УДК 004.453.4

Р.П. Струбицький, Н.Б. Шаховська
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

АНАЛІЗ ІНФРАСТРУКТУРИ ТА МОДЕЛЕЙ ОРГАНІЗАЦІЇ ХМАРКОВИХ СХОВИЩ ДАНИХ

Проаналізовано архітектурні підходи до побудови хмаркових сховищ даних. Розглянуто як зосереджені, так і розподілені сховища, проаналізовано ефективність їх використання.

Ключові слова: STaaS, хмаркові сховища даних, організація сховищ даних.

The paper analyzes the architectural approaches for building cloud data warehouses. The different approaches, including concentrated and distributed storages, were examined and effectiveness of their use was analyzed.

Key words: STaaS, cloud computing, cloud data warehousing.

Вступ. Постановка проблеми

Хмарні обчислення змінюють звичайні уявлення про обчислення. Ця технологія дає змогу обчисленням набути статусу сервісу інформаційних комунікацій з можливістю використання на вимогу у широкому спектрі пристроїв, у будь-якому місці за допомогою мережі Інтернет.

Ця нова бізнес-модель дає організаціям змогу збільшувати продуктивність надання послуг за нижчою ціною і з більшою легкістю. Світові дослідження показують, що використання хмаркових обчислень сприяє економічному зростанню підприємств.

Ще кілька років тому в комп'ютерній галузі панувала гонитва за зростанням тактових частот процесорів та зростанням об'ємів пам'яті накопичувальних пристроїв. Програми, які приходили на зміну старим, вимагали чималих обчислювальних потужностей, втім, як і нові операційні системи. Але останнім часом, всупереч очікуванням, все пішло іншим шляхом. Із впровадженням технології Web 2.0, орієнтованої на соціальну складову інформаційного середовища, почала бурхливо розвиватися "філософія" програм за запитом. Ідея програм за запитом полягає в тому, що користувач, який має засоби доступу до мережі Інтернет, може скористатися сервісом залежно від своїх потреб. Дані користувача зберігаються на сервері, який приєднаний до мережі Інтернет, місцерозташування якого не відоме кінцевому користувачу, а користуватися ними можна за допомогою різних пристроїв. "Хмарка", простіше кажучи, це віддалений дата–центр, який надає послуги для кінцевого користувача або для бізнесу, що оплачуються за фактом використання сервісу.

У статті аналізується інфраструктура сховищ даних, організованих у хмару.

Аналіз останніх досліджень

Хмаркова система зберігання даних, або зберігання даних як послуга – це абстрактне поняття, яке відповідає системі зберігання даних, яку можна адмініструвати за вимогою через спеціальний інтерфейс. Цей інтерфейс абстрагує і місцезнаходження системи, так що локальна вона, чи віддалена, чи гібридна – не має суттєвого значення. Хмаркові інфраструктури зберігання даних утворюють нові архітектури, які підтримують різні рівні обслуговування поверх потенційно великої групи користувачів і географічно розподілених накопичувачів.

За тих темпів, з якими сьогодні зростають обсяги даних, не дивно, що зростає й популярність хмаркових систем зберігання. Швидше за все зростають обсяги архівних даних, які ідеально підходять для зберігання в хмарці при дотриманні ряду умов, зокрема умов економічності, частоти звертань, захисту і доступності. Але не всі хмаркові системи зберігання однакові. Один постачальник може зосередитись на вартості, тоді як інший фокусується на доступності або

продуктивності. Жодна архітектура не орієнтована на щось одне, і міра реалізації архітектурою заданої характеристики визначає її ринок і цільові моделі використання.

На перший погляд, хмаркові системи зберігання даних мають низку переваг порівняно з традиційними системами зберігання даних. Наприклад, якщо зберігають дані в хмаркових системах зберігання, то отримують доступ до цих даних з будь-якого місця, в якому є доступ до Інтернету. Не потрібно носити з собою фізичний пристрій зберігання або використовувати той самий комп'ютер, щоб зберегти і відновити інформацію. При правильній організації системи зберігання можна навіть дозволити іншим отримати доступ до даних.

На самому базовому рівні хмарковим системам зберігання даних потрібен всього один сервер даних, під'єднаний до Інтернету. Клієнт (користувач послуги хмаркового зберігання) надсилає копії файлів через Інтернет до сервера даних, який потім записує інформацію. Коли клієнт хоче отримати інформацію, він отримує доступ до даних сервера через веб-інтерфейс. Потім сервер або відправляє файли назад клієнту, або дозволяє клієнту доступ і управління файлами на сервері.

Аналіз архітектури вимагає вираховування її робочих параметрів. Під ними розуміють різні характеристики архітектури, зокрема вартість, продуктивність, можливість віддаленого доступу тощо. На жаль, у науковій літературі відсутній детальний аналіз використовуваних інфраструктур та моделей зберігання даних. Основні матеріали, в яких висвітлено сервіс хмаркового зберігання даних, орієнтовані на кінцевих користувачів і описують існуючі системи, їхні переваги та недоліки з погляду користувача. Для проектування нових сховищ і удосконалення існуючих доцільно проаналізувати інфраструктури таких сховищ, моделі зберігання тощо.

Виклад основного матеріалу

Хмаркова архітектура зберігання даних – це насамперед доставка ресурсів зберігання даних на вимогу в високомасштабованому і мультитенантному середовищі (можливість ізольовано обслуговувати користувачів з різних організацій у межах одного сервісу). Узагальнено хмаркова архітектура зберігання даних являє собою зовнішній інтерфейс, який надає API для доступу до накопичувачів (рис. 1).

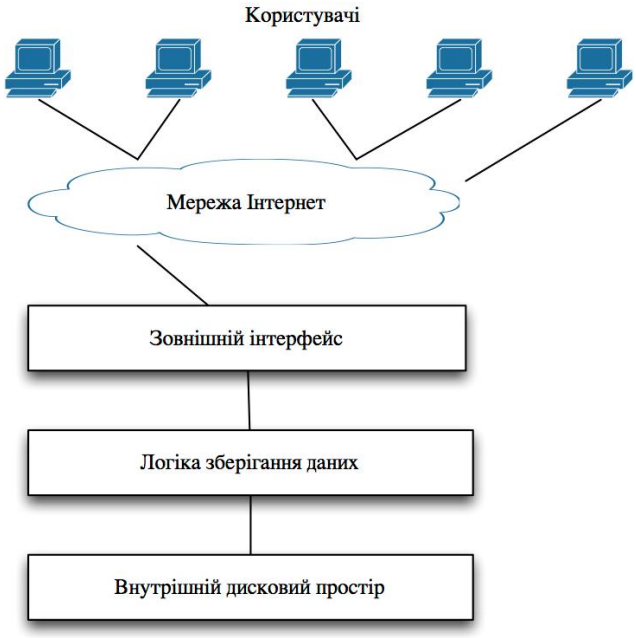

Рис. 1. Хмаркова архітектура зберігання даних, побудовано за даними [2, 4, 5]

У традиційних системах зберігання даних це протокол SCSI [1], але в хмаркових сховищах з'явилися нові протоколи. Серед них можна знайти зовнішні протоколи Web–сервісів, файлові протоколи і навіть традиційніші зовнішні інтерфейси (Internet SCSI, iSCSI та ін.) [2, 3]. За зовнішнім інтерфейсом розташовується рівень проміжного програмного забезпечення – логіка

зберігання даних. Цей рівень реалізує такі функції, як реплікація даних і скорочення обсягу даних, за традиційними алгоритмами розміщення даних з урахуванням географічного розташування. Нарешті, внутрішній інтерфейс організовує фізичне зберігання даних. Це може бути внутрішній протокол, який реалізує специфічні функції, або традиційний сервер з фізичними дисками.

Загальні характеристики хмаркової архітектури

Представлена загальна архітектура (див. рис. 1) дає змогу виділити деякі характеристики сучасної хмаркової архітектури зберігання даних. Ці характеристики не належать винятково до певного рівня, а можуть відповідати декільком (табл. 1).

Таблиця 1

Характеристики хмаркової архітектури зберігання даних, побудовано за даними [2, 4, 5]

Характеристика	Опис
Керованість	Здатність керувати системою за наявності мінімальних ресурсів
Метод доступу	Протокол, через який надаються послуги хмаркового зберігання даних
Продуктивність	Вимірюється пропускною здатністю і часом затримки
Багатокористувацькість	Підтримка багатьох користувачів (орендаторів)
Масштабованість	Можливість поступового нарощування для задоволення нових вимог або обробки підвищеного навантаження
Готовність даних	Вимірюється часом безвідмовної роботи системи
Управління	Можливість управляти системою – зокрема, вибираючи вартість, продуктивність або інші характеристики
Ефективності зберігання	Міра ефективності використання накопичувачів
Вартість	Міра вартості зберігання даних (часто в доларах за гігабайт)

Розглянемо детальніше кожну з цих характеристик з аналізом її впливу на архітектуру сховища даних.

Керованість. Однією з ключових характеристик хмаркової системи зберігання даних є її вартість. Якщо клієнт зможе купувати і керувати ресурсами зберігання локально, а не орендувати їх у хмарці, ринок хмаркових систем зберігання зникне. Його витрати можна поділити на дві загальні категорії: вартість самої фізичної екосистеми зберігання і витрати на керування нею. Вартість управління прихована, але є довгостроковим компонентом загальної вартості. Через це хмаркова система зберігання повинна бути значною мірою самокерованою. Вирішальне значення має можливість додавати нові накопичувачі, коли система автоматично змінює конфігурацію для їх розміщення, і здатність системи знаходити і автоматично виправляти помилки. У майбутньому такі концепції, як автономні обчислення, відіграватимуть ключову роль у хмаркових архітектурах зберігання [2].

Метод доступу. Однією з найяскравіших відмінностей між хмарковою та традиційною системами зберігання є засоби доступу до них (рис. 2). Більшість постачальників пропонує різні методи доступу, однак загальноприйнятими є API Web-сервіс. Більшість з них реалізовані на принципах REST [6], що основані на об’єктно-орієнтованій схемі, розробленій поверх протоколу HTTP.

Методи доступу за REST–API без запам’ятовування стану прості та ефективні. Ці методи реалізують багато постачальників хмаркових послуг зберігання, включаючи Amazon Simple Storage Service (Amazon S3) [7], Windows Azure [8] і Mezeo Cloud Storage Platform [9, 10].

Однак проблема API Web–сервісів полягає в тому, що для того, щоб скористатися перевагами хмаркової системи зберігання, вони вимагають інтеграції з додатком. Тому разом із хмарковими системами зберігання для забезпечення безпосередньої інтеграції використовуються також загальні методи доступу. Наприклад, протоколи на основі файлів, такі як NFS/Common Internet File System (CIFS) [11] або FTP, або протоколи на основі блоків, такі як iSCSI. Такі методи доступу надають Nirvanix, Zetta, Cleversafe та інші постачальники послуг хмарного зберігання.

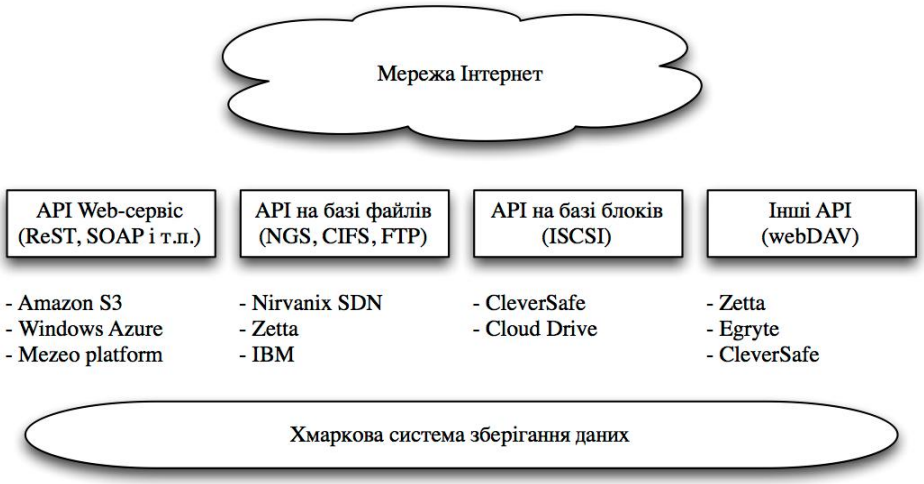

Рис. 2. Методи доступу до хмаркових систем зберігання даних, побудовано за даними [2]

Вищезазначені протоколи найпоширеніші, але для хмаркового зберігання підходять й інші. Один з найцікавіших – Web–based Distributed Authoring and Versioning (WebDAV) [12]. WebDAV також базується на HTTP і дозволяє використовувати Web як ресурс для читання і запису. До постачальників, що використовують WebDAV, входять Zetta, Cleversafe та інші.

Можна знайти і такі рішення, які підтримують кілька протоколів доступу. Наприклад, IBM Smart Business Storage Cloud дає змогу використовувати протоколи на основі файлів (NFS і CIFS) і протоколи на основі SAN в одній і тій самій інфраструктурі віртуалізації систем зберігання даних.

