УДК 004.9 **doi: 10.26906/SUNZ.2024.3.084**

А. С. Білоконь, С. О. Борисов, М. В. Усатенко, В. М. Федорченко

Харківський національний технічний університет радіоелектроніки, Харків, Україна

АНАЛІЗ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНИХ СИСТЕМ ОБРОБКИ ТА ЗБЕРІГАННЯ ДАНИХ

Анотація. Актуальність. Зі зростанням обсягу даних, генерованих користувачами, IoT пристроями, соціальними медіа, бізнес-процесами тощо, потреба в масштабованих рішеннях для зберігання стає все більш очевидною. Розподілені системи дозволяють ефективно масштабуватися, забезпечуючи збільшення обсягів зберігання і обчислювальної потужності без значних затрат. Сучасний бізнес вимагає високої доступності та надійності систем, оскільки навіть мінімальний час простою може призвести до значних фінансових втрат та зниження довіри клієнтів. Розподілені системи забезпечують високу доступність і витривалість, автоматично відновлюючи роботу після збоїв і реплікуючи дані для забезпечення їх цілісності. Глобалізація бізнесу вимагає роботи з даними у різних географічних локаціях. Розподілені системи дозволяють локалізувати зберігання та обробку даних ближче до кінцевих користувачів, знижуючи затримки і підвищуючи загальну продуктивність систем. Збільшення загроз безпеці та посилення нормативних вимог до захисту даних змушують організації шукати більш надійні рішення для зберігання даних. Розподілені системи пропонують розширені можливості для шифрування даних, регулювання доступу, аудиту та відповідності нормативним актам. Для обробки великих обсягів даних часто потрібні великі обчислювальні потужності. Розподілені системи зберігання даних ідеально підходять для роботи в парі з розподіленими обчисленнями, такими як обробка потокових даних, машинне навчання, великі дані, дозволяючи ефективно розподіляти завдання і обробляти великі обсяги інформації.Серед викликів, з якими можуть стикатися розподілені системи зберігання даних, - забезпечення консистентності даних між вузлами, управління затримками мережі, захист даних і забезпечення безпеки. Для вирішення цих викликів застосовуються різні стратегії та технології, включаючи алгоритми консистентного хешування, реплікацію даних, транзакційні протоколи з гарантією атомарності, консистентності, ізоляції та довговічності та моделі послідовної консистентності. Таким чином, в умовах постійного зростання обсягів даних та збільшення вимог до їх обробки, розподілені системи зберігання даних є ключовим елементом інфраструктури будь-якої організації, що прагне до інновацій та ефективності. **Метою даної роботи** є аналіз функціонування розподілених систем обробки та зберігання даних. **Об'єктом дослідження** є розподілені системи обробки та зберігання даних. **Предметом дослідження** є архітектурні рішення розподілених систем зберігання та обробки даних. **Результати.** Проведено аналіз функціонування розподілених систем обробки та зберігання даних. Вибір архітектурного рішення для розподіленої системи залежить від специфіки задач, які необхідно вирішити, вимог до продуктивності, масштабованості, надійності та доступності. Зазвичай, ефективні розподілені системи використовують комбінацію зазначених підходів для досягнення оптимальних результатів. Вибір архітектурного рішення для розподілених систем - це комплексний процес, який вимагає балансування між технічними, бізнесовими та операційними вимогами. Врахування майбутнього зростання, потенційних викликів і гнучкості системи є ключовими факторами для забезпечення її довгострокового успіху.

Ключові слова: розподілена інформаційна система, обробка даних, SOA, P2P, Web-Scale, сховище даних, центр обробки даних, event-driven architectures.

Вступ

Розподілені системи зберігання даних – це комплексні інформаційні системи, які дозволяють зберігати дані в розподіленому вигляді по декількох фізично рознесених серверах чи вузлах [1]. Такі системи забезпечують високу доступність, масштабованість і надійність зберігання даних. Вони є основою для багатьох сучасних веб-сервісів, облачних платформ і корпоративних додатків. Система може бути легко розширена шляхом додавання нових вузлів, що дозволяє збільшувати обсяги зберігання та обчислювальні можливості без перерв у роботі. дані реплікуються між декількома вузлами, що забезпечує їх збереження у разі виходу з ладу одного чи декількох вузлів. Система забезпечує доступ до даних навіть при збоях окремих компонентів, завдяки реплікації даних і резервуванню. Розподілені системи намагаються забезпечити прозорість розміщення, виконання та міграції, щоб кінцевий користувач або додаток не відчував розподіленості ресурсів.

