УДК 004

О.Лівітчук, магістр гр. КІ-23Мз
Центральноукраїнський національний технічний університет

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ МАСШТАБОВАНОЇ РОЗПОДІЛЕНОЇ ФАЙЛОВОЇ СИСТЕМИ

У статті розроблено програмне забезпечення, яке призначено для системи масштабованої розподіленої файлової системи. Метою розробки є дослідження та програмна реалізація системи масштабованої розподіленої файлової системи. Об'єктом дослідження є процес масштабованої розподіленої файлової системи. Предметом дослідження є методи масштабованої розподіленої файлової системи. Методи дослідження базуються на методах файлових систем, методах математичної статистики, методах розробки програмного забезпечення. Результат роботи – програмна реалізація системи масштабованої розподіленої файлової системи. В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Постановка проблеми. Як економічно вигідне сховище неструктурованих або напівструктурованих даних підходять і програмно обумовлені рішення.

Для зберігання даних підприємствам потрібні добре масштабовані модернізуємі рішення, але зовсім не обов'язково здобувати дорогі системи SAN. Як економічно вигідне сховище неструктурованих або напівструктурованих даних підходять і програмно обумовлені рішення, які до того ж мають переваги в плані забезпечення лінійної масштабованості ємності й продуктивності.

Стрімке зростання обсягів даних є серйозною проблемою для багатьох підприємств. Підтвердженням цьому може служити розвиток середньостатистичного ЦОД протягом останніх шести років. Якщо припустити, що в 2023 році обсяг даних, що зберігаються в ньому, був дорівнює 100 Тбайт, а середній щорічний приріст становив 50% (цілком звичайний показник), то до 2028 року вони повинні були перетворитися в масив обсягом більше 1 Пбайт.

Однак подібне стрімке зростання спостерігається не для всіх видів даних. Відповідно до досвіду середніх і великих підприємств, за останні п'ять-шість років помітно змінилася пропорція між структурованими й неструктурованими даними. Протягом довгого часу обидва сегменти були приблизно рівні, а сьогодні близько 90% загального обсягу збереженої інформації доводиться на частку напівструктурованих і неструктурованих даних. І саме в цій області, як правило, спостерігається експонентний ріст.

Аналіз останніх досліджень і публікацій. При аналізі останніх досліджень і публікацій [1-10] було виявлено певні прогалини у забезпеченні системи масштабованої розподіленої файлової системи.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи масштабованої розподіленої файлової системи.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- Огляд існуючих систем масштабованої розподіленої файлової системи.
- Дослідження системи масштабованої розподіленої файлової системи.
- Програмна реалізація системи масштабованої розподіленої файлової системи.

Об'єктом дослідження є процес масштабованої розподіленої файлової системи.

Предметом дослідження є методи масштабованої розподіленої файлової системи.

Методи дослідження базуються на методах файлових систем, методах математичної статистики, методах розробки програмного забезпечення.

Виклад основного матеріалу. Обмін файлами може означати розподіл доступу до фізичних або електронних файлів. Хорошим прикладом є передача документів або файлів певному персоналу організації для спільної мети вручну або на пристрої зберігання, такі як флеш-диск, компакт-диск або жорсткий диск.

Однак спільний доступ до файлів майже завжди означає спільний доступ до файлів у мережі, навіть якщо вона знаходиться в невеликій локальній мережі (LAN) [2]. Це практика розповсюдження або надання доступу до цифрових медіа, таких як комп'ютерні програми, мультимедіа (аудіо, зображення та відео), документи або електронні книги [3]. Це публічний або приватний спільний доступ до комп'ютерних даних або простору в мережі з різними рівнями привілеїв доступу [2]. Отже, спільний доступ до файлів включає дві або більше осіб (точніше кажучи, комп'ютери) для доступу або використання файлу для читання чи запису, або обох. Таким чином, він підкреслює практику обміну або надання доступу до цифрової інформації чи ресурсів, включаючи документи, мультимедіа (аудіо/відео), графіку, комп'ютерні програми, зображення та електронні книги. Це приватний або публічний розподіл даних або ресурсів у мережі з різними рівнями привілеїв спільного використання.