Продуктивність. Продуктивність можна аналізувати з багатьох аспектів, але основне завдання хмаркової системи зберігання даних – це переміщення даних між користувачем і віддаленим постачальником хмаркових послуг. Проте існує проблема в транспортному протоколі TCP, який є головним робочим протоколом Інтернету. TCP керує потоком даних на основі підтвердження прийому пакетів з віддаленого вузла. Втрата або затримка пакетів дозволяють керувати перевантаженням, що ще більше обмежує продуктивність для уникнення глобальних мережевих проблем. TCP ідеально підходить для переміщення невеликих обсягів даних через глобальну мережу Інтернет, але не прийнятний для доставки великих обсягів даних – у цьому випадку час обміну даними (RTT) збільшується.

Amazon за допомогою програмного забезпечення Aspera Software вирішила цю проблему, вилучивши рівень TCP. Новий протокол Fast and Secure Protocol (FASP) було розроблено для прискорення передавання різних даних в умовах великого часу відгуку і втрати пакетів. Базою слугує UDP, що є допоміжним транспортним протоколом TCP. UDP дає змогу керувати заторами, передаючи цей аспект до протоколу прикладного рівня FASP (рис. 3).

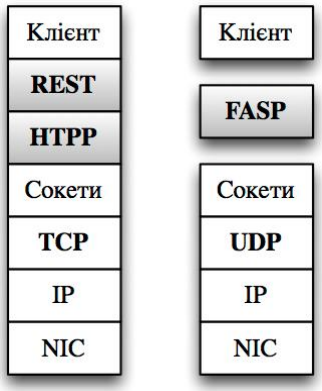

Рис. 3. Архітектура протоколу Fast and Secure Protocol, побудовано за даними [2]

Працюючи зі стандартними мережевими адаптерами (без прискорення), FASP ефективно використовує доступну для додатків смугу пропускання і вилучає головні вузькі місця традиційних схем масової передачі даних.

Багатокористувацькість. Одна з ключових особливостей хмаркової архітектури є її багатокористувацькість. Це означає, що сховище використовується багатьма користувачами (або "орендарями"). Ця особливість архітектури стосується різних рівнів хмаркової системи зберігання, від рівня додатка, де користувачам виділяються ізольований простір імен, до рівня зберігання, де окремим користувачам або категоріям користувачів можуть виділятися окремі фізичні накопичувачі. Використання багатьма користувачами поширюється навіть на мережеву інфраструктуру, яка з'єднує користувачів з накопичувачами, забезпечуючи гарантовану якість обслуговування і виділену смугу пропускання для конкретного користувача.

Масштабованість. Масштабованість можна розглядати з кількох точок зору, але стосовно сховищ даних розглядається як виділення хмаркових ресурсів зберігання на вимогу. Можливість нарощувати ресурси зберігання (як догори, так і донизу) означає поліпшену економічну ефективність для користувача і підвищену складність для постачальника хмаркових послуг.

Масштабованість повинна забезпечуватися не тільки для самої системи зберігання (функціональне масштабування), але й для її пропускної здатності (масштабування навантаження). Ще одна ключова особливість хмаркового зберігання – географічний розподіл даних (географічна масштабованість), яка дає змогу розташовувати дані в максимальній близькості до користувача завдяки групі центрів хмаркового зберігання даних (шляхом міграції). У випадку даних "тільки для читання" можливі також реплікація і розповсюдження (як в мережах розповсюдження аудіо / відеоконтенту). Це проілюстровано на рис. 4.