В теперішній час існує багато архітектурних рішення розподілених систем, які відіграють критичну роль у забезпеченні їх ефективності, масштабованості, надійності та доступності. Мікросервісна архітектура полягає у розбитті додатку на невеликі, незалежні сервіси, кожен з яких виконує певну функцію та спілкується з іншими через легковісні механізми, такі як HTTP/REST або черги повідомлень. Цей підхід дозволяє незалежно розгортати та масштабувати кожен сервіс, що підвищує гнучкість та відмовостійкість системи. SOA - це архітектурний стиль, який забезпечує взаємодію різних сервісів через веб-сервіси або API. Він спрямований на забезпечення високого рівня інтеграції та взаємодії між різноманітними застосунками і системами в розподіленому середовищі [2].

У архітектурі з шарами розподілена система організована у вигляді шарів, кожен з яких має свої відповідальності (наприклад, презентаційний шар, бізнес-логіка, доступ до даних). Це спрощує розробку та тестування, але може вплинути на продуктивність через додаткові затримки при переході між шарами. У peer-to-peer (P2P) архітектурі всі вузли є однорідними та можуть виконувати роль як клієнта, так і сервера [3]. Це дозволяє системі ефективно масштабуватися та витримувати високі навантаження, розподіляючи ресурси та завдання між вузлами. Архітектура, яка заснована на подіях, передбачає реагування компонентів системи на

події (наприклад, повідомлення про зміни даних), що дозволяє створювати високоефективні та гнучкі системи, здатні швидко адаптуватися до змін умов роботи. Слід відзначити, що розподілена система може одночасно забезпечити тільки дві з трьох характеристик: консистентність, доступність та стійкість до розподілу (переривань мережі). Вибір між цими характеристиками залежить від конкретних вимог до системи.

Отже**, метою цієї роботи** є аналіз функціонування розподілених систем обробки та зберігання даних, а також огляд існуючих архітектур.

Основна частина

Існуючі на сьогодні вимоги до проектування та розробки сховищ даних (СД), що забезпечують прийнятну надійність та продуктивність, зумовлюються об'єктивною тенденцією високого зростання обсягу даних, які потребують довготривалого зберігання. Фахівці багатьох організацій зазначають, що Web-Scale IT включає реалізацію різних підходів до побудови IT-інфраструктури. Основною відмінністю Web-Scale IT є реалізація стійкості до відмови на рівні програмного забезпечення і вимагає оптимізації коду під апаратне забезпечення. Слід відзначити, що відмова від будь-яких технологій віртуалізації вносить ще один шар взаємодії з апаратним забезпеченням (і, цілком імовірно, відповідні затримки). Апаратне забезпечення розробляється з урахуванням архітектури програмного забезпечення. , а також може бути інтегрованим або розподіленим на компоненти. архітектури, побудовані за Web-Scale IT, високо масштабуються, оскільки рівні продуктивності досягаються великою кількістю ефективних неспеціалізованих вузлів та використанням типових серверів. Сукупні ресурси багатьох цих вузлів об'єднуються в пул для створення обчислювальної матриці та розподілу відповідно до потреби. Обладнання Web-масштабу - відносно недороге та доступне. При відмові вузол просто замінюється. Керує цією мережею вузлів інтегроване програмне забезпечення, яке, на відміну від апаратних засобів управління, може перепрограмувати систему, щоб вона відповідала мінливій динаміці. Така система по суті своїй еластична, оскільки дані, метадані та операції розподіляються на велику кількість малих вузлів, що замінюються, і, можливо, навіть на кілька центрів обробки даних.

Ключовими елементами архітектурного рішення СД з Web-Scale IT є: орієнтація використання центрів обробки даних, використання веб-орієнтованої чи мікросервісної архітектури надання послуг; наявність програмованого управління СД; можливість налаштування будь-яких бізнес-процесів, орієнтованих високу продуктивність; застосування методології «корпоративна культура організації, що навчається», сфокусована на інноваціях і безперервному навчанні.

