувається блочно, розмір блоків залежить від умов використання і операційних систем. Позначимо t кількість блоків, в яких зберігаються файли. При постановці задачі розмір блока дорівнює 4 Кб. Відповідно, кількість блоків теж змінюється в процесі роботи сховища. Кількість пристроїв швидкого рівня на одному ФС позначимо m, кількість пристроїв повільного рівня – c. Для опису моделі і вирішення задач управління роботою сховища поточного ФС введемо наступні змінні:

d_i – розмір i-го файлу, $i=\overline{1,n}$;

b – розмір блоку даних;

q_j – кількість разів доступу до j-го блоку протягом заданого часу (це може бути день, тиждень, місяць і т. д.), $j=\overline{1,t}$;

B_{ij} – логічна змінна розбиття файлів на блоки (B_{ij}=1 якщо j-ий блок є частиною i-го файлу, B_{ij}=0 – в іншому випадку), $i=\overline{1,n}$, $j=\overline{1,t}$;

s_k – розмір k-го швидкого пристрою, $k=\overline{1,m}$;

x_{kj} – логічна змінна розміщення j-го блоку на k-ому швидкому пристрої, $k=\overline{1,m}$, $j=\overline{1,t}$;

C_k – вартість використання k-го швидкого пристрою, $k=\overline{1,m}$;

G_k – вартість зберігання одного блоку на k-му швидкому пристрої, $k=\overline{1,m}$;

h_v – розмір v-го повільного пристрою, $v=\overline{1,c}$;

y_{vj} – логічна змінна розміщення j-го блоку на v-му повільному пристрої, $v=\overline{1,c}$, $j=\overline{1,t}$;

D_v – вартість використання v-го повільного пристрою, $v=\overline{1,c}$;

g_v – вартість зберігання блоку на v-му повільному пристрої, $v=\overline{1,c}$;

r_{vk} – вартість міграції блоку з v-го повільного пристрою на k-ий швидкий пристрій, $k=\overline{1,m}$, $v=\overline{1,c}$.

Робота з блоками відбувається на швидкому рівні. Якщо необхідного блоку на швидкому рівні немає, то відбувається зчитування його з повільного рівня, відправка у відповідні канали обміну з пам’яттю, запис блоку на швидкий рівень та видалення блоку з повільного рівня. Подальша робота з цим блоком відбувається вже читанням/записом із швидкого рівня. У випадку, коли неможливо записати блок на швидкий рівень, робота з блоком відбувається читанням/записом з повільного рівня до тих пір, поки не з’явиться

вільне місце на швидкому рівні. При міграції блоку зі швидкого рівня на повільний рівень спочатку відбувається його запис на повільний рівень, а потім видалення із швидкого рівня. Відповідно, при міграції блоку з повільного рівня на швидкий рівень, спочатку виконується запис на швидкий рівень, а потім видалення з повільного рівня. Таким чином, блок існує в одному екземплярі при закінченні процесу міжрівневої міграції.

Причиною для міграції між рівнями постає наявність/відсутність доступу до блоків даних протягом деякого часу. Через це необхідна максимізація середньої кількості транзакцій роботи з блоками, які розміщуються на пристроях швидкого рівня (1).

$$\max \sum_{k=1}^{m} \frac{\sum_{j=1}^{t} q_j x_{kj}}{\sum_{j=1}^{t} x_{kj}}. \qquad (1)$$

Кількість блоків на поточному ФС знаходиться з використанням наступного рівняння:

$$t = \frac{\sum_{i=1}^{n} d_i}{b}.$$

В моделі треба врахувати обмеження.