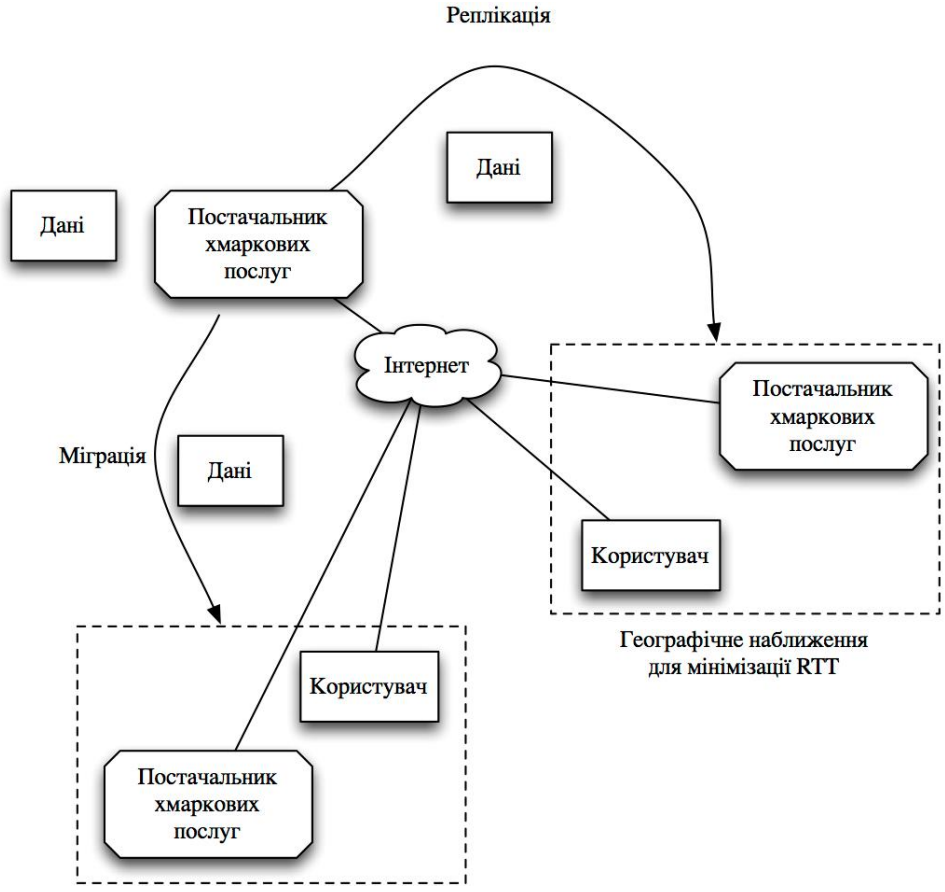

Рис. 4. Масштабованість хмаркової системи зберігання даних, побудовано за даними [2, 9]

Висока готовність. Коли постачальник хмаркових послуг зберігає дані користувача, він повинен мати можливість повернути ці дані користувачеві на вимогу. З урахуванням простоїв

мережі, помилок користувачів та інших обставин виконання цієї умови надійним і детермінованим способом може виявитися складним.

Існують цікаві нові схеми забезпечення високої готовності, такі як розосередження інформації. Компанія Cleversafe, яка надає послуги зберігання даних у приватній хмарці, використовує алгоритм розосередження інформації (Information Dispersal Algorithm – IDA) для підвищення доступності даних при фізичних відмовах і простоях мережі. Алгоритм IDA, який спочатку розробив для телекомунікаційних систем Майкл Рабін, дає змогу "нарізати" дані за допомогою кодів Ріда–Соломона для їх відновлення у випадку втрати частини даних. Крім того, IDA дає змогу настроювати кількість часток даних, так щоб заданий об'єкт даних можна було розрізати на чотири частки при одному допустимому збої або на 20 часток при восьми допустимих збоях. Як і RAID, IDA забезпечує відновлення даних з підмножини вихідних даних при деяких накладних витратах на коди помилок (залежно від кількості допустимих збоїв). Це ілюструє рис. 5.

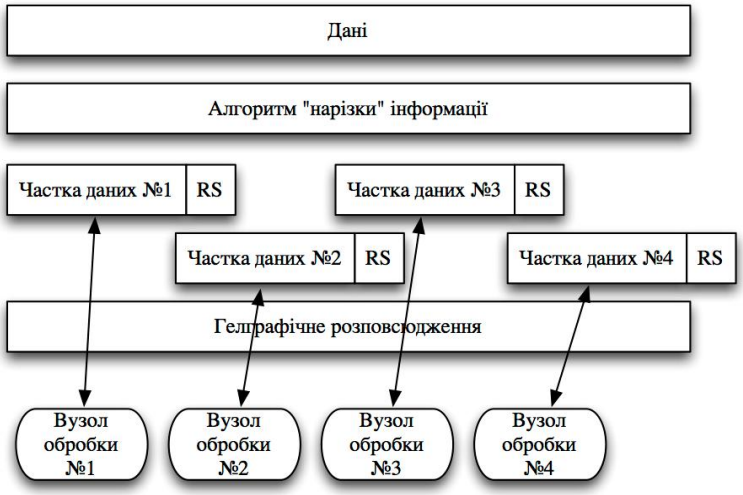

Рис. 5. Підхід Cleversafe до забезпечення високої готовності даних, побудовано за даними [2]

Можливість нарізати дані із застосуванням кодів корекції Ріда-Соломона дає змогу географічно розподіляти накопичувачі. При кількості часток p і допустимої кількості збоїв m результуючі накладні витрати становлять $p/(p-m)$. Так, у випадку, показаному на рис. 5, накладні витрати для системи зберігання при p=4 і m=1 складають 33 %.

Зворотний бік IDA – інтенсивна обробка без апаратного прискорення. Реплікація – ще один корисний метод, який використовують багато постачальників хмаркових послуг. Він простий і ефективний, хоча і накладні витрати великі (прямують до 100 %).

Управління. Важливе значення має здатність клієнта контролювати і управляти тим, як зберігаються його дані, і пов'язаними з цим витратами. Численні постачальники хмаркових послуг пропонують засоби управління, які забезпечують користувачам підвищений контроль над витратами.

Amazon, щоб надати користувачам засоби мінімізації загальних витрат на зберігання даних, застосовує Reduced Redundancy Storage (RRS). Дані реплікуються в інфраструктурі Amazon S3, але RRS дозволяє реплікувати їх меншу кількість разів з можливістю відновлення у випадку втрати даних. Це ідеально підходить для даних, які можна відтворювати, або коли копії даних розташовуються в різних місцях. Nirvanix також забезпечує реплікацію на основі правил, допускаючи детальніший контроль над тим, де і як зберігаються дані.

Ефективність. Ефективність зберігання даних – важлива характеристика хмаркової інфраструктури зберігання, особливо враховуючи її акцент на загальну економію.

Щоб зробити систему зберігання ефективнішою, потрібно зберігати більше даних. Загальним рішенням є скорочення обсягу вихідних даних, щоб вони займали менше фізичного простору. Два способи досягнення цієї мети: стиснення – упакування даних шляхом їх кодування з використанням

різних представлень – і дедублікація – вилучення всіх дублікатів даних. Хоча обидва методи корисні, стиснення передбачає обробку (перекодування даних в інфраструктуру і з неї), а дедублікація – обчислення сигнатур для пошуку дублікатів.

Вартість. Одна з найпомітніших особливостей хмаркового зберігання даних – здатність забезпечити економію. Це економія на придбання накопичувачів, їх енергопостачання, ремонт, а також на управління зберіганням. Якщо розглядати хмаркове зберігання з цього погляду (включаючи SLA і підвищену ефективності зберігання), воно може виявитися вигідним за певних моделей використання.

Цікавий приклад вирішення хмаркового зберігання представляє компанія Backblaze. Вона побудувала недорогу систему зберігання спеціально для хмаркових пропозицій. Backblaze POD (поличка накопичувачів) містить 67 ТБ дискового простору в корпусі висотою 4U за ціною меншою за $ 8000. Комплекс складається з корпусу 4U, системної плати, 4-ГБ ОЗУ, чотирьох контролерів SATA, 45 жорстких дисків SATA ємністю по 1,5 ТБ і двох блоків живлення. На системній платі Backblaze працює Linux (з JFS в якості файлової системи) і мережеві адаптери GbE як зовнішній інтерфейс з використанням HTTPS і Apache Tomcat. У програмне забезпечення BackBlaze входять засоби редуплікації, шифрування і RAID6 для захисту даних. Із запропонованого BackBlaze опису її POD (де в деталях показано, як побудувати свій власний) видно, якою мірою компанія може скоротити вартість зберігання, що робить хмаркове зберігання даних доцільним і економічно ефективним рішенням [9, 13].

При розгляді інфраструктури хмаркових сховищ даних доцільно звернути увагу на моделі інтеграції цих сховищ. У більшості випадків це проводиться за допомогою програмних інтерфейсів. [13]

API доступу до сховища є найважливішим компонентом послуг. Багато додатків вимагають доступу до сховища послуг з використанням API, який оптимізований для цієї конкретної системи зберігання даних, або на власному обладнанні, або хмарним. Так, хмаркова система зберігання Amazon S3 API надає розробникам SDK для .Net і Java, а також бібліотеки для додаткових платформ і мов. Ці інтерфейси зазвичай використовують протоколи веб–сервісів передавання репрезентативного стану (REST) та/або простий протокол доступу до об’єктів (SOAP) [11, 14].

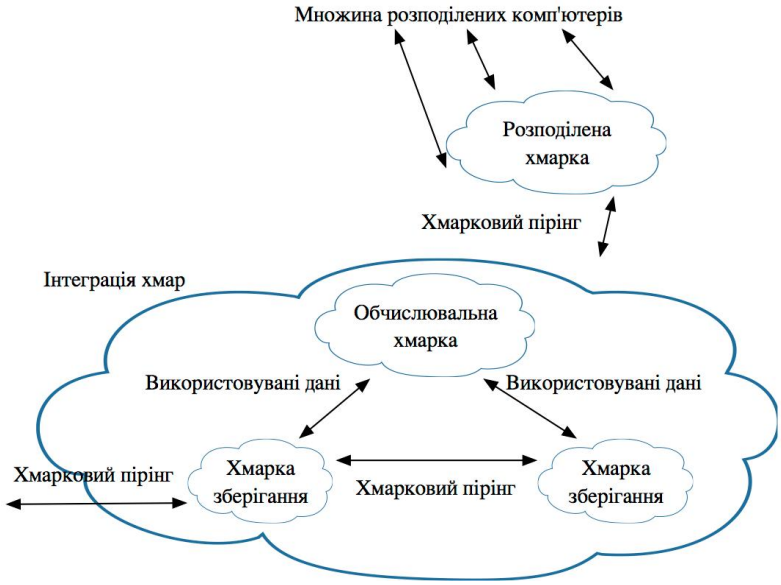

Рис. 6. Моделі хмаркової екології, побудовано за даними [2, 9, 11]

Багато провайдерів хмаркового зберігання почали запроваджувати відкриті API, але більшість з них тільки рухається в цьому напрямку. Користувач може просто перетягувати файли з локальних систем до провайдера хмаркового зберігання і назад. Всі провайдери хмарних систем зберігання даних пропонують зрозумілий інтерфейс користувача, який подібний до інтерфейсу операційної системи.

Плануючи реалізацію, слід керуватися основними принципами. Стандартне обладнання та процеси оперативного аналізу якості необхідно використовувати як основу, на якій будувати інфраструктуру центру обробки даних. Це зменшує розростання, знижує витрати і спрощує управління. Програмне забезпечення дає змогу, не порушуючи масштабування даних, створити і легко переміщувати модулі зберігання та зміни конфігурації для зростання. Інструменти, які полегшують управління через сервери, системи зберігання та мережі, мають широко використовуватись [11].

Дотепер було розглянуто головним чином постачальників хмаркових послуг зберігання даних, але існують хмаркові моделі, які дають змогу користувачам зберігати контроль над своїми даними. Хмаркове зберігання розвивається в трьох напрямах, один з яких допускає злиття двох інших для досягнення економічної ефективності та безпеки [4].

Постачальники загальнодоступних хмарних систем зберігання даних надають інфраструктуру на умовах оренди (ресурси для довгострокового або короткострокового зберігання даних і смугу пропускання мережі) [13]. Приватні хмари використовують ті самі концепції, що й загальнодоступні, але в такій формі, що інфраструктура може бути надійно вбудована в приватну мережу користувача. Нарешті, гібридні хмарні системи зберігання дають змогу з'єднати обидві моделі, визначаючи правила, що регулюють те, які дані необхідно зберегти в приватному володінні, а які можна захистити в межах публічних хмар (рис. 7).

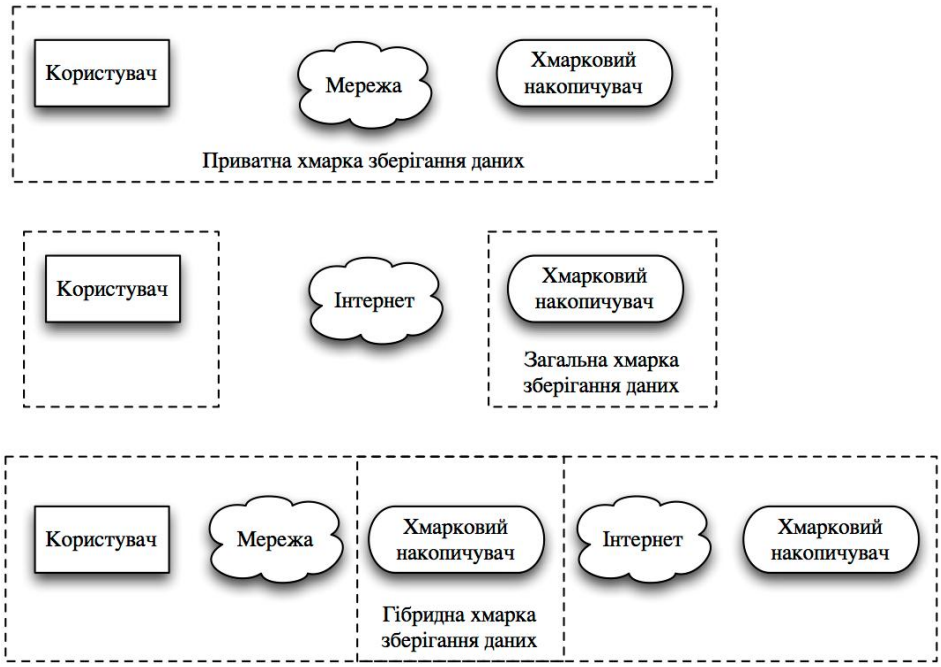

Рис. 7. Хмаркові моделі зберігання даних, побудовано за даними [2]