Також виділяються два підходи підходи до забезпечення масштабованості: scale-up та scale-out. Вертикально ¬масштабовані системи (scale-up), яким властиве збільшення кількості доступних ресурсів за рахунок збільшення потужності застосовуваних обчислювальних засобів, та горизонтально-масштабувані системи (scale-out), що характеризуються наявністю значного числа невеликих серверів, об'єднаних у кластер, на яких розподіляється поставлена замовником прикладне завдання. Незважаючи на різницю, scale-up і scale-out зазначені підходи можуть комбінуватися в рамках однієї СД. Однак масштабування до великої кількості вузлів, у тому числі з можливим розподілом по географічно рознесеним ЦОД, може призвести до проблеми забезпечення доступності та консистентності даних.

В даний час найбільш характерними представниками горизонтально-масштабованих систем, спроектованих згідно з Web-Scale IT, є СД компаній EMC Isilon та IBM XIV. Кластер горизонтально ¬масштабованої системи, зазначених рішень складається з серверів, що включають процесори, кеш-пам'ять, диски, Ethernet-порти засобів підключення . Крім даних пристроїв, вузол EMC Isilon містить також порти Infiniband RDMA, за допомогою яких сервери об'єднуються у внутрішню мережу з метою забезпечення стійкості до відмови кластера [4]. Загальною для цих горизонтально масштабованих систем є одиниця масштабування, що є апаратним модулем з фіксованою кількістю приєднаних до нього дисків. Основні відмінності цих систем одна від одної полягають в особливостях організації доступу до даних та способу їх зберігання. В одному випадку забезпечується доступ до даних через організацію файлової системи, в іншому випадку використовується блоковий пристрій. Всі дані розподілені між усіма вузлами даних СД, і будь-який вузол може на рівних обслуговувати запити до будь-якого файлу, незалежно від того, на дисках або кеші якого вузла знаходяться дані. Тим самим досягається висока продуктивність роботи з будь-яким файлом.

Класичні платформи зберігання SAN та NAS, хоч і критично важливі для корпоративних додатків, але не призначені для роботи з сучасними хмарними додатками та не відповідають вимогам масштабування у хмарі [5]. Крім того, постійне зростання обсягів неструктурованих даних вимагає створення більш простої архітектури зберігання, яка призначена для управління мільярдами та трильйонами файлів, прискорює розробку хмарних та мобільних додатків та додатків для великих даних і при цьому дозволяє знизити витрати, пов'язані із зберіганням, та витрати в цілому. Щоб вирішити ці завдання, IT-служби та постачальники послуг почали тестувати та впроваджувати економічні типові та відкриті інфраструктури. Типові компоненти та відкриті технології на базі стандартів знижують витрати на зберігання, але окремо забезпечують нижчі показники продуктивності та надійності. Також слід пам'ятати, що інфраструктура зручна в обслуговуванні, якщо IT-фахівці замовника мають достатній досвід її експлуатації.

Безпрецедентне зростання обсягів даних (структурованих та неструктурованих) призводить до того, що файли великих обсягів (наприклад, зображення, відео тощо) зберігаються у дорогих ізольованих СД. У класичній інфраструктурі, зазвичай, часто дані ізолюється, ускладнюючи цим обмін і управління ними і збільшуючи витрати. Зазначений фактор не підтримує ефективне та економічне масштабування, що змушує замовників шукати рішення, яке поєднує в собі переваги публічної та приватної хмари. Рішення дозволяють будь-якій організації об'єднати кілька

систем зберігання та архівів змісту в єдиний глобально доступний та ефективний репозиторій змісту, який може обслуговувати велику кількість програм.

Покладена в основу рішення архітектура типу «активний-активний» з підтримкою кількох майданчиків, єдиний глобальний простір імен та універсальна доступність забезпечують доступ до змісту з будь-якої точки для будь-якої програми чи пристрою. Завдяки розподілу контейнерів даних між майданчиками операції запису та читання можуть виконуватися в будь-якій точці світу. Рішення забезпечують високий рівень семантичної узгодженості, що дозволяє спростити розробку додатків та отримувати доступ до даних із будь-якої точки. Також у рішенні є функція геокешування, яка визначає схеми доступу з декількох майданчиків та кешує дані на майданчику, де ці дані використовуються найчастіше.