Розподілена файлова система (DFS) підтримує обмін інформацією у формі файлів по всій інтрамережі. І це дозволяє користувачам зберігати та отримувати віддалені файли як локально, але з будь-якого комп'ютера в інтрамережі. Таким чином, це програма на основі клієнта/сервера, яка дозволяє клієнтам отримувати доступ і обробляти дані, що зберігаються на сервері, як на їх власному комп'ютері [4] [5]. Коли користувач отримує доступ до файлу на сервері, сервер надсилає користувачеві копію файлу, яка зберігається в кеші на комп'ютері користувача під час обробки даних, а потім повертається на сервер. В ідеалі розподілена файлова система організовує служби файлів і каталогів окремих серверів у глобальний каталог таким чином, щоб віддалений доступ до даних не залежав від місця розташування, а був ідентичним для будь-якого клієнта. І всі файли стають доступними для всіх користувачів глобальної файлової системи, а організація є ієрархічною та базується на каталозі. Він пропонує нам файлову систему, яка зберігає свої дані на сервері, і до її даних можна отримати доступ і використовувати їх так, ніби вони знаходяться на локальному вузлі. Розподілена файлова система дозволяє зручно обмінюватися інформацією та файлами між користувачами в мережі контрольованим і авторизованим способом. А сервер дозволяє користувачам клієнта обмінюватися файлами та зберігати дані так само, як вони зберігають інформацію локально. Однак сервер має повний контроль над даними та надає контроль доступу клієнтам.

Фактори, що впливають на продуктивність розподіленої файлової системи порівняно з традиційною системою клієнт/сервер

Мета розподіленої файлової системи (DFS) полягає в тому, щоб дозволити користувачам фізично розподілених комп'ютерів спільно використовувати дані та ресурси зберігання за допомогою спільної файлової системи [5]. І, на відміну від традиційних рішень клієнт/сервер, у розподіленій файловій системі можна досягти кращої продуктивності. Фактори, які впливають на кращу продуктивність розподіленої файлової системи, включають:

1. Замість того, щоб зберігати дані на одному сервері, дані можна зберігати на кількох вузлах. Це відомо як реплікація.

2. Система завжди доступна щоразу, коли до неї підключається клієнт. Це надійність.

3. Система має можливість обслуговувати запити клієнтів при кожному вході в екземпляр. Це називається доступністю.

4. Дані не втрачаються, але захищені, оскільки вони не зберігаються на одному сервері.

У порівнянні з традиційною системою клієнт/сервер, де дані зберігаються на одному сервері, унікальна функція безпеки розподіленої файлової системи полягає в тому, що

важливі або часто потрібні дані в розподіленій файловій системі можуть зберігатися на кількох вузлах (вузли означає комп'ютер, що працює в розподілена файлова система) [6]. Це називається «реплікацією». Реплікацію можна використовувати для досягнення кращої продуктивності системи та/або «надійності» системи. Дані в розподіленій файловій системі більш захищені від збою вузла. Якщо один або кілька вузлів виходять з ладу, інші вузли можуть забезпечити всі функції. Ця властивість також відома як «доступність» або «надійність». Різниця між доступністю та надійністю проста: доступність означає, що система може обслуговувати запит клієнтів у момент, коли клієнт підключається до системи. Надійність означає, що система доступна весь час, коли до неї підключаються клієнти. Файли також можна переміщувати між вузлами. Зазвичай це викликає адміністратор, і це робиться для покращення балансування навантаження між вузлами. Користувачі не повинні знати, де розташовані служби, а також передача з локальної машини на віддалену також має бути прозорою. У розподіленій файловій системі ця властивість відома як прозорість. Якщо ємності вузлів недостатньо для зберігання файлів, до існуючої розподіленої файлової системи можна додати нові вузли, щоб збільшити її ємність. І це відомо як «масштабованість». Зазвичай клієнт зв'язується з розподіленою файловою системою за допомогою локальної мережі (LAN).

Ключові характеристики розподіленої файлової системи

Очікується, що розподілена файлова система матиме три основні функції, які забезпечать надійне та захищене середовище обміну файлами серед багатьох інших; а саме прозорість, відмовостійкість і масштабованість.

1. Прозорість: користувачі повинні отримувати доступ до системи незалежно від місця входу, мати можливість виконувати однакові операції з розподіленою файловою системою та локальною файловою системою, і не повинні піклуватися про помилки через розподілену природу файлової системи; завдяки механізмам відмовостійкості. Крім того, Transparency можна розглядати з точки зору продуктивності. Таким чином, маніпуляції з даними мають бути принаймні такими ж ефективними, як у звичайних файлових системах. Іншими словами, складність основної системи має бути прихована від користувачів: кінцевому користувачеві не потрібно знати, як спроектована система, як дані мають бути розташовані та доступні, і як виявляються несправності [7]. І функції, які забезпечують прозорість:

– Іменування: це зіставлення між локальною назвою та фізичним розташуванням даних. Наприклад, у класичній файловій системі клієнти використовують логічне ім'я (текстове ім'я) для доступу до файлу, який відображається на фізичних блоках диска. А в розподіленій файловій системі потрібно додати імена серверів, що містять диск, на якому зберігаються дані. Розподілена файлова система повинна дотримуватися прозорості розташування: деталі того, як і де зберігаються файли, приховані для клієнтів. Крім того, може існувати кілька копій файлів, тому зіставлення має повертати набір розташувань усіх доступних копій. Розподілена файлова система має бути незалежною від розташування: логічне ім'я не повинно змінюватися, навіть якщо файл переміщується в інше фізичне розташування. Для цього використовуються таблиці розподілу або складні алгоритми для забезпечення глобальної структури простору імен, яка є однаковим простором імен для всіх клієнтів.

– Інтерфейси доступу до даних: клієнти можуть створювати, читати, писати, видаляти файли, не замислюючись про складні механізми основної системи, яка виконує операції, і має бути забезпечений доступ до системи за допомогою простих інструментів. Ось кілька прикладів таких інструментів:

I) Інтерфейс командного рядка (CLI), який використовується для доступу до файлів за допомогою традиційних команд UNIX.

II) Java, CC $^{++}$, інші мови програмування та REST (веб-інтерфейс) API можна використовувати для графічного інтерфейсу, такого як провідник вікон.

III) Користувачам можна дозволити монтувати (приєднувати) віддалені каталоги до їх локальної файлової системи: таким чином, доступ до віддалених файлів так, ніби вони зберігаються на локальному пристрої. Деякі приклади механізму FUSE або команди монтування UNIX.

– Кешування: це техніка, яка полягає в тимчасовому зберіганні запитуваних даних у пам’яті клієнта. Розподілені файлові системи використовують кешування, щоб уникнути додаткового мережевого трафіку та споживання ЦП, викликаного повторними запитами до того самого файлу, і таким чином підвищити продуктивність. Коли дані потрібні вперше, з сервера, який зберігає їх, робиться копія в основну пам’ять клієнта. Таким чином, для кожного наступного запиту цих даних клієнт буде використовувати локальну копію, уникаючи зв’язку з сервером і доступу до диска. Ця функція пов’язана з прозорістю продуктивності, оскільки запити можуть виконуватися швидко, приховуючи передачу даних користувачам за допомогою цієї техніки. Однак, коли дані змінюються, модифікація повинна бути передана серверу та будь-якому іншому клієнту, який кешував дані [7].

Таким чином, кеш-пам’ять у комп’ютерній системі розглядається як компонент, який зберігає запитувані дані, отже, може потенційно використовуватися в майбутньому [6]. Коли запитуються кешовані дані, час відповіді коротший, ніж коли дані не зберігаються в кеші та їх потрібно завантажити. Кешовані дані можна зберігати в оперативній пам’яті для швидкого доступу та/або на жорсткому диску. Кеш-пам'ять може бути з обох сторін зв'язку. На стороні сервера кеш зазвичай знаходиться в оперативній пам'яті. На стороні клієнта кеш може розташовуватися в оперативній пам'яті або на жорсткому диску. На стороні сервера, якщо вміст файлу кешується, механізм кешу економить час, оскільки немає необхідності отримувати доступ до вмісту файлу з жорсткого диска. На стороні клієнта, якщо клієнт запитує файл, механізми кешу економлять час, оскільки немає необхідності спілкуватися з сервером. Кешування на стороні клієнта також іноді називають реплікацією, ініційованою клієнтом. Клієнтський кеш також може надавати так званий офлайн-кеш. Це означає, що клієнт може отримати доступ до файлу з кешу після відключення від сервера. Автономний кеш часто зберігається на жорсткому диску; Механізм автономного кешування використовується в CODA.

– Виявлення несправностей: відмовостійку систему не слід зупиняти у разі тимчасових або часткових відмов. Розглянуті помилки – це збої мережі та сервера, які роблять служби даних недоступними, цілісність і узгодженість даних, коли кілька користувачів одночасно отримують доступ до даних. Іншими словами, це здатність виявляти перевантажені сервери, виправляти поведінку сервера або пошкоджені дані та приймати рішення щодо усунення цих несправностей. У розподіленій файловій системі помилки повинні бути виявлені системою з використанням мінімальних ресурсів перед виправленням, щоб користувачі не знали, що такі помилки виникають. Усі машини спілкуються разом у прозорий спосіб, обмінюючись невеликими повідомленнями. Наприклад, звіт дозволяє серверам, що керують простором імен, знати, які дані зберігаються на якому сервері. Оскільки дані постійно переміщуються, це дозволяє системі визначити, які дані втрачено та потребують переміщення або повторного копіювання, коли сервер стає недоступним або перевантаженим. Ще одне повідомлення – це серцеві сигнали, які використовуються для підтвердження доступності сервера. Якщо якийсь час не надсилає серцеві сигнали, він переміщується в карантин, а звітні повідомлення використовуються для застосування правильного рішення [7].

2. Стійкість до відмов: помилками вважаються збої мережі та сервера, які роблять дані та служби недоступними, цілісність і узгодженість даних, коли кілька користувачів одночасно отримують доступ до даних. Тому відмовостійку систему не слід зупиняти у разі цих тимчасових або часткових збоїв: у розподілених файлових системах збій мережі та сервера є нормою, а не винятком. Інструменти повинні бути розгорнуті для підтримки та забезпечення постійної доступності даних, щоб гарантувати обробку запитів у разі помилок.

Необхідно також враховувати цілісність і узгодженість даних, оскільки передбачені такі механізми, як кешування або реплікація.

Крім того, деякі функції для забезпечення відмовостійкості:

– Політика реплікації та розміщення: для того, щоб дані були завжди доступними, навіть якщо сервер виходить з ладу, розподілені файлові системи використовують реплікацію файлів шляхом створення кількох копій даних на різних серверах. Коли клієнт запитує дані, він прозоро отримує доступ до однієї з копій. Щоб підвищити відмовостійкість, репліки зберігаються на різних серверах відповідно до політики розміщення. Наприклад, репліки можна зберігати на різних вузлах, на різних доріжках, у різних географічних місцях, так що, якщо в будь-якому місці системи виникне збій, дані залишаться доступними.

– Синхронізація: у розподілених файлових системах необхідно враховувати синхронізацію між копіями даних. Коли дані перезаписуються, усі їх копії необхідно оновити, щоб надати користувачам останню версію даних. Існує три основні підходи:

I) У синхронному методі будь-який запит щодо змінених даних блокується, доки не буде оновлено всі копії. Це забезпечує користувачам доступ до останньої версії даних, але затримує виконання запитів.

II) У другому методі, який називається асинхронним, запити щодо змінених даних дозволені, навіть якщо копії не оновлені. Таким чином, запити можуть бути виконані в розумний час, але користувачі зможуть отримати доступ до застарілої копії.

III) Останній підхід є компромісом між першими двома в напівасинхронному методі, запити блокуються, доки деякі копії, але не всі, будуть оновлені. Наприклад, припустімо, що є файлові копії даних, запит щодо цих даних буде дозволено після оновлення трьох копій. Це обмежує можливість доступу до застарілих даних, одночасно зменшуючи затримку для виконання запитів.

– Узгодженість кешу: це та ж проблема, що й синхронізація – як оновити всі копії даних у кеші, коли одна з них змінена. Дані можна кешувати для покращення продуктивності системи, що може призвести до неузгодженості між копіями, коли користувач змінює одну з них. Ці зміни потрібно поширити на всі копії та дані в кеші, щоб надати користувачам їх актуальну версію. Щоб уникнути цієї проблеми, використовуються різні підходи:

I) Лише запис/читання багатьох (WORM): це перший підхід для забезпечення лише узгодженості. Створений файл не можна змінювати. Кешовані файли знаходяться в режимі лише для читання. Тому кожне зчитування відображає останню версію даних.

II) Другим методом є транзакційне блокування, яке полягає в отриманні блокування читання даних запиту, щоб будь-який інший користувач не міг виконати або записати дані, або оновити блокування запису, щоб запобігти будь-якому читанню або запису на диск. Таким чином, кожне читання відображає останній запис, і кожен запис виконується по порядку.

III) Іншим підходом є лізинг: це контракт на обмежений період між сервером, на якому зберігаються дані, та клієнтом, який запитує ці дані для запису. Оренда надається, коли запитуються дані, і під час оренди клієнту гарантується, що жоден інший користувач не зможе змінити дані. Дані знову доступні після закінчення терміну оренди або якщо клієнт відмовляється від свого права. Для майбутніх запитів на читання кеш оновлюється, якщо дані були змінені. Для майбутніх запитів на запис клієнту надається оренда, якщо це дозволено (тобто для цих даних не існує оренди або права звільняються).

– Балансування навантаження: це можливість автоматичного балансування системи після додавання або видалення серверів. Повинні бути надані інструменти для відновлення втрачених даних, їх зберігання на інших серверах або переміщення з головного пристрою на щойно доданий. Зв'язок між машинами дозволяє системі виявляти збої сервера та його перевантаження. І щоб виправити ці помилки, сервери можна додавати або видаляти. Коли сервер видаляється із системи, останній повинен мати можливість відновити втрачені дані та зберігати їх на інших серверах. Коли сервер додається до системи, необхідно надати

інструменти для переміщення даних із хост-сервера на щойно доданий сервер. Користувачі не повинні знати про цей механізм. Зазвичай розподілені файлові системи використовують запланований список, до якого вони розміщують дані для переміщення або повторного копіювання. Періодично алгоритм повторює цей список і виконує належну дію. Наприклад, CEPH використовує функцію «Контрольована реплікація під масштабованим хешуванням» (CRUSH) для випадкового зберігання нових даних, переміщення об'єкта існуючих даних до нових ресурсів зберігання та рівномірного відновлення даних із видалених ресурсів зберігання.

3. Масштабованість: це здатність ефективно використовувати велику кількість серверів, які динамічно та постійно додаються в систему [7]. Всупереч загальним відомостям про розподілену файлову систему, це означає, що ця система може залучати більше ніж один сервер у локальній мережі для кращої продуктивності спільного використання файлів. Практичним прикладом є ситуація, коли два або більше серверів використовуються в мережевому середовищі: коли один сервер у внутрішній мережі не працює, очікується, що операції та служби розподіленої файлової системи не припиняться, а синхронізуються та віртуально передаються на інший сервер. (s); отже, масштабованість. Таким чином, масштабована система розподіленого обміну файлами стала можливою завдяки децентралізації файлів сервера в інтрамережі, щоб вони не зберігалися лише на одному сервері.

Реалізація масштабованості в системі розподіленого обміну файлами

Як зазначалося раніше щодо більш детального вивчення цього відповідного технологічного рішення, розподілена файлова система (DFS) показує, що вона представляє низку проблем, які не можна ігнорувати. І ключовою проблемою, яку він представляє, є масштабованість (особливо, коли сервер потрібно підключити або відключити). Таким чином, додаткові фактори, які слід враховувати при реалізації масштабованості, включають:

1. Кількість серверів: пам'ятайте, що масштабованість передбачає, що в розподіленому середовищі існує більше однієї серверної системи, і жоден клієнт не повинен бути недоступним, коли сервер у мережі виходить з ладу, оскільки інші сервери все ще працюють і файли копіюються на кожен з них. Таким чином, кількість серверів забезпечує і підтримує доступність і надійність внутрішньої мережі.

2. Операційна система: серед інших мережевих операційних систем рекомендується LINUX, враховуючи її надійність і захист від вірусів.

Характеристики масштабованої розподіленої файлової системи

Масштабована система клієнт/сервер має такі характеристики

1. Клієнти більше не страждають від внутрішніх проблем, таких як збій сервера.

2. До зовнішньої програми можна отримати доступ на будь-якій комп'ютерній системі в мережі; отже, подолаючи проблему небажаного та неавторизованого доступу.

3. Безпека мережевих (або організаційних) даних гарантується.

4. Клієнти зможуть легко передавати свої файли в мережевому середовищі.

Серверні механізми розроблені для зв'язку між усіма комп'ютерами в мережі; отже, пропонування та забезпечення прозорості.

З огляду на швидке зростання завдань/робочого навантаження та робочих процесів у наших сучасних установах, а також проблеми, які постають перед існуючими системами обміну файлами та їхніми моделями контролю доступу в Інтернеті, існує потреба у файловій системі з можливістю масштабування. У наші дні великі установи бажають мати автоматизований робочий процес, який допоміг би усунути виснажливі ручні процеси спілкування та обміну файлами/поштою. Таким чином, потенційним кандидатом є масштабована розподілена система обміну файлами. Серед усіх файлових систем мережева файлова система (NFS) є найбільш перевіреною технікою. У цій роботі кожен користувач/клієнт у мережі може отримати доступ до мережевої системи організації так, ніби файли знаходяться в його локальній системі. Крім того, для досягнення масштабованості розподіленої файлової системи можна використовувати два або більше

серверів. Таким чином, інституційний робочий процес не припиняється, коли сервер виходить з ладу, оскільки існує резервний сервер, який забезпечує роботу мережевої системи/середовища.

Наразі ця стаття досягла поставленої мети, яка полягає в тому, щоб запропонувати реалізацію функції «масштабованості» розподіленої системи спільного використання файлів як вирішення проблем розподіленої файлової системи (DFS), згаданих раніше.

У цій роботі представлено перехід від традиційної централізованої системи зберігання та поганої файлової системи (мережевих дисків) до масштабованої розподіленої файлової системи. Проблему масштабованості, особливо коли сервер потрібно монтувати або демонтувати, і повільне транспортування файлів можна усунути за допомогою впровадження децентралізованої серверної системи за допомогою двох або більше серверів, які працюють синхронно. А коли масштабованість буде досягнута, клієнти більше не будуть страждати від внутрішніх проблем, таких як збій сервера. Система також гарантує легшу/швидшу транспортування файлів між клієнтськими системами.

Розробка структурної схеми

Багато додатків працюють швидше й ефективніше, якщо використовується програмно-визначаєме сховище (SDS), що відокремлює програмне забезпечення від устаткування систем зберігання. Такі рішення універсальні, сприяють активізації інноваційних розробок і дозволяють зробити крок у майбутнє, адаптуючи архітектуру сховища до потреб робочих навантажень.

Деякі програмно-визначаємі підходи ставлять вас у залежність від спеціалізованого ПЗ або неадаптуємих архітектур. Корпорація Dell, навпроти, пропонує широкий вибір сучасних SDS-Рішень і дозволяє вам використовувати встаткування й ПЗ, що підходить для ваших конкретних додатків. Завдяки цьому ви одержуєте розширений контроль над ресурсами, а прискорений доступ до апаратних і програмних інновацій дає вам можливість швидко впроваджувати нові IT-можливості**.**

Таких програмно-визначаємих рішень зберігання, які підходили б для будь-яких завдань, просто не існує. Тому, працюючи в партнерстві із провідними постачальниками ПЗ для систем зберігання, ми розробили адаптуємі відкриті рішення. Вони дозволяють зберегти існуючі капіталовкладення й розширити можливості ваших систем у майбутньому. Вибирайте з них ті, які підійдуть саме вам: ми пропонуємо як випробуваної й перевірені еталонні архітектури з мінімальним ступенем ризику, так і готові й зручні гіперконвергентні рішення, розгортання яких займає менш чверті години**.**

Швидше за все, програмно-визначаємі системи – це лише одна з тих інновацій, які ви плануєте для свого сховища. Наші платформи є по-справжньому програмно-визначаємими, тобто момент впровадження інновацій в організації визначаєте ви, а не наступний цикл відновлення встаткування. Розвивайтеся тоді, коли будете впевнені, що готово зробити крок уперед. Наприклад, серія Dell XC і розроблені Dell рішення для EVO:RAIL на базі процесорів Intel® Xeon® дозволяють впроваджувати нові функції в міру їхньої появи й розгортати нові вузли менш чим за хвилину.

Програмно-визначаємі сховища допомагають підвищити продуктивність і швидкість роботи за рахунок переносу функцій з устаткування масивів зберігання даних ближче до обчислювальних ресурсів. Інтеграція й інструменти автоматизації дозволяють зменшити складність адміністрування й заощадити робочий час. Що ви одержуєте в результаті? Інфраструктуру віртуальних робочих столів, розгортання якої займає в 6 разів менше часу при на 27% більше низькій сукупній вартості володіння. У порівнянні із традиційними підходами до сховища для інфраструктури віртуальних робочих столів, рішення здатні підтримувати подвоєна кількість користувачів, а місця при цьому займати на 91% менше.

Хоча підтримка будь-якого встаткування й кожного ПЗ вважається головною перевагою програмно-визначаємих систем, у реальності всі набагато складніше. Рішення власного складання типу "білий ящик" часто не виправдують очікувань. Установка програмно-визначаємого сховища на випробуване встаткування корпоративного класу з

підходящими конфігураціями дозволяє домогтися значно кращих результатів. Крім того, використовувати корпоративні сховища такого типу стає просто, як ніколи раніше, адже послуги підтримки світового класу дозволяють звертатися до фахівців з будь-яких питань із будь-якої крапки земної кулі.