1. Наявність вільного місця на пристроях швидкого рівня позначимо

$$\sum_{k=1}^{m} \sum_{j=1}^{t} x_{kj} b < \sum_{k=1}^{m} s_k. \qquad (2)$$

2. Наявність вільного місця на пристроях повільного рівня позначимо

$$\sum_{v=1}^{c} \sum_{j=1}^{t} y_{vj} b < \sum_{v=1}^{c} h_v. \qquad (3)$$

3. На пристроях швидкого рівня та пристроях повільного рівня *j*-й блок має зберігатись лише в одному екземплярі:

$$\sum_{k=1}^{m} x_{kj} = 1, \qquad (4)$$

$$\sum_{v=1}^{c} y_{vj} = 1. \qquad (5)$$

4. Кожен з блоків *j* належить тільки одному файлу

$$\sum_{i=1}^{n} B_{ij} = 1,\ j = \overline{1,t}. \qquad (6)$$

Кількість пристроїв для збереження даних на кожному рівні визначається, з одного боку, можливостями інтерфейсів підключення, а з іншого – вартістю використання пристроїв. Вартість збереження даних на всіх швидких пристроях визначається наступним чином:

$$\sum_{k=1}^{m} C_k. \qquad (7)$$

Вартість збереження даних на всіх повільних пристроях визначається за допомогою формули

$$\sum_{v=1}^{c} D_v. \qquad (8)$$

Вартість зберігання блоків даних на *k*-му швидкому пристрої знаходиться за допомогою виразу

$$\sum_{j=1}^{t} G_k x_{kj}. \qquad (9)$$

Вартість зберігання блоків даних на *v*-му повільному пристрої знаходиться за допомогою виразу

$$\sum_{j=1}^{t} g_v y_{vj}. \qquad (10)$$

У випадках, коли зберігання даних на пристроях рівня з кращою продуктивністю може стати невиправданим (дорогим), то необхідно мігрувати ці дані на рівень з нижчою продуктивністю, де зберігання даних буде більш дешевим. Вартість міграції між двома пристроями різних рівнів розраховується наступним чином:

$$\sum_{v=1}^{c} \sum_{k=1}^{m} r_{vk}. \qquad (11)$$

Таким чином, враховуючи вирази (7) – (11), мінімізація вартості збереження даних у локальному сховищі ФС з можливістю міграції даних між пристроями різних рівнів виконується за допомогою критерію:

$$\min \begin{bmatrix} \sum_{k=1}^{m}(C_k + \sum_{j=1}^{t} G_k x_{jk}) + \\ \sum_{v=1}^{c}(D_v + \sum_{j=1}^{t} g_v y_{jv}) + \\ \sum_{v=1}^{c}\sum_{k=1}^{m} r_{vk} \end{bmatrix}, \qquad (12)$$

за умов виконання обмежень (2) – (6).

4. Метод міграції даних між рівнями сховища

Ідея методу управління дворівневим сховищем ФС базується на міграції даних між швидким та повільним рівнями сховища і полягає у наступному. Локальний пул дисків ФС розбивається на два рівні: швидкий і повільний (рис. 1). Застосунки, що виконуються у ВМ, в будь-який момент часу використовують не всі дані логічних томів (дисків). Якщо виникають запити до певних файлів протягом деякого часу необхідно мігрувати відповідні блоки даних з повільного рівня на швидкий. Якщо запити до цих файлів зникають і блоки даних цих файлів більше не потрібні, то виконується їх міграція на повільний рівень, що призводить до вивільнення місця на пристроях швидкого рівня. Таким чином, щоб на рівні швидких пристроїв було в наявності місце, необхідно постійно підтримувати частину вільного місця, яку можливо використовувати для зберігання «гарячих» даних. Якщо створюються нові дані, то в першу чергу вони розміщуються на пристроях швидкого рівня.

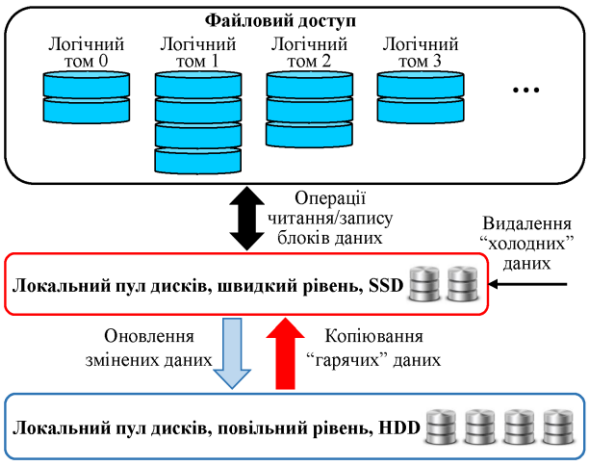

Рис. 1. Архітектура дворівневого локального сховища

В основу метода покладено такі алгоритми: алгоритм сортування файлів за двома критеріями, алгоритм міграції файлів з швидкого рівня на повільний та алгоритм міграції файлу з повільного рівня на швидкий.

