УДК 004.75

ТЕЛЕНИК С.Ф.,
БУКАСОВ М.М.,
КАРНАУХОВ О.К.,
ФІЛІМОНОВ В.Ф.,
МОРГУН М.В.

МОДЕЛІ ОПТИМІЗАЦІЇ БАГАТОРІВНЕВОГО ЗБЕРІГАННЯ ДАНИХ

Розглянуто особливості збереження даних у багаторівневих сховищах в умовах «великих даних». Сформульовано задачу оптимального розташування блоків даних у багаторівневих сховищах з урахуванням технічних характеристик окремих рівнів збереження даних. Запропоновано три формальних моделі задачі, що враховують особливості сховищ та запитів користувачів до даних і політики щодо мінімізації витрат. Запропоновано методи розв'язання цих задач. Розроблено варіант генетичного алгоритму для визначення оптимального розміщення блоків даних на різних рівнях зберігання даних.

The particularities of data storing in multilevel data storages under "big data" conditions are considered. For multilevel data storages the task of data blocks optimal allocation with taking in consideration technical chara-?teristics of individual storage levels is formulated. Three models for different variants of data storage types, user requests and expenditures minimization policies are proposed. Methods of solving these problems were proposed. The variant of genetic algorithm to determine the optimal allocation of data blocks onto different levels of data storage is developed.

Вступ

З кожним роком в умовах глобалізації і проникнення інформаційних технологій (ІТ) у всі галузі людської діяльності породжуються все більші обсяги даних. Відповідно зростають потреби у зборі, збереженні і обробленні великих обсягів даних, відборі з них корисних і прийнятті на їх основі ефективних рішень. Так, в системах управління великих провайдерів інфокомунікаційних сервісів накопичуються терабайти даних про послуги, їх надання, клієнтів, характеристики обладнання і технологій, відмови, технічне обслуговування тощо.

Процес постійного накопичення даних ускладнює роботу з ними, оскільки одночасно скорочується час, відведений на прийняття рішень. Накопичені дані є дуже цінними для бізнесу, але якщо ними не можна скористатися вчасно, то їхня корисність починає викликати сумніви. Тому необхідно навчитися контролювати великі інформаційні потоки, ефективно опрацьовувати та використовувати накопичені дані. Постає складний комплекс проблем організації і контролю великих динамічних інформаційних потоків даних, збору, збереження і оброблення великих обсягів даних, оперативного забезпечення користувачів адекватною для прийняття рішень інформацією.

Класичний підхід до збереження і оброблення даних у різноманітних інформаційних та інформаційно-управляючих системах все гірше виконує покладені на нього функції.

Великі постачальники програмно-апаратних засобів збору, збереження і оброблення даних, насамперед EMC, Cisco, NetApp, Veritas, пропонують нові рішення на базі концепції "big data" («великих даних») [1, 2]. Важливою особливістю цих рішень є намагання розробити нові або модифікувати відомі системи і технології, щоб адаптувати системи збереження даних (СЗД) під нові більш жорсткі умови.

Система збереження даних – це комплексне програмно-апаратне рішення з організації надійного збереження інформаційних ресурсів і забезпечення гарантованого доступу до них [3].

При роботі з даними у СЗД в сучасних умовах в цих системах визначають наступні загальні властивості:

- доступність;
- захищеність;
- масштабованість;
- місткість;
- керованість.

Також окремі частини або блоки збереженої інформації мають певні статичні і динамічні властивості. До них найчастіше відносять цінність, інтенсивність використання та інші інтегральні характеристики в залежності від задач. Значення і динаміка змін значень визначають життєвий цикл даних, управління яким визначає ефективність використання збереженої ін-

формації і інформаційної системи у цілому. На сьогодні управління життєвим циклом даних – Information Lifecycle Management (ILM) становить невід'ємну складову інформаційних та інформаційно-управляючих систем підприємств і організацій [4].

Загалом ILM охоплює всі процеси – від створення даних до їх видалення – і все, що відбувається з даними у цей період. Призначення ILM полягає в тому, щоб організувати процеси і технології для управління даними з метою підтримання найбільш вигідного і ефективного життєвого циклу інформації. При цьому необхідно враховувати залежність IT-інфраструктури підприємства чи організації від об'єктів бізнесу і зміну значимості даних з плином часу. Дійсно, дані, що є важливими для бізнесу сьогодні, можуть бути майже незначними завтра. Дані досліджень, наприклад [5], свідчать, що приблизно 90 відсотків збереженої інформації можуть ніколи не бути використанні після 90 днів від моменту їх створення. Тому, коли дані старіють, вони повинні бути переміщенні автоматично на дешевші сховища і технології, в залежності від їх релевантності, щоб зберегти кошти і звільнити ресурси сховищ для нових даних [5].

Отже, ILM – це не програмно-апаратне рішення, а організаційно-технологічна концепція, що за рахунок реалізації системи визначених процесів забезпечує:

- стратегію управління даними;
- орієнтацію на бізнес;
- централізацію;
- цілісність;
- гетерогенність;
- оптимізацію ресурсів.

Реалізація ILM включає в себе певний спектр дій. Перш за все, має бути здійснена класифікація самих даних і програм їх оброблення. На наступному кроці визначаються стратегії створення і видалення даних, в залежності від їх класу, об'ємів та інших параметрів. Окремим аспектом виділяють управління процесами збереження потоків вхідних даних, їх збереженням та екстракцією. Логічним завершенням концепції ILM виглядає описання багаторівневого збереження даних на різних видах носіїв, які, в свою чергу, розділяють за параметрами ціни, швидкості доступу, надійності та ін.

Для інтеграції ILM у інформаційну чи інформаційно-управляючу систему необхідно виконати такі дії:

- встановити систему зберігання даних;
- класифікувати дані (рівні, цикли, політики);
- виконати розподіл даних між рівнями;
- автоматизувати основні процеси роботи з даними;
- повністю інтегрувати засоби ILM в усі програми.

Ручний розподіл даних між рівнями потребує значних зусиль аналітиків на кожній окремій реалізації багаторівневої СЗД. Тому все більшої популярності набирає архітектура СЗД з повністю автоматизованим багаторівневим зберіганням – Fully Automated Storage Tiering (FAST).

Наприклад, ієрархія СЗД може мати такі рівні:

- cache;
- SSD;
- SATA;
- IDE;
- Tape Library.

Підхід, при якому забезпечення ефективності системи збереження відбувається за рахунок переміщення між її рівнями (tiers) збереження називається міжрівневим перерозподілом даних (Tiering). Кожен із рівнів характеризується власним набором параметрів, таких як ціна, швидкодія, ємність, захищеність та інші. Зазвичай міжрівневий перерозподіл даних реалізується механізмами переміщення даних між дисками різних рівнів, або між дисками та магнітною стрічкою. Часто він використовується для організації ILM-сховищ даних у відповідності з визначеними статусом даних та рівнем QoS.

Сьогодні найбільшого поширення набули такі рішення для міжрівневого перерозподілу даних:

- EMC FAST;
- EMC FASTcache;
- NetApp FlashCache.

Диски, які найчастіше використовуються в СЗД відрізаються продуктивністю, ємністю, вартістю. Причому за однакової ціни продуктивність та ємність зазвичай величини обернено пропорційні. Так, для дисків SATA характерна більша ємність при меншій продуктивності, для дисків SAS – менша ємність при більшій продуктивності, а для SSD – ще більша продуктивність при ще меншій ємності. У цілому СЗД характеризуються широко вживаним відношенням «IOPS/вартість».

У загальному випадку невелика частина так званих активних даних використовується більшу частину часу роботи, і тому в більшості випадків визначають швидкодію системи. В противагу, значна частина даних належить до так званих «холодних» даних, які впливають на загальну швидкодію набагато менше. Оскільки в окремих системах холодні дані можуть містити частини або блоки, швидкість доступу до яких є критичною, це треба враховувати при проектуванні.

Розглянемо систему, у якій всі дані є рівноцінними і 20% з них становлять активні даних, швидкість доступу до яких визначає загальну продуктивність системи. Очевидно, що такі дані доцільно розмістити на максимально швидкому носієві (хоч і дорогому). Результатом буде суттєвий приріст швидкодії роботи системи у цілому.

Саме така ідея лежить в основі технологій міжрівневого перерозподілу даних. Якщо певний тип даних є активним, він автоматично переноситься на більш дорогі та швидкі носії, що призводить до підвищення продуктивності. Одночасно менш активні дані переносяться на менш продуктивні та дешевші типи дисків. Таким чином можна контрольовано підвищувати співвідношення IOPS/вартість.

Для великих систем розбиття на рівні є набагато складнішою задачