УДК 004.67
JEL C80
DOI: 10.26906/EiR.2023.4(91).3218

СУЧАСНІ ПІДХОДИ ДО ЗБЕРІГАННЯ ДАНИХ НА РОЗПОДІЛЕНІЙ ІНФРАСТРУКТУРІ

Приймак Василь Іванович*, доктор економічних наук, професор,
завідувач кафедри інформаційних систем у менеджменті
Рибак Юрій Михайлович, магістрант
Голубник Ольга Романівна, кандидат економічних наук,**
доцент кафедри інформаційних систем у менеджменті
Львівський національний університет імені Івана Франка

**ORCID 0000-0003-0244-8661*
***ORCID 0000-0003-1211-4614*

Стаття отримана редакцією 01.12.2023 р.
The article was received by editorial board on 01.12.2023

Вступ. Сьогодні для значної кількості населення нашої планети звичними стали терміни «Інтернет», «соціальні мережі», «інформаційні системи» та інші. Людство у своєму технологічному розвитку трансформувалось з індустріального в інформаційне суспільство, для якого характерним є зростання ролі інформації в суспільному житті, виробництві товарів та наданні послуг населенню. Різке збільшенні необхідних для оброблення обсягів інформації спонукало відповідних фахівців до пошуку нових методів та алгоритмів опрацювання і зберігання даних. Ефективними виявились підходи розосередженого розміщення необхідної для оброблення користувацької інформації. Стрімкий розвиток інформаційних технологій веде до необхідності розуміння найновіших підходів до зберігання та обробки даних, що вказує на актуальність виконаних в даній роботі наукових досліджень.

Огляд останніх джерел досліджень і публікацій. Висвітленням сучасних підходів до оптимального зберігання великих обсягів даних розглядали у своїх роботах як українські, так і зарубіжні науковці і практики. Зокрема, в науковій статті [1] для швидкого пошуку та аналізу запитів запропоновано використовувати розподілені бази даних, в яких інформація розподіляється і зберігається на декількох пристроях. Для взаємозв'язку всіх даних та швидкого пошуку пропонується застосувати метод колонкових індексів. Теоретичним і практичним аспектам роботи з розподіленими базами даних присвячена книга Мартіна Клеппмана [2]. В книзі цього автора розглянуто ключові принципи, алгоритми і компроміси, без яких не обійтись при розробленні високонавантажених систем для роботи з даними. У книзі Алекса Петрова [3] показано як організувати кластер із кількох обчислювальних вузлів. Автор зупинився на важливості розуміння теоретичних концепцій для побудови відмовостійких розподілених систем, як розподілені системи відрізняється від одновузлових і, які проблеми, обмеження і складнощі виникають в розподіленому середовищі. У книзі автора Алекса Гореліка [4] розглянуто перероодові підходи до впровадження розподілених систем як бізнес рішень, а саме автор описує процес впровадження озера даних (Data Lake). Також автор досліджує плюси та мінуси різних варіантів побудови Data Lake – на хмарних платформах, на внутрішніх серверах компаній та у віртуальному середовищі.

Однак, в літературі відсутні огляди джерел про сучасні підходи до зберігання даних на розподіленій інфраструктурі і публікації з порівняльним аналізом на цю тематику.

Метою даної статті є дослідження проблем оптимального зберігання великих даних і розгляд найновіших підходів до вирішення цих проблем для забезпечення необхідного рівня продуктивності обчислювальних систем.

Основний матеріал і результати. Сучасний світ технологій вимагає від компаній швидкої та точної аналітики, яка б допомагала приймати обґрунтовані рішення. Однак збільшення обсягів даних та різноманітність їх типів створює виклик для зберігання та обробки інформації, на основі яких можуть прийматися рішення. Саме тут виникають питання щодо вибору оптимальної системи зберігання та обробки даних.

Останніми роками компанії для прийняття рішень, які керуються даними виділяють все більше ресурсів для впровадження Big Data технологій (технологій роботи з великими обсягами даних). Практика показує неефективність обробки таких даних, які містяться у відповідній базі даних (БД) чи файловій системі з використанням єдиного обчислювального пристрою будь-якої потужності. Більш раціонально розподіляти інформацію на багатьох вузлах з можливості її подальшої паралельної обробки. Технічно при цьому використовувати не один фізичний пристрій, а декілька, які територіально можуть бути розміщені в різних місцях. У результаті, не тільки збільшується продуктивність і швидкість роботи системи за рахунок паралелізму, а й отримується можливість одночасної роботи з інформацією декільком користувачам.

Для паралельного опрацювання інформації її розподіляють на менші частини, які можна незалежно обчислювати на різних пристроях. Це істотно пришвидшує процес оброблення даних, дає можливість організувати роботу із запитами гнучко та ефективно. Такий підхід дає змогу не тільки пришвидшити оброблення запитів, а й забезпечити можливість одночасного доступу до інформації і підвищити надійність її зберігання [1].

На даний час існує два принципових підходи до забезпечення необхідного рівня продуктивності систем:

– вертикально масштабований підхід;

– горизонтально масштабований підхід.

Якщо все, що потрібно, це збільшити навантаження, найпростішим підходом є придбання потужнішої машини – це називається вертикальним масштабуванням. Безліч процесорів, оперативної пам'яті та безліч дисків можуть бути об'єднані під однією операційною системою, швидке з'єднання дозволяє будь-якому центральному процесору (CPU) отримати доступ до будь-якої частини пам'яті або диска. У цьому типі архітектури, або підходу зі спільною пам'яттю (shared-memory architecture), усі компоненти можна розглядати як єдине ціле [2, с. 146].

Однак, така архітектура не позбавлена певних недоліків. Один з них стосується вартості системи, зростання якої відбувається швидшими темпами як зростання обсягів її технічних пристроїв. Вартість машини з подвійним збільшення CPU, оперативної пам'яті і ємності диску зросте значно більше, ніж удвічі. Інший недолік стосується потужності машини. Вдвічі більшій за розміром системі через певні вузькі місця може бути не під силу подвійне навантаження.

Окрім вище перелічених недоліків, shared-memory architecture підхід, також, очевидно, має низький рівень відмовостійкості, оскільки у разі відмови одного компоненту вся система може вийти з ладу. Ще одна важлива характеристика, яку не можуть задовільнити такі системи, це забезпечення географічної близькості клієнта до даних. Тобто, такий архітектурний підхід передбачає знаходження сервера тільки в одній географічній локації, і для сервісів які працюють по всьому світу неможливо забезпечити близькість клієнта до сервера.

На противагу, shared-memory architecture існує архітектура без розділення ресурсів (shared-nothing architecture) – цей підхід називають горизонтальним масштабуванням. При такому підході кожна машина або віртуальна машина, на якій працює програмне забезпечення бази даних, називається вузлом. Кожен вузол використовує свої CPU, пам'ять та диски окремо. Будь-яка координація між вузлами здійснюється на програмному рівні, використовуючи звичайну мережу [2, с. 146].

Для shared-nothing architecture не потрібне спеціальне обладнання, тому можна використовувати будь-які комп'ютери, які були б для нас найкращими за співвідношенням їхньої вартості до продуктивності. Потенційно можна поширювати дані у територіальному співвідношенні для того, щоб зменшити користувачам час доступу до них і при необхідності мати можливість пережити втрату цілого центру обробки даних.

Слід зауважити, що крім вказаних виділяють ще третій архітектурний підхід із загальними дисками (Shared Disk Architecture). В його основі лежить використання кількох машин з незалежними центральними процесорами та оперативною пам'яттю. Самі дані в цьому випадку зберігаються на спільно використовуваних через швидкісну мережу масиві дисків. Цей підхід використовується для деяких завдань зберігання в сховищах даних (data warehousing), але конфлікти та деякі інші причини, зокрема, накладні витрати на блокування обмежують масштабованість архітектури зі спільним диском.

До розподіленої архітектури чи розподілу потрібної інформації на декількох машинах існують певні вимоги. До основних з них можна віднести [2, с. 146]:

– масштабованість, про яку ми говорили вище. Якщо обсяг даних, навантаження на читання чи записування переростають можливості однієї машини, то можна розподілити це навантаження на декілька комп'ютерів. Важливо щоб система могла як збільшувати так із зменшувати кількість вузлів;

– відмовостійкість / висока доступність. Якщо додаток повинен продовжувати працювати навіть у випадку відмови однієї чи декількох машин, мережі чи навіть всього центру обробки даних (ЦОД), то можна використовувати надлишкові комп'ютери. При відмові одного з них виконання задач делегується іншому;

– затримка. За наявності користувачів по всьому світу необхідні сервери у різних точках земної кулі, щоб кожен користувач обслуговувався вузлом, географічно розташованим максимально близько від нього. При цьому користувачам не потрібно буде чекати, доки мережні пакети обійдуть половину земної кулі.

Якщо говорити про архітектуру без розділення ресурсів, то при цьому підході до зберігання даних на декількох вузлах можна скористатись одним з двох найпоширеніших способів: реплікацією чи партиціонуванням (хоча використовують і інші терміни). Використання певного з цих способів для зберігання даних залежить від обсягу цих даних.

Якщо набір даних досить малий і його можна розмістити на одному комп'ютері, то використовують реплікацію. Реплікація це спосіб досягнення розподіленої архітектури в якому ці самі дані зберігають не в одному, а в кількох вузлах. На рис. 1 показано як концептуально виглядає реплікація.

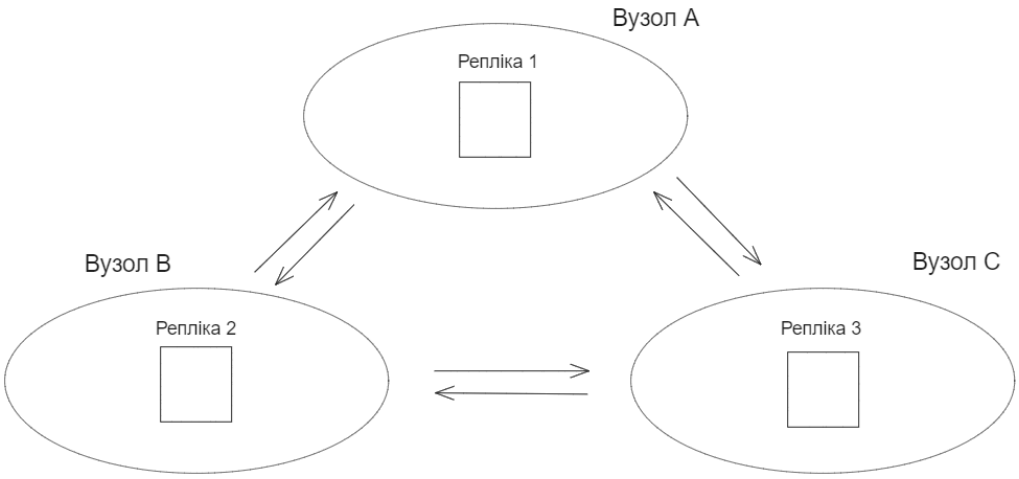

Рис. 1. Реплікація даних

Джерело: сформовано авторами

Причому оптимально репліки розміщувати в територіально віддалених вузлах для швидшого доступу до них користувачів. Геореплікація в цьому випадку служить багатьом цілям: вона підвищує доступність і здатність протистояти збоям одного або кількох вузлів даних шляхом підтримки реплік. Це також може допомогти зменшити затримку, розмістивши копію даних фізично ближче до клієнта [3, с. 340].

Хоча, на перший погляд, зберігати ті самі дані у багатьох вузлах досить просто, така архітектура має свої недоліки. Крім надлишковості даних, недоліками реплікації є затримка в отриманні користувачем потрібної інформації у разі можливих збоїв, недоступності до потрібного вузла, розривів мережі та інших ситуацій. Коли записи даних модифіковані, кожна репліка має отримати відповідні зміни. Говорячи про реплікацію, найбільш важливими є три події: запис, оновлення репліки та читання. У деяких випадках оновлення реплік може відбутись після завершення запису з точки зору клієнта, але це все ще не змінює того факту, що клієнт повинен мати можливість зберігати порядковість операцій [3, с. 342].

Оновлення інформації при використанні розглянутої архітектури зберігання даних залежить від того, який з підходів до реплікації використовується. При цьому реплікація може бути з ведучим вузлом чи без нього. Якщо використовується тільки один такий вузол, то користувацькі дані від всіх клієнтів потрапляють у цей вузол, а потім розповсюджуються по всіх ведених вузлах (їх називають репліками).

У той час як читання інформації можливе з будь-якої репліки. Особливість цієї інформації полягає у тому, що вона може бути застарілою, оскільки оновлена інформація з ведучого вузла ще не встигла дійти до веденого вузла. У випадку ж кількох ведучих вузлів потрібно мати лідера в кожному датацентрі. Користувач зв'язується тільки з одним із них та відправляє йому інформацію про здійснювані операції. Корегування інформації в інших вузлах відбувається цим ведучим вузлом, якому проінформовано про зміни і який відправляє скореговані відомості іншим ведучим і всім веденим вузлам.

Щодо третього з можливих способів реплікації коли відсутні ведучі вузли, то користувачі відправляють необхідну інформацію до кількох вузлів. Запис вважається успішним якщо встановлена кількість реплік із усіх наявних отримала інформацію. Аналогічно із вичитуванням, клієнт читає відповідну інформацію з декількох вузлів. Отримана інформація порівнюється і тим самим виявляється найсвіжіша та застаріла інформація. У разі наявності такої інформації вносяться зміни і клієнту повертається актуальні дані.

Кожний з цих способів реплікації має свої особливості, переваги і недоліки. Оскільки перший з них, коли існує тільки один ведучий вузол, досить простий і при ньому майже відсутні конфліктні ситуації, то він досить широко розповсюджений на практиці. Однак, у разі відмови цього головного вузла (ведучим вузлом в цьому випадку стане якийсь з ведених) можлива втрата інформації. Два інші способи реплікації за рахунок їхньої складності і досить низького рівня гарантій узгодженості дещо більш стійкі до надзвичайних ситуацій з вузлами і мережею в цілому, а також до ймовірної часової затримки з виконання замовлення.

Якщо для розміщення даних недостатньо однієї машини, то для виконання цієї процедури використовується підхід, який називається партиціонуванням (шардингом). Головна мета партиціонування це досягнення масштабованості [2, с. 199]. Дані можуть бути поділені на безліч партицій які можуть бути розміщені на різних вузлах. Це дозволяє розподілити навантаження на систему по всіх цих вузлах. Щоб виконати партиціонування наборів даних недостатньо їх поділити на певні частини. Цей поділ повинен бути виконаний так, щоб на всі вузли навантаження по даних і запитах було приблизно однакове.

У сучасних базах даних партиціонування зазвичай поєднується із реплікацією, тобто кожна партиція має певну кількість копій які зберігаються на різних вузлах. На рис. 2 показаний приклад побудови кластера із чотирьох вузлів. Дані розділені на партиції A, B, C, D. На кожному вузлі зберігається 3 різні партиції, таким чином достягається коефіцієнт реплікації 3, тобто кожна партиція має 3 копії.

Рис. 2. Партиціонування із реплікацією

Джерело: сформовано авторами

При чому, в такому підході також застосовується принцип із лідером та підпорядкованими партиціями. Тобто одна з копій обирається ведучою, а інші веденими. Для досягнення високого рівня відмовостійкості важливо щоб ведуча і ведені партиції зберігалися на різних вузлах.

Одним із ключових аспектів якісного партиціонування є забезпечення рівномірності розподілу. Оскільки, при паралельній обробці даних нерівномірний розподіл партицій може призвести до піково-

го навантаження на одні вузли, в той час як інші будуть простоювати. Найчастіше використовуються два підходи до партиціонування: на основі ключа або на основі хешу. Перша техніка включає поділ даних на партиції на основі конкретного діапазону значень ключів у наборі даних. Кожна партиція містить дані, які належать до попередньо визначеного діапазону, що полегшує розподіл та обробку даних в розподіленому обчислювальному середовищі. Типовим прикладом такого підходу є партиціонування на основі дати. Наприклад, набір даних містить колонки "рік", "місяць", "день", тоді можна виконати партиціонування на основі цих колонок, де кожна окрема дата буде вважатися окремою партицією, тобто дані за один день лежать в одній партиції. Цей підхід є ефективним, оскільки дані є відсортовані, що пришвидшує доступ до них, але очевидним мінусом даного підходу є відсутність гарантії рівномірності розподілу, оскільки, як в нашому прикладі, в різні дні може накопичуватися різний об'єм даних. Інший підхід до партиціонування передбачає поділ даних на частини на основі хеш-значення певного поля даних, як правило, одного або кількох ключів. Цей підхід порушує порядковість ключів, але розподіл, як правило, більш рівномірний.

У процесі функціонування певного кластера машин можуть виникати проблеми у випадку введення в нього нових вузлів чи видалення деяких працюючих, з якими стався збій у роботі. Тоді треба виконати перерозподіл даних у партиціях, щоб уникнути перенавантаження в деяких вузлах (гарячих точках).

У роботі із розподіленими системами, попри ряд їхніх переваг, існують певні виклики, подолання яких вимагає застосування складних архітектурних рішень. В один момент часу дані на різних вузлах можуть відрізнятися, це очікувано, оскільки запис в базу можуть приймати різні вузли в різний час. Хоча, слід зауважити, що більшість реплікованих баз даних забезпечують кінцеву узгодженість (eventual consistency). Це означає, що якщо ви припиняєте запис до бази даних і чекаєте деякий невизначений проміжок часу зрештою всі запити на читання повертатимуть однакове значення [2, с. 322]. Тобто, вказана неузгодженість є тимчасовою, але суттєвою проблемою є затримка в часі, оскільки невідомий час коли неузгодженість буде усунута. Хоча і кінцева узгодженість не є універсальним рішенням. Для прикладу, якщо виникають проблеми в мережі і деякі репліки від'єдналися, тобто вони не можуть обробляти запити. У такому випадку система може очікувати на відновлення невизначений час мережі або ж бути недоступною взагалі. Якщо ж прийняти, що система не вимагає узгодженості даних, то кожна репліка може обробляти запити незалежно, але тоді немає гарантії що дані на різних репліках будуть однаковими в конкретний момент.

Щоб описати технічні вимоги до БД зазвичай використовують певні характеристики у залежності від типу цих даних. Для характеристики реляційних СУБД зазвичай застосовують принципи ACID (від англ. atomicity, consistency, isolation, durability), відповідно це – атомарність або неподільність, консистентність або цілісність, ізольованість транзакцій та довговічність чи стійкість. Ці принципи визначають модель надійних транзакцій в середовищі з багатьма користувачами. Для характеристики розподілених систем прийнято опиратися на принципи, описані в CAP (від англ. consistency, availability, partition tolerance) теоремі, відповідно це – консистентність, доступність, стійкість до розподілення. Теорема говорить про те що, ми можемо мати щонайбільше дві з цих властивостей для будь-якої системи розподілених даних [5].

Для кращого розуміння вище згаданих властивостей деталізуємо їх. Консистентність або узгодженість означає, що всі вузли в розподіленій системі бачать ті самі дані одночасно. Іншими словами, коли клієнт читає з одного вузла, він повинен отримати найновіше значення. Досягнення узгодженості бажано в системах, де цілісність і коректність даних є критичними, наприклад у фінансових програмах. Однак забезпечення узгодженості часто відбувається за рахунок збільшення затримки та зниження доступності.

Доступність передбачає, що розподілена система повинна завжди відповідати на запити клієнта, навіть за наявності збоїв. Системи із високою доступністю призначені для забезпечення безперебійного обслуговування, гарантуючи, що клієнти можуть отримати доступ до системи та виконувати операції в будь-який час. Досягнення доступності вимагає більшої надлишковості даних, механізмів відмовостійкості та здатності чітко справлятися з відмовами. Однак, досягнення високої доступності може призвести до потенційної неконсистентності у системі.

Стійкість до розподілення – це здатність системи продовжувати функціонувати та забезпечувати доступність навіть у разі виникнення мережевих збоїв. Тобто кожна з частин системи, якщо вона доступна, повинна мати можливість працювати автономно, віддаючи коректну відповідь та повертаючи свої дані.

Теорема CAP дає цінну інформацію про те на які компроміси треба йти при проектуванні розподілених систем. Хоча вона стверджує, що неможливо досягти консистентності, високої доступності та стійкості до розподілення одночасно, розробники мають приймати обґрунтовані рішення на основі конкретних потреб своїх програм. Розуміння теореми CAP і її наслідків, допомагає орієнтуватися в складнощах розподілених систем і досягти відповідного балансу між узгодженістю, доступністю та відмовостійкістю.

Висновки. Виконані дослідження показали, що з розвитком інформаційних технологій та збільшенням обсягів використовуваних даних виникає питання оптимального зберігання цих даних для забезпечення необхідного рівня продуктивності обчислювальних систем. На сьогодні існує два принципових підходи до забезпечення необхідного рівня продуктивності систем: вертикально та горизонтально масштабовані підходи. Найбільш придатним для вирішення задач є другий з цих підходів чи, по іншому, тип архітектури без розділення ресурсів. При цьому підході до зберігання даних на декількох вузлах залежно від обсягу цих даних використовують два найпоширеніші способи: реплікацію (у випадку малого набору даних і можливості його розміщення на одному комп'ютері) або партиціонування (шардинг), який включає поділ даних на партиції на основі конкретного діапазону значень ключів у наборі даних. У разі проектування розподілених систем потрібно старатись досягти відповідного балансу між такими принципами як узгодженість, доступність та відмовостійкість, всі з яких згідно теореми CAP одночасно досягти неможливо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Климаш М., Гордійчук-Бублівська О., Чайковський І., Данильченко Т. Дослідження алгоритмів паралельного опрацювання інформації в базах даних. *Інфокомунікаційні технології та електронна інженерія*. 2021. № 1 (1). С. 51–62.

2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. 2017. 590 p.

3. Petrov A. Database Internals: A Deep Dive into How Distributed Data Systems Work 1st Edition. 2019. 590 p.

4. Gorelik A. The Enterprise Big Data Lake. 2019. 197 p.

5. Brewer E. Towards Robust Distributed System. Symposium on Principles of Distributed Computing, *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*. July 16-19, 2000. Portland, 2000.

REFERENCES:

1. Klymash M., Hordiichuk-Bublivska O., Chaikovskyi I. and Danylchenko T. (2021) Research of algorithms for parallel processing of information in databases, *Infocommunication technologies and electronic engineering*, no. 1 (1), pp. 51–62.

2. Kleppmann M. (2017) *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. 590 p.

3. Petrov A. (2019) *Database Internals: A Deep Dive into How Distributed Data Systems Work.* 1st Edition. 590 p.

4. Gorelik A. (2019) *The Enterprise Big Data Lake*. 197 p.

5. Brewer E. (July 16-19, 2000) Towards Robust Distributed System. Symposium on Principles of Distributed Computing, *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing.* Portland.

УДК 004.67
JEL C80

Приймак Василь Іванович, доктор економічних наук, професор, завідувач кафедри інформаційних систем у менеджменті. **Рибак Юрій Михайлович,** магістрант. **Голубник Ольга Романівна,** кандидат економічних наук, доцент кафедри інформаційних систем у менеджменті, Львівський національний університет імені Івана Франка. **Сучасні підходи до зберігання даних на розподіленій інфраструктурі.**

Метою даної статті є дослідження проблем оптимального зберігання великих даних і розгляд найновіших підходів до вирішення цих проблем для забезпечення необхідного рівня продуктивності обчислювальних систем. Під час проведення наукового дослідження використовувалися методи аналізу та узагальнення, синтез та системний підхід для вивчення сучасних способів зберігання даних на розподіленій інфраструктурі, які застосовуються в процесі оброблення інформації. В роботі розглянуто ключові аспекти, які виникають із розвитком технологій та значним збільшенням обсягів використовуваних даних. Встановлено, що на сьогодні існує два принципових підходи до забезпечення необхідного рівня продуктивності систем: вертикально (підходу зі спільною пам'яттю (shared-memory architecture)) та горизонтально (архітектура без розділення ресурсів (shared-nothing architectures)) масштабовані підходи, кожен з яких має як позитивні, та і негативні сторони. Згідно останніх розробок та досліджень у сфері роботи із великими даними, найбільш придатною

для вирішення задач є архітектура без розділення ресурсів, яка детально проаналізована в роботі, а саме, висвітлено які основні концептуальні та архітектурні рішення розподілених систем та вказано вимоги до цієї архітектури. При такій архітектурі кожен вузол (машина або віртуальна машина, на якій працює програмне забезпечення бази даних) використовує свої CPU, пам'ять та диски окремо і будь-яка координація між вузлами здійснюється на програмному рівні використовуючи звичайну мережу. При цьому підході до зберігання даних на декількох вузлах залежно від обсягу цих даних використовують два найпоширеніші способи: реплікацію (у випадку малого набору даних і можливості його розміщення на одному комп'ютері) або партиціонування (шардинг), який включає поділ даних на партиції на основі конкретного діапазону значень ключів у наборі даних. Вказано на необхідність досягнення при проектуванні розподілених систем відповідного балансу між такими принципами як узгодженість, доступність та стійкість до розподілення. Обґрунтовано твердження про можливість мати щонайбільше дві з цих властивостей для будь-якої системи розподілених даних.

Ключові слова: зберігання великих даних, архітектура без розділення ресурсів, архітектура, вузол, партиціонування, партиції, розподілені системи.

UDC 004.67
JEL C80

Vasyl Pryimak, Doctor of Economic Sciences, Head of the Information Systems in Management Department. **Yurii Rybak,** Master's Student. **Olha Holubnyk,** Ph.D. in Economics, Associate Professor of Information Systems in Management Department, Ivan Franko National University of Lviv. **Modern approaches to data storage on a distributed infrastructure.**

The purpose of this article is to study the problems of optimal big data storage and consider the latest approaches to solving these problems to ensure the required level of performance of computing systems. During the scientific research, methods of analysis and generalization, synthesis and system approach were used to study modern methods of data storage on a distributed infrastructure, which are used in the process of information processing. The paper considers the key aspects that arise with the development of technology and a significant increase in the amount of data used. It is established that today there are two fundamental approaches to ensuring the required level of system performance: vertically (approach with shared-memory architecture) and horizontally (shared-nothing architectures) scalable approaches, each of which has both positive and negative sides. According to the latest developments and research in the field of working with big data, the most suitable for solving problems is an architecture without separation of resources, which is analyzed in detail in the work, namely, what are the main conceptual and architectural solutions of distributed systems and the requirements for this architecture are indicated. With this architecture, each node (the machine or virtual machine running the database software) uses its own CPU, memory and disks separately and any coordination between nodes is carried out at the software level using a conventional network. In this approach to data storage on several nodes, depending on the volume of this data, two most common methods are used: replication (in the case of a small data set and the possibility of placing it on one computer) or partitioning (sharding), which includes dividing data into parties based on a specific range of key values in the data set. The need to achieve an appropriate balance between such principles as consistency, accessibility and resistance to distribution in the design of distributed systems is indicated. The statement about the possibility of having at most two of these properties for any distributed data system is substantiated.

Key words: big data storage, architecture without division of resources, architecture, node, partitioning, partitions, distributed systems.