Для забезпечення роботи методу управління дворівневим сховищем необхідно сортувати файли за двома критеріями: за розміром файлу і за кількістю транзакцій доступу до файлу. Кожен з критеріїв може мати певну вагу, яку треба визначати експериментально.

Сортування файлів, які розміщуються на швидкому рівні сховища даних, виконується Алгоритмом 1. Позначимо L^F – список файлів швидкого рівня; $new\,L^F$ – відсортований за двома критеріями список файлів швидкого рівня; W_i – комплексна вага i-го файлу; L^W – список двійок (i, W_i) файлів для ідентифікації і доступу; $size\,L^F$ – список файлів, відсортованих за розміром; $access\,L^F$ – список файлів, відсортованих за кількістю транзакцій доступу; w_{size} – вага критерію розміру; w_{access} – вага критерію кількості транзакцій доступу до файлу. Функція $access(L_i^F)$ повертає кількість транзакцій доступу до i-го файлу, функція $size(L_i^F)$ повертає розмір i-го файлу; функція $sizeof(L^F)$ повертає довжину списку файлів.

Алгоритм 1. Сортування файлів за критеріями розміру та кількості транзакцій доступу до файлів.

Вхідні дані: L^F, w_{size}, w_{access}.

Вихідні дані: $new\,L^F$.

1. Ініціалізація списків:
   $new\,L^F \leftarrow$ NULL,
   $W_i \leftarrow$ NULL,
   $size\,L^F \leftarrow$ NULL,
   $access\,L^F \leftarrow$ NULL.
2. $access\,L^F \leftarrow$ відсортувати L^F за збільшенням кількості транзакцій доступу до фійлів.
3. $size\,L^F \leftarrow$ відсортувати L^F за зменшенням розміру файлів.

4. $N \leftarrow sizeof(L^F)$.
5. **for** i = 1 **to** N **do**.
6. $W_i \leftarrow w_{size} size(L_i^F) + +w_{access} access(L_i^F)$.
7. $L^W \leftarrow (i, W_i)$.
8. **end for.**
9. $newL^F \leftarrow$ відстортувати L^W за зменшенням коефіцієнту W_i.
10. **return** $new\,L^F$.

Нові файли завжди за можливістю зберігаються на швидкому рівні сховища. Так як кількість вільного місця на швидкому рівні не має бути менше визначеного порогового значення, файли з найменшою кількістю доступів і найбільшого розміру необхідно мігрувати з швидкого рівня. Процес міграції відбувається паралельно з роботою швидкого рівня з обслуговування транзакцій читання/запису. Блоки даних мігрують на повільний рівень поки на швидкому рівні кількість вільного місця не стане більше визначеного порогового значення Th^{SSD}. Значення Th^{SSD} залежить від інтенсивності роботи з файлами великого розміру і підбирається експериментально. Алгоритм враховує, що на пристроях повільного рівня завжди є місце для мігруючих файлів. Алгоритм міграції блоків на повільний рівень (Алгоритм 2) працює наступним чином:

Алгоритм 2. Міграція блоків файлу з швидкого рівня на повільний рівень.

Вхідні дані: $new\,L^F$ – список файлів для міграції.

Вихідні дані: M – результат міграції файлів (позитивний/негативний).

1. $i \leftarrow 1$.
2. $M \leftarrow$ false.
3. **while** $\sum_{k=1}^{m} s_k - size(newL^F) < Th^{SSD}$ **then.**
4. Мігрувати i-й файл на повільний рівень.
5. Видалити i-й файл зі швидкого рівня.
6. Видалити i-й файл з L^F та $new\,L^F$.
7. $i \leftarrow i+1$.
8. $M \leftarrow$ true.
9. **end while**.
10. **return** M.

Алгоритм міграції даних з повільного рівня на швидкий (Алгоритм 3) працює наступним чином:

Алгоритм 3. Міграція файлу з повільного рівня на швидкий.

Вхідні дані: F – запитуваний для читання/запису файл.

Вихідні дані: результат міграції файлу (позитивний/негативний).

1. $k \leftarrow 1$, $M \leftarrow$ True.
2. **repeat.**
3. **if** (файл F розташований на швидкому рівні) **then**.
4. Зчитати/записати файл зі швидкого рівня.
5. **else**.
6. Налаштувати зчитування/запису файлу F з повільного рівня.
7. Зчитати/записати файл з повільного рівня.
8. **while** (k <= m) and M **do**.
9. **if** $s_k - size(F) > 0$ **then**.
10. Записати файл F на швидкісний пристрій k.
11. Налаштувати зчитування/запис файлу F з швидкісного рівня.
12. Видалити файл F з повільного пристрою.
13. $M \leftarrow$ False.
14. **end if**.
15. $k \leftarrow k+1$.
16. **end while**.
17. **if** k=m and M **then**.
18. Виконати алгоритм 2.2.
19. **end if**.
20. **end if**.
21. **until** (транзакції з файлом відбуваються).

Алгоритм 3 працює кожного разу, коли починаються транзакції з певним файлом. При цьому Алгоритм 2 може запускатись як у відповідь на нестачу вільного місця на швидкому рівні, так і через певні інтервали часу. Визначення таких інтервалів не входить до розгляду в цій статті.

5. Моделювання дворівневого сховища

Для дослідження запропонованої моделі дворівневого сховища та методу управління розроблено програмний застосунок з використанням мови програмування Java. Запуск застосунку для симуляції виконувався на комп'ютері з процесором Intel i7-3632QM і обсягом пам'яті 8 GB під управлінням ОС Windows 10 Pro 64bit.

Для формування вхідних даних, що подаються на вхід застосунку, використано програмне забезпечення Process Monitor [18]. Process Monitor дає можливість записати перелік процесів, які виконуються і отримують доступ до файлів на дисках при роботі з операційною системою Windows.

Збір вхідних даних для симуляції включає в себе декілька етапів. На першому етапі, з використанням Process Monitor, було зібрано дані про використання файлів на дисковому пристрої застосунками Windows звичайного офісного комп'ютера на протязі 8 годин.

Другий етап. Збереження звіту про роботу Process Monitor у форматі CSV, в який записуються результати моніторингу (рис. 2).

Третій етап. Формування вхідних даних для дослідження моделі дворівневого сховища та якості використання представлених в роботі алгоритмів (рис. 3). На вхід кожної ВМ при моделюванні подається наступна інформація про активність дискової підсистеми: файл та шлях до нього; розмір файлу; чи змінювався його розмір протягом часу роботи Process Monitor; кількість запитів до файлу.

Для виконання третього етапу підготовки вхідних даних розроблений окремий застосунок з метою конвертації даних формату Process Monitor у формат застосунку симуляції.

Для моделювання дворівневого сховища і дослідження запропонованого методу управління необхідно задати наступні початкові дані: кількість серверів, які необхідні для симуляції; кількість пристроїв, які знаходяться на швидкому рівні на кожному з серверів системи; кількість пристроїв, які знаходяться на повільному рівні. Крім того, в конфігураційний файл симулятора треба додати дані про конфігурацію пристроїв кожного рівня та кількість віртуальних машин на кожному сервері.

Характеристики роботи дискових пристроїв, які розміщуються на швидкому рівні: затримка 1 мс; середній час доступу до файлів 1 мс; пропускна спроможність дискового пристрою 300 МБ/с; об'єм диску 64ГБ.

Характеристики роботи дискових пристроїв на повільному рівні сховищ: затримка 6 мс; середній час доступу до файлів 9 мс; пропускна спроможність дискового пристрою 120 МБ/с; об'єм диску 500ГБ.