Щоб упоратися з напливом неструктурованим і напівструктурованим даних, підприємствам потрібно масштабована життєздатна програмно обумовлена інфраструктура зберігання даних, здатна забезпечити високу доступність, економічну ефективність і простоту керування. Основою такого рішення служить добре масштабована розподілена файлова система, що при необхідності дозволяє здійснити лінійне розширення ємності ресурсів зберігання.

Лінійна масштабованість

Коли в контексті систем зберігання даних мова заходить про можливість їхнього розширення, як правило, мається на увазі так звана лінійна масштабованість – термін, що часто трактується неправильно. У теорії мається на увазі, що дворазове збільшення ємності зберігання забезпечує подвоєння продуктивності: при збереженні часу відгуку (при інших незмінних умовах), пропускна здатність (вимірювана в гігабайтах у секунду) повинна збільшитися теж у два рази. Однак на практиці традиційні системи зберігання не здатні виконати ця вимога.

Причина такої розбіжності криється в тім, що масштабованість систем зберігання залежить від безлічі факторів, і ємність сховищ – це лише один з них. Апаратне забезпечення системи зберігання, відповідальної за керування окремими жорсткими дисками, також повинне масштабуватися відповідним чином. Для оптимального використання всіх шпинделів жорстких дисків при пікових навантаженнях потрібний центральний процесор з достатньою продуктивністю. Крім цього, підтримка подвоєної ємності зберігання можлива тільки в тому випадку, коли файлова система й функція метаданих масштабуються лінійно – тоді система зможе встановити місцезнаходження даних на нових доданих дисках. Інакше кажучи, для забезпечення лінійної масштабованості систем зберігання даних необхідно, щоб продуктивність всіх компонентів збільшувалася пропорційно.

Для традиційних розподілених систем зберігання даних це означає, що кожному вузлу зберігання доводиться нести додаткові системні витрати на комунікацію з іншими вузлами при виконанні яких-небудь операцій з файлами. Оскільки ємності збільшуються швидше, ніж продуктивність, лінійного росту потужності системи досягти не вдається.

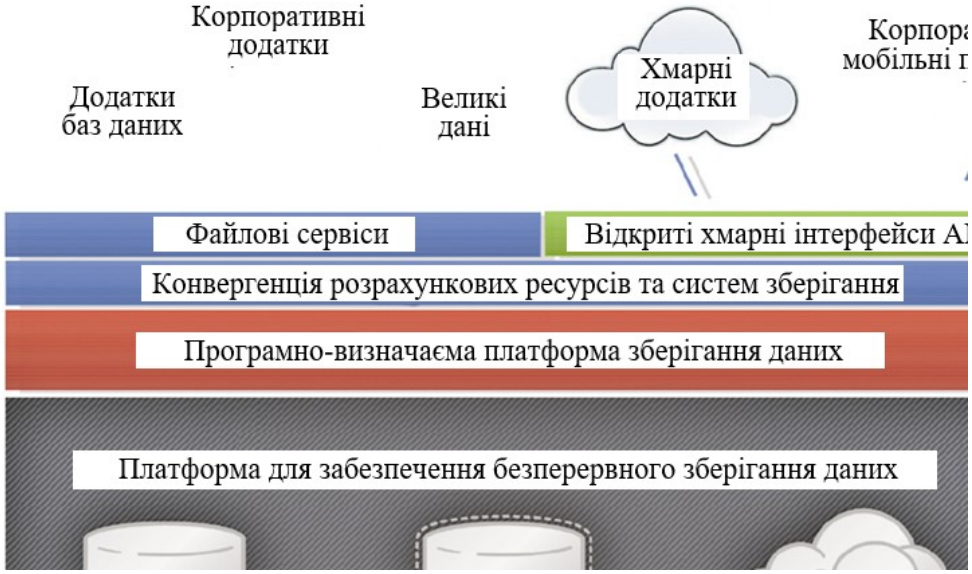

Рисунок 1 – Структурна схема системи

У випадку повністю програмно обумовлених масштабованих рішень зберігання даних ці обмеження не виникають – обсяги сховищ ростуть синхронно із продуктивністю (див.

Рисунок 1). У результаті створюється ефективна інфраструктура зберігання для розміщення напівструктурованих і неструктурованих даних на стандартних серверах x86. Завдяки спільному використанню процесорів і ресурсів уведення-виводу недорогих стандартних серверів формується великий високопродуктивний пул (кластер) зберігання даних. Тепер, якщо підприємству потрібно збільшити ємність зберігання, IT-фахівці можуть просто додати додаткові жорсткі диски, а для підвищення продуктивності – збільшити число серверів. Додаткових накладних витрат при цьому не виникає.

Програмно обумовлена інфраструктура

Для лінійного збільшення продуктивності і ємності таких рішень повинні дотримуватися три умови:

– відсутність сервера метаданих;

– ефективний розподіл даних, що зберігаються, для забезпечення високого ступеня масштабованості й надійності;

– паралельний доступ до даних для досягнення максимальної продуктивності в повністю розподіленій архітектурі.

У програмно обумовлених масштабованих рішеннях логічна й фізична локалізація даних є складним завданням для розроблювачів. У більшості розподілених систем ця проблема вирішується за допомогою окремого індексу, де втримуються імена файлів і метадані для їхньої локалізації. Але в результаті виникає критична крапка відмови (Single Point of Failure) і створюється вузьке місце для продуктивності: при росту кількості серверів, жорстких дисків і даних сервер метаданих починає гальмувати роботу всього рішення. Ситуація ще більше загострюється у випадку безлічі малих по розмірі файлів, тому що обсяг їх метаданих росте випереджальними темпами. У системі компанії Red Hat, приміром, це завдання вирішується за допомогою алгоритму хешування: для кожного ім'я файлу розраховується його хеш-сума. Це дозволяє усунути основне джерело зниження продуктивності операцій введення-виводу або навіть потенційну причину виникнення збоїв.

Одночасна підтримка зберігання файлів і об'єктів (File/Object Storage) в одному пулі зберігання також виявляється дуже корисною. Сполучення обох видів зберігання значно спрощує процес керування різними даними й забезпечує підприємствам більшу гнучкість при зберіганні корпоративної інформації, чим специфічні рішення SAN від різних виробників. Це досить вигідний спосіб протидії стрімкому зростанню обсягів напівструктурованих і неструктурованих даних.

Таким чином, програмно обумовлені рішення добре підходять для збереження різних структурованих даних, керування складним мультимедійним умістом і архівування в безпосередній близькості від робочого місця. Приміром, Posix-сумісне рішення від Red Hat підтримує такі стандарти NAS, як NFS і SMB, для забезпечення доступу до файлів і OpenStack Swift для доступу до об'єктів, а крім того, воно оснащено клієнтом Glusterfs для паралельного доступу.

Відсутність залежності від масивів зберігання даних

Якщо на передній план виходять напівструктурованого й неструктурованого дані, підприємства, що використовують повністю програмно обумовлене рішення, перестають залежати від дорогих і погано піддаються масштабуванню монолітних масивів зберігання даних. Таке рішення дозволяє в найкоротший термін уводити до ладу додаткові недорогі сервери x86 і додавати в інфраструктуру зберігання даних – будь те власний ЦОД підприємства або гібридна хмара – гарно масштабовану й високопродуктивну ємність. Синхронна реплікація даних підтримує їх локальне дзеркалювання й сприяє забезпеченню безперервності бізнес-процесів. Асинхронна реплікація даних, у свою чергу, дозволяє робити дистанційне копіювання даних для створення резервних копій на випадок аварійного відновлення.

Якщо підприємствам доводиться боротися з лавиноподібним ростом обсягів напівструктурованих і неструктурованих даних, то програмно обумовлене рішення дозволить одержати додаткову ємність із хмари. Так, інструмент керування Red Hat Storage

надає адміністраторам централізований огляд усього пула зберігання даних. Він базується на проекті Ovirt – відкритій платформі для керування інфраструктурою й віртуалізацією. Завдяки цьому адміністрування зростаючих обсягів інформації спрощується без необхідності інвестування засобів у нове апаратне забезпечення, що дозволяє впоратися з підривним збільшенням кількості даних у діапазоні від декількох тера- до багатьох петабайтів.

Висновки. У статті наведені теоретичне узагальнення й рішення наукового завдання дослідження методів масштабованої розподіленої файлової системи.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем масштабованої розподіленої файлової системи.

– Досліджена система масштабованої розподіленої файлової системи.

– На основі отриманих результатів досліджень створена програмна реалізація системи масштабованої розподіленої файлової системи.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання масштабованої розподіленої файлової системи. Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Список літератури

1. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.
2. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.
3. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.
4. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143
5. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.
6. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.
7. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.
8. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.
9. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.
10. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.
11. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.
12. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.
13. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 .

P.517-522.
14. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.
15. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.
16. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.
17. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.
18. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 3(73), С. 155-166.
19. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». Кібербезпека: освіта, наука, техніка. 2024. №3(23), С. 111-131.
20. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.
21. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.
22. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.
23. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.