У загальноприйнятих класичних інфраструктурах зберігання даних, як правило, запровадити системи бізнес-аналітики може бути непросто. Так, дані часто розподілені між кількома складними системами, і ускладнює та збільшує витрати на доступ та управління. Організації змушені отримувати дані від операційних систем і потім завантажувати їх у виділений кластер для аналізу. В основі ефективної аналітики лежать точність та своєчасність даних. Рішення надає замовникам можливість видобувати практично значущі дані бізнес-аналітики з великих обсягів розподіленого змісту, не використовуючи робочого процесу ETL (Extract, Transform, Load). Рішення можуть підтримувати озеро даних, надаючи можливості доступу та управління будь-яким змістом організації за допомогою кількох дистрибутивів Hadoop [6].

При використанні звичних СД зі зростанням обсягів даних неминуче зростає кількість контролерів, дискових груп, томів, файлових систем, так чи інакше з'являються «острівці» різнорідних систем, стає все складніше та складніше підтримувати ефективність їх використання. Нарощування систем вимагає попереднього аналізу, складних робіт з конфігурації та міграції даних [6]. При проєктуванні рішень СД, їх загальна продуктивність визначається продуктивністю модуля управління, вплинути на який з боку персоналу (адміністраторів) практично неможливо. А ось перспективнішим варіантом управління продуктивністю є виділення двох рівнів масштабування: масштабування за ресурсами, яке виконується управлінням кількості апаратно-керованих масивів жорстких дисків, і масштабування за продуктивністю - управлінням кількістю апаратних блоків підготовки та обробки інформації. Далі перейдемо до існуючих рішень. Аналіз існуючих мікросервісних архітектур розподілених систем виявляє цілий ряд ключових компонентів, практик та викликів, які є актуальними для більшості сучасних високонавантажених та високодоступних систем.

Netflix є одним з найвідоміших прикладів успішного застосування мікросервісної архітектури. Система розподілена на сотні мікросервісів, які відповідають за різні аспекти платформи, від потокового відтворення до рекомендацій контенту та обробки користувацьких даних. Платформа базується на використанні контейнерізації, оркестрації, внутрішньої мережевої інфраструктури для забезпечення зв'язку між сервісами (Netflix Zuul як API Gateway).

Amazon перетворив свою монолітну архітектуру на мікросервісну, щоб забезпечити масштабування та гнучкість своєї величезної електронної комерції та хмарних сервісів. Даний сервіс використовує AWS для розгортання та управління мікросервісами, таких як Elastic Container Service (ECS), Lambda для безсерверних обчислень, та інших сервісів для моніторингу, логування та автоматизації.

Обробкою величезної кількості запитів з мінімальною затримкою, зберігання та аналіз великих обсягів даних займається Google, який використовує мікросервісну архітектуру для різних своїх продуктів, включаючи пошук, Gmail, Maps. Їхня система розроблена для високої продуктивності, масштабування та надійності. В роботі використовуються власні технологічні розробки для управління розподіленими системами, таких як Borg для оркестрації контейнерів, та інших інструментів для мікросервісів, моніторингу, логування.

Uber перейшов на мікросервісну архітектуру, щоб впоратися зі швидким зростанням та потребами в глобальному масштабуванні. Сервіс використовує георозподілених мікросервісів, розробка власних рішень для геопросторових обчислень, інтенсивне застосування контейнеризації та оркестрації.

Таким чином, кожен мікросервіс управляє своїми даними незалежно, що забезпечує гнучкість та високу доступність. Інтенсивне використання CI/CD для автоматизації тестування, розгортання та моніторингу. Використання контейнерів та оркестрації типу Docker, Kubernetes та інших інструментів значно спрощують розгортання, масштабування та управління мікросервісами. Забезпечення видимості та прозорості всієї системи через централізоване логування та моніторинг. Використання API шлюзів для спрощення зв'язку між мікросервісами та клієнтами, а також для забезпечення безпеки, маршрутизації запитів, обробки помилок. Мікросервісна архітектура дозволяє створювати високоадаптивні та масштабовані системи, здатні ефективно вирішувати завдання різної складності.

Сервісно-орієнтована архітектура (SOA) для розподілених систем орієнтована на використання вебсервісів та інших технологій для забезпечення модульності, масштабованості, та гнучкості систем. У таких системах компоненти, або "сервіси", надають стандартизовані інтерфейси для взаємодії, що дозволяє легко інтегрувати та перевикористовувати їх у різних додатках та бізнес-процесах. AWS надає широкий спектр хмарних сервісів, кожен з яких може вважатися компонентом у SOA. Сервіси охоплюють обчислення, зберігання даних, бази даних, мережеві функції, і багато іншого, дозволяючи розробникам легко інтегрувати потрібні компоненти у свої додатки. Salesforce використовує SOA у своїй платформі для керування відносинами з клієнтами. Через API Salesforce надає доступ до своїх сервісів, таких як продажі, обслуговування клієнтів, маркетинг тощо, що дозволяє інтегрувати ці сервіси з іншими бізнес-додатками. SAP пропонує комплексні бізнес-рішення, які також базуються на SOA, дозволяючи компаніям оптимізувати свої бізнес-процеси. SAP

NetWeaver дозволяє інтегрувати дані та бізнес-процеси з SAP додатками та зовнішніми системами.

SOA спрощує інтеграцію різних сервісів і систем, незалежно від їхньої реалізації та місця розташування, завдяки використанню стандартизованих протоколів і форматів даних. Системи, побудовані за принципами SOA, складаються з незалежних сервісів, які можуть бути розроблені, розгорнуті, і вдосконалені окремо один від одного. SOA дозволяє легко масштабувати системи, додаванням або модифікацією сервісів, для задоволення зростаючих потреб бізнесу. Розподіленість сервісів у SOA сприяє відмовостійкості системи, дозволяючи їй продовжувати роботу навіть у разі збоїв окремих компонентів.

Таким чином, SOA залишається важливим підходом у розробці розподілених систем, оскільки він пропонує значну гнучкість та ефективність при інтеграції різноманітних сервісів та додатків. Водночас успіх впровадження SOA залежить від уважного планування, управління змінами, та заходів безпеки.

Архітектура з шарами, або багаторівнева архітектура, є традиційним підходом до проектування розподілених систем, який включає розділення функціоналу програми на окремі рівні, кожен з яких має свою специфічну відповідальність. Такий підхід дозволяє досягти модульності, спрощує розробку та тестування, а також полегшує масштабування та адміністрування системи. Презентаційний шар відповідає за взаємодію з користувачами системи, представлення даних та обробку введення користувача. Веб-інтерфейси, мобільні додатки чи настільні застосунки є прикладами компонентів цього шару.Бізнес-логіка (Business Logic Layer): Містить основну функціональність системи, виконує обробку даних згідно з правилами бізнесу. Цей шар взаємодіє з презентаційним шаром та шаром доступу до даних, обслуговуючи їх запити. Шар доступу до даних забезпечує абстракцію та управління доступом до даних, збережених у базі даних або інших сховищах. Це може включати виконання запитів до бази даних, обробку транзакцій та керування підключеннями. Шар даних зберігає дані, з якими працюють інші шари системи. Архітектура з шарами залишається популярним вибором для розробки як простих, так і складних розподілених систем, завдяки її гнучкості, модульності та здатності до масштабування.

Peer-to-Peer (P2P) архітектура розподілених систем відіграє ключову роль у створенні мереж, де кожен вузол (peer) функціонує як як клієнт, так і сервер для інших вузлів. Це контрастує з традиційними клієнт-серверними архітектурами, де централізований сервер надає ресурси, а клієнти їх споживають. P2P архітектури забезпечують різноманітні переваги, але також стикаються з певними викликами. 2P мережі не мають єдиної точки відмови, на відміну від централізованих систем, що робить їх більш відмовостійкими та надійними. Додавання нових вузлів до P2P мережі покращує загальну продуктивність системи, оскільки кожен новий вузол збільшує доступну пропускну спроможність та обчислювальні ресурси. P2P мережі ефективно розподіляють ресурси серед великої кількості вузлів, зменшуючи навантаження на окремі вузли та забезпечуючи більш рівномірне використання ресурсів. Вузли в P2P мережах можуть динамічно приєднуватися та від'єднуватися без значного впливу на роботу мережі в цілому. P2P мережі, такі як BitTorrent, дозволяють ефективно обмінюватися файлами між великою кількістю користувачів, зменшуючи навантаження на індивідуальні сервери. Біткойн та інші криптовалюти використовують P2P мережі для розподіленого ведення реєстру транзакцій (блокчейн), забезпечуючи високий рівень безпеки та незалежність від централізованих фінансових установ. Деякі децентралізовані соціальні мережі побудовані на P2P архітектурі для забезпечення більшої приватності та контролю користувачів над їхніми даними. Платформи на кшталт Golem, використовують P2P мережі для створення децентралізованих обчислювальних мереж, що дозволяє користувачам "орендувати" невикористані обчислювальні ресурси інших учасників. P2P архітектура продовжує розвиватися, надаючи нові можливості для створення масштабованих, відмовостійких та ефективних розподілених систем. Незважаючи на виклики, пов'язані з безпекою та управлінням, переваги, які надає P2P, роблять цей підхід привабливим для широкого спектру застосувань.

Архітектури, основані на подіях (event-driven architectures), є ключовим компонентом у проектуванні розподілених систем, забезпечуючи високу гнучкість, масштабованість та реактивність. У таких архітектурах компоненти системи взаємодіють один з одним за допомогою подій, що дозволяє їм залишатися слабо зв'язаними та асинхронно реагувати на зміни стану або зовнішні запити. Компоненти системи обробляють події асинхронно, що зменшує залежності між ними та покращує загальну продуктивність. Взаємодія між компонентами заснована на подіях, що зменшує прямі залежності та спрощує заміну або модифікацію окремих компонентів.Систему легко розширити, додавши нові обробники подій для реагування на існуючі або нові події. Асинхронна обробка подій може сприяти більшій відмовостійкості, оскільки система здатна продовжувати роботу навіть при виході з ладу окремих компонентів. Використовують таку архітектуру системи, такі як Apache Flink або Spark Streaming, що підтримують обробку потокових даних в реальному часі, використовуючи події як основні одиниці даних. Архітектури, основані на подіях, надають потужний механізм для побудови розподілених систем, які вимагають високої гнучкості, масштабованості та реактивності. Водночас, ефективне використання таких архітектур вимагає уважного планування та розуміння потенційних викликів.

Консистентне хешування є фундаментальною технікою у проектуванні розподілених систем, особливо тих, що вимагають масштабованості, високої доступності та гнучкого управління ресурсами. Вперше запропоноване як спосіб розподілу запитів між кеш-серверами у веб-мережах, консистентне хешування широко використовується в системах кешування, базах даних NoSQL, розподілених файлових системах та інших додатках. Консистентне хешування використовує кільцевий простір хешу, де кожен вузол та кожен ключ даних отримують хеш-значення, яке визначає їх позицію на

хеш-кільці. Ключі даних призначаються вузлу на кільці, що знаходиться найближче за годинниковою стрілкою. Це забезпечує розподіл даних між вузлами. При додаванні або видаленні вузлів з системи, лише невелика частина ключів потребує перерозподілу. Це значно спрощує масштабування системи, мінімізуючи вплив на доступність даних та продуктивність. Для забезпечення більш рівномірного розподілу даних вузли можуть мати кілька віртуальних хеш-позицій на кільці. Це допомагає уникнути проблеми "гарячих точок", коли один вузол обслуговує непропорційно велику частину запитів. Консистентне хешування залишається фундаментальною технікою у розробці ефективних, масштабованих та відмовостійких розподілених систем, пропонуючи унікальне рішення для викликів, пов'язаних з розподілом та управлінням даними.

Висновки

Вибір архітектурного рішення для розподілених систем є критичним етапом, що вимагає ретельного аналізу вимог до системи, цілей бізнесу та очікуваних викликів. Вибір правильної архітектури може значно покращити продуктивність, масштабованість, відмовостійкість та легкість управління системою. Проведено аналіз архітектурних рішень для розподілених систем зберігання та обробки даних, який показав, що мікросервіси ідеально підходять для комплексних застосунків, що потребують швидкого розвитку, масштабованості та легкої підтримки; сервісно-орієнтована архітектура підходить для інтеграції різноманітних бізнес-додатків та сервісів у єдину екосистему;архітектура з шарами підходить для традиційних веб-додатків із чітким розділенням логіки, даних і інтерфейсу користувача; P2P ідеально підходить для додатків, які потребують високої відмовостійкості, децентралізації та розподілених обчислень; архітектура, якак основана на подіях підходить для систем, що потребують високої реактивності та асинхронної обробки подій. Вибір архітектурного рішення для розподілених систем - це комплексний процес, який вимагає балансування між технічними, бізнесовими та операційними вимогами. Врахування майбутнього зростання, потенційних викликів і гнучкості системи є ключовими факторами для забезпечення її довгострокового успіху.