Time of Da	Process Na	PID	Operation	Path	Result	Detail	User
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, l	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.7	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, l	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:14.8	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:14.9	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM
01:15.0	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 0, l	NT AUTHORITY\SYSTEM
01:15.0	McUpdate	10052	ReadFile	C:\Prograr	SUCCESS	Offset: 36,	NT AUTHORITY\SYSTEM
01:15.1	McUpdate	10052	CreateFile	C:\Prograr	SUCCESS	Desired Ac	NT AUTHORITY\SYSTEM

Рис. 2. Фрагмент CSV файлу при експорті з Process Monitor

Path	Size	Change Siz	Count
"C:\Windo	0	FALSE	9
"C:\Windo	0	FALSE	22
"C:\Windo	14336	FALSE	2
"C:\Windo	249856	FALSE	394
"C:\Windo	0	FALSE	24
"C:\Windo	444752	FALSE	20
"C:\Windo	636048	FALSE	28
"C:\Windo	65536	FALSE	11
"C:\Windo	2319872	FALSE	91
"C:\Windo	491520	FALSE	24
"C:\Progra	16086	TRUE	1490
"C:\Progra	190	FALSE	13
"C:\Windo	115200	FALSE	2953
"C:\Windo	16896	FALSE	1681
"C:\Progra	19618	FALSE	438
"C:\Progra	25179	FALSE	104
"C:\Progra	32	FALSE	144

Рис. 3. Вхідні дані для симуляції дворівневого сховища

Симулятор створює введену кількість серверів у системі моделювання. На кожному сервері створюються два рівні пристроїв, на кожному з рівнів створюється задана кількість пристроїв. Після створення серверів генеруються віртуальні машини, які будуть отримувати доступ до сховища згідно параметрам, що записані у вхідних даних. Для кожної ВМ, в три етапи, з використанням Process Monitor, згенеровано окремий вхідний файл з описом характеристик доступу до файлів (частота, розмір, вид операції та ін.) як описано вище.

В експерименті на кожному сервері моделюється робота п’яти віртуальних машин. Відповідно, для роботи із застосунком моделювання використано п’ять CSV файлів, в яких містяться дані про доступ до файлів ОС Windows загальним розміром 8742 Мб. Міграція даних між рівнями сховища одного фізичного сервера відбувалась у випадку наявності певної визначеної кількості транзакцій доступу до певного файлу ОС, що приводило до необхідності міграції. При великому навантажені на сервер, коли протягом всього часу моделювання від кожної віртуальної машини до сховища надходять запити на обробку великої кількості даних, міграція допомагає зменшити час очікування даних з сховищ шляхом збереження часто запитуваних файлів на пристроях швидкого рівня.

В поточній версії застосунку моделювання роботи з окремими блоками файлів не виконується. Таким чином, при доступі до файлу він цілком записується на швидкий рівень.

6. Оцінка результатів моделювання

Під час запусків застосунку моделювання встановлено, що наявна інтенсивність роботи з файлами з боку віртуальних машин не призвела до вичерпання дискового простору на швидкому рівні сховища. Поріг підтримки вільного місця на кожному з дисків швидкого рівня встановлений у розмірі 100 МБ, виходячи з розміру найбільшого файлу у вхідних даних.

В результаті моделювання отримані залежності часу доступу до файлу в залежності від його розміру. Моделювання виконано для двох конфігурацій сховищ: з дворівневим сховищем і з однорівневим сховищем. Однорівневе сховище моделюється пристроями повільного рівня. Для кожної конфігурації виконано десять запусків моделювання. В результаті отримані середні показники.

Так як запис файлів на дворівневе сховище в більшості випадків відбувається на швидкий рівень, то час запису визначається показниками пристроїв швидкого рівня (рис. 4). Виключення може бути тоді, коли створюється новий файл, розмір якого вище, чим вільне місце на одному з пристроїв швидкого рівня. В такому випадку файл створюється на пристрої повільного рівня. При цьому, паралельно, виконується очистка пристроїв швидкого рівня з метою подальшого перенесення цього нового файлу на швидкий рівень (в фоновому режимі). Таким чином, середній час запису такого файлу трохи зростає.

При записі файлів в однорівневе сховище витрачається більший час (рис. 5) через те, що продуктивність пристроїв повільного рівня нижча за продуктивність пристроїв швидкого рівня.

При роботі дворівневого сховища в режимі читання файлів перша транзакція читання файлу відбувається з пристрою

повільного рівня з одночасним записом цього файлу на швидкий рівень. Подальша робота з файлом вже буде відбуватися шляхом доступу до пристроїв швидкого рівня. Виключенням є ситуація, коли перенесення цього файлу на швидкий рівень неможливо через нестачу вільного місця. В такому випадку запускається алгоритм очистки пристроїв швидкого рівня паралельно з доступом до цього файлу на пристрої повільного рівня. Коли вільне місце з'являється на пристрої швидкого рівня цей файл мігрує на швидкий рівень і подальша робота з ним відбувається вже з вищою продуктивністю. Таким чином, при зчитуванні файлів з багаторівневого сховища (рис. 6) час очікування доступу до файлу є меншим через те, що дані знаходяться на швидкому рівні.

При зчитуванні файлів з однорівневого сховища витрачається більший час (рис. 7) через те, що продуктивність пристроїв повільного рівня нижча за продуктивність пристроїв швидкого рівня.

Таким чином, розглядаючи усереднені значення зчитування і запису деякої кількості файлів з використанням дворівневого сховища та однорівневого, можна зробити висновок, що використання міграції дозволяє зменшити час очікування доступу до файлів при виконанні одночасних дискових операцій віртуальними машинами на фізичних серверах.

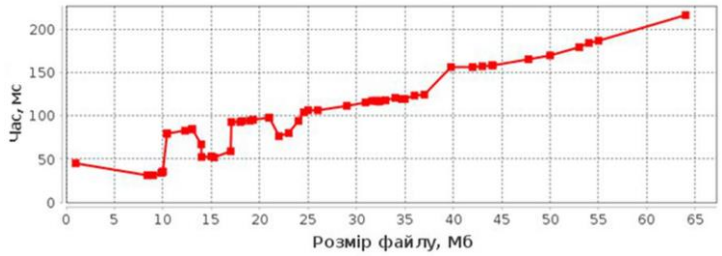

Рис. 4. Запис файлів в дворівневе сховище

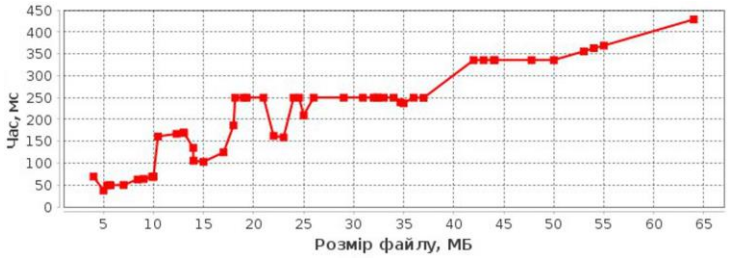

Рис. 5. Запис файлів в однорівневе сховище

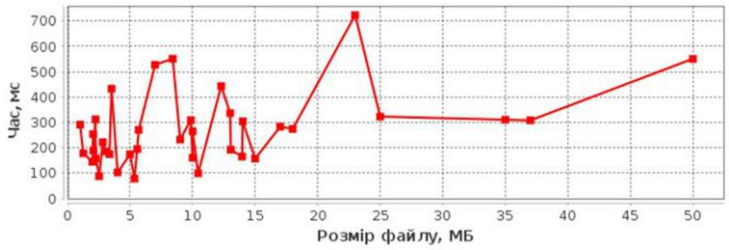

Рис. 6. Читання файлів з дворівневого сховища

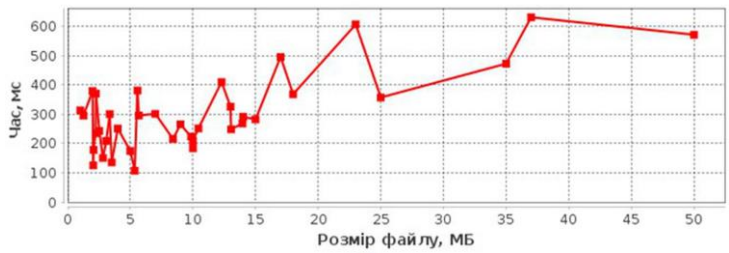

Рис. 7. Читання файлів з однорівневого сховища

Висновки

Для управління процесами доступу до даних на рівні сховища фізичного сервера в статті запропоновано і досліджено метод управління дворівневим сховищем на основі міграції даних між швидким і повільним рівнями сховища.

Запропонований метод управління базується на моделі дворівневого сховища і алгоритмах міграції даних між швидким та повільним рівнями сховища за критерієм мінімізації вартості збереження даних. Вивільнення місця з швидкого рівня відбувається з використанням двох критеріїв: за розміром файлу і за кількістю транзакцій доступу до файлу.

Для дослідження запропонованої моделі та методу управління розроблено програмний застосунок, що дозволяє моделювати роботу дворівневого і одноріневого сховища. Результати дослідження показують, що використання дворівневих сховищ із запропонованим методом управління призводить до зменшення необхідного об'єму пристроїв швидкого рівня (зниження вартості збереження даних) та зменшення часу очікування завершення транзакцій доступу до файлів при одночасній роботі віртуальних машин зі сховищем фізичного сервера.

Подальше дослідження запропонованої моделі і методу управління пов'язане з доробкою застосунка моделювання з метою урахування блочного обміну та підбору вагових коефіцієнтів для досягнення високої продуктивності роботи дворівневого сховища.

Література

1. Top 10 Digital Transformation Trends For 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/sites/danielnewman/2018/09/11/top-10-digital-transformation-trends-for-2019/#17d55d1e3c30.
2. ZFS L2ARC [Електронний ресурс] – Режим доступу до ресурсу: http://137.254.16.27/brendan/entry/test.
3. Chen F., Koufaty D.A., Zhang X. Hystor: making the best use of solid state drives in high performance storage systems. *In Proceedings of the international conference on Supercomputing*. 2011. P. 22–32.
4. Guerra J., Pucha H., Glider J.S., Belluomini W., Rangaswami R. Cost Effective Storage using Extent Based Dynamic Tiering. FAST. 2011. N 11. P. 1–14.
5. Arteaga D., Zhao M. Client-side flash caching for cloud systems. *In Proc. 7th ACM Int. Syst. Storage Conf.* 2014. P. 1–11.
6. NVM Express [Електронний ресурс] – Режим доступу до ресурсу: https://nvmexpress.org/.
7. Ethernet Storage Fabric – Part 1 [Електронний ресурс] – Режим доступу до ресурсу: http://www.mellanox.com/blog/2018/05/ethernet-storage-fabric-part-1/.
8. Top 10 storage trends for 2018 and why you should care [Електронний ресурс] – Режим доступу до ресурсу: https://community.hpe.com/t5/Around-the-Storage-Block/Top-10-storage-trends-for-2018-and-why-you-should-care/bap/6995176#.W7DrSXnWhmA.
9. Yang Z. et al. AutoTiering: automatic data placement manager in multi-tier all-flash datacenter. *Performance Computing and Communications Conference* (*IPCCC*), 2017 IEEE 36th International. IEEE. 2017. P. 1–8.
10. Боданюк М.Є., Карнаухов О.К., Ролік О.І., Теленик С.Ф. Управління системами збереження даних. Electronics and Communications. 2013. № 5 (76). С. 81–90.
11. Ryu J., Lee D., Han C., Shin H., and Kang K. File-System-Level Storage Tiering for Faster Application Launches on Logical Hybrid Disks. IEEE Access. 2016. Vol. 4. P. 3688–3696.
12. Kakoulli E., Herodotou H. OctopusFS: A distributed file system with tiered storage management. *In Proceedings of the 2017 ACM International Conference on Management of Data*. 2017. P. 65–78.
13. Intel IOMeter [Електронний ресурс]. – Режим доступу до ресурсу: http://www.iometer.org.
14. FIO: Flexible I/O Tester [Електронний ресурс] – Режим доступу до ресурсу: http://linux.die.net/man/1/fio.
15. Borthakur D. The hadoop distributed file system: Architecture and design. Hadoop Project Website. 2007. T. 11. 2007. P. 21.
16. T. White Hadoop: The Definitive Guide. Yahoo! Press, 2010.
17. Zaharia M., Chowdhury M., Das T., Dave A., Ma J., et al. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. *In Proc. of the 9th Symp.*

on Networked Systems Design and Implementation (*NSDI*). 2012. P. 15–28.
18. Process Monitor. [Електроний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon.

References

1. Top 10 Digital Transformation Trends For 2019 Framework [Online] – Available from: https://www.forbes.com/sites/danielnewman/2018/09/11/top-10-digital-transformation-trends-for-2019/#17d55d1e3c30.
2. ZFS L2ARC [Online] – Available from: http://137.254.16.27/brendan/entry/test.
3. Chen, F., Koufaty, D. A., & Zhang, X. (2011, May). Hystor: making the best use of solid state drives in high performance storage systems. *In Proceedings of the international conference on Supercomputing* (P. 22–32). ACM.
4. Guerra J., Pucha H., Glider J.S., Belluomini W., & Rangaswami R. (2011, February). Cost Effective Storage using Extent Based Dynamic Tiering. In *FAST* (11). P. 1–14.
5. Arteaga D., & Zhao M. (2014, June). Client-side flash caching for cloud systems. *In Proceedings of International Conference on Systems and Storage* (P. 1–11). ACM.
6. NVM Express [Online] – Available from: https://nvmexpress.org/.
7. Ethernet Storage Fabric – Part 1 [Online] – Available from: http://www.mellanox.com/blog/2018/05/ethernet-storage-fabric-part-1/.
8. Top 10 storage trends for 2018 and why you should care [Online] – Available from: https://community.hpe.com/t5/Around-the-Storage-Block/Top-10-storage-trends-for-2018-and-why-you-should-care/ba-p/6995176#.W7DrSXnWhmA.
9. Yang Z., Hoseinzadeh M., Andrews A., Mayers C., Evans D.T., Bolt R.T., … & Swanson S. (2017, December). AutoTiering: automatic data placement manager in multi-tier all-flash datacenter. *In Performance Computing and Communications Conference* (IPCCC), 2017 IEEE 36th International (P. 1–8). IEEE.
10. Bodanyuk M.E., Karnaukhov O.K., Rolik O.I., Telenyk S.F. (2013). Management of data storage systems *Electronics and Communications*. (5–76). P. 81–90. (in Ukrainian).
11. Ryu J., Lee D., Han C., Shin H., & Kang K. (2016). File-System-Level Storage Tiering for Faster Application Launches on Logical Hybrid Disks. *IEEE Access*, 4, 3688–3696.
12. Kakoulli E., & Herodotou H. (2017, May). OctopusFS: A distributed file system with tiered storage management. *In Proceedings of the 2017 ACM International Conference on Management of Data* (P. 65–78). ACM.
13. Intel IOMeter [Online] – Available from: http://www.iometer.org.
14. FIO: Flexible I/O Tester [Online] – Available from: http://linux.die.net/man/1/fio.
15. Borthakur D. (2007). The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11(2007), 21.
16. White T. (2012). Hadoop: The definitive guide. " O'Reilly Media, Inc.".
17. Zaharia M., Chowdhury M., Das T., Dave A., Ma J., McCauley M., ... & Stoica I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (P. 15–28). USENIX Association.
18. Process Monitor [Online] – Available from: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon.

Одержано 02.10.2018

***Про автора*:**

Жаріков Едуард В'ячеславович,
кандидат технічних наук,
доцент кафедри АСОІУ.
Кількість наукових публікацій в українських виданнях – 94.
Кількість наукових публікацій в зарубіжних виданнях – 24.
Індекс Гірша – 2.
http://orcid.org/0000-0003-1811-9336.

Місце роботи автора:

Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського".
Тел.: 38 (044) 204 86 10.
E-mail: zharikov.eduard@acts.kpi.ua