На рис. 7 хмаркові моделі показані графічно. До постачальників загальнодоступних хмар зберігання даних належать Amazon і Nirvanix (які пропонують зберігання даних як послуги). Прикладами постачальників приватних систем зберігання слугують IBM, Parascale і Cleversafe (яка пропонує програмне забезпечення та / або обладнання для внутрішньої хмарки). Нарешті, постачальники гібридних хмар – це Nirvanix, Egnyte та інші.

Висновки

Хмарка зберігання даних – це перспективний і цікавий напрямок розвитку моделей зберігання, який відкриває нові можливості для побудови, доступу та адміністрування систем зберігання даних на підприємствах та установах. Хоча сьогодні хмаркова система зберігання – переважно споживча технологія, вона швидко розвивається в напрямку підприємств. Гібридні моделі хмар дають змогу організаціям зберігати конфіденційність своїх даних у межах локальних

центрів обробки даних, передаючи менш конфіденційні дані в хмарку для економії витрат і географічного захисту.

Як показано в аналізі різних підходів до інфраструктури сховищ та застосування моделей зберігання, сьогодні визначено декілька основних напрямків проектування хмаркових сховищ. Вибір оптимального варіанта реалізації переважно залежить від логічної реалізації сховища та розрахунку навантаження, оскільки швидкий розвиток засобів комунікації постійно знижує ціни на міжмережеві зв'язки. Своєю чергою, логічна структура хмаркового сховища визначається її функціональним призначенням – оперативний і поширений доступ до даних, незалежно від місце перебування користувача даних.

1. Internet small computer systems interface (iscsi), RFC 3720 / J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner. -2004. 2. Jones M. T. Anatomy of a cloud storage infrastructure - models, features, and internals.// IBM DeveloperWorks, 2010. 3. An iscsi design and implementation / Hui Xiong, Renuga Kanagavelu, Yaolong Zhu,Khai Leong Yong // In In 12-th NASA Goddard, 21-st IEEE Conference on Mass Storage Systems and Technologies (MSST2004), 2004. 4. A cloud storage architecture model for data-intensive applications / Yanmei Huo, Hongyuan Wang, Liang Hu, H. Yang. // In Computer and Management (CAMAN), 2011 International Conference on, 2011, 1–4. 5. Oceanstore: An architecture for global-scale persistent storage./ J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao. //SIGPLAN Not., №11, 2000, 190–201. 6 Pautasso C., Restful web services vs. big web services: Making the right architectural decision. / C. Pautasso, O. Zimmermann, F. Leymann /Beijing china, №25. 2008. 7. Amazon s3 for science grids: a viable solution? / M. Palankar, A. Onibokun, A. Iamnitchi, M. Ripeanu. 8. Early observations on the performance of windows azure. /Z. Hill, J. Li, M. Mao, A. Ruiz-alvarez, M. Humphrey. // In In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pages 367–376. ACM, 2010. 9. Scale and performance in a distributed file system. / J.H. Howard, M.L. Kazar, S.G. Menees, A. Nichols, M. Satyanarayanan, R.N. Sidebotham, M.J. West. // ACM Transactions on Computer Systems, pages 51–81, 1988. 10. Cloudcmp: Comparing public cloud providers. / A. Li, X. Yang, S. Kandula, M. Zhang. 11. Rosenblum M., The design and implementation of a log-structured file system. / M. Rosenblum, J.K. Ousterhout // ACM Transactions on Computer Systems, 10, 1992. 1–15. 12. Webdav - based hypertext annotation and trail system. // S.K. Mark, M. Slater, E.J. Whitehead, H. Factors. 13. Ghemawat S., The google file system / S. Ghemawat, H. Gobioff, Shun-Tak Leung. // 2003. 14. Efficient replica maintenance for distributed storage systems / Byung gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, Robert Morris. // In IN PROC. OF NSDI, 2006. 45–58.