Список літератури

1. Tanenbaum, Andrew S., and Maarten van Steen. Distributed Systems: Principles and Paradigms. 3rd ed., Pearson, 2017.p.17-30.
2. Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian. SOA with REST. – Prentice Hall, 2013.
3. Kovalenko, A., Kuchuk, H., Kuchuk, N. and Kostolny, J. (2021), “Horizontal scaling method for a hyperconverged network”, *2021 Int. Conf. on Information and Digital Technologies* (IDT), Zilina, doi: https://doi.org/10.1109/IDT52577.2021.9497534
4. Panda, Dhabaleswar K.; Sayantan Sur. Network Speed Acceleration with IB and HSE. Designing Cloud and Grid Computing Systems with InfiniBand and High-Speed Ethernet. Newport Beach, CA, USA: CCGrid – 2011. p. 23.
5. HWM Singapore. An introduction to network attached storage. SPH Magazines – 2003. pp. 90–92.
6. Wang, Yandong; Goldstone, Robin; Yu, Weikuan; Wang, Teng. Characterization and Optimization of Memory-Resident MapReduce on HPC Systems. IEEE 28th International Parallel and Distributed Processing Symposium. IEEE – 2014. pp. 799–808. doi:10.1109/IPDPS.2014.87

Received (Надійшла) 25.04.2024
Accepted for publication (Прийнята до друку) 17.07.2024

Analysis of the functioning of distributed data processing and storage systems

Anton Bilokon, Stanislav Borisov, Maxim Usatenko, Volodymyr Fedorchenko

Abstract. **Relevance**. As the amount of data generated by users, IoT devices, social media, business processes, etc. grows, the need for scalable storage solutions becomes more and more evident. Distributed systems allow for efficient scaling, providing an increase in storage volumes and computing power without significant costs. Modern business requires high availability and reliability of systems, because even minimal downtime can lead to significant financial losses and a decrease in customer confidence. Distributed systems provide high availability and resilience by automatically recovering from failures and replicating data to ensure data integrity. Globalization of business requires working with data in different geographical locations. Distributed systems allow localizing data storage and pro-cessing closer to end users, reducing delays and increasing the overall performance of systems. Increasing security threats and in-creasing regulatory requirements for data protection are forcing organizations to look for more secure data storage solutions. Dis-tributed systems offer enhanced capabilities for data encryption, access control, auditing, and regulatory compliance. Processing large amounts of data often requires large computing power. Distributed data storage systems are ideal for working together with distributed computing, such as streaming data processing, machine learning, big data, allowing to efficiently distribute tasks and process large amounts of information. Among the challenges that distributed data storage systems may face are - ensuring data con-sistency between nodes, network delay management, data protection and security. Various strategies and technologies are used to address these challenges, including consistent hashing algorithms, data replication, transaction protocols with guaranteed atomicity, consistency, isolation, and durability, and sequential consistency models. Thus, in conditions of constant growth of data volumes and increasing requirements for their processing, distributed data storage systems are a key element of the infrastructure of any or-ganization striving for innovation and efficiency. **The purpose** of this work is to analyze the functioning of distributed data processing and storage systems. **The object** of research is distributed data processing and storage systems. **The subject** of research is architectural solutions of distributed data storage and processing systems. **The results**. The analysis of the functioning of distributed data processing and storage systems was carried out. The choice of an architectural solution for a distributed system depends on the specifics of the tasks to be solved, requirements for performance, scalability, reliability and availability. Usually, effective distributed systems use a combination of these approaches to achieve optimal results. Choosing an architectural solution for distributed systems is a complex process that requires balancing technical, business, and operational requirements. Consideration of future growth, potential challenges and flexibility of the system are key factors to ensure its long-term success.

Keywords: distributed information system, data processing, Web-Scale, data storage, data processing center, event-driven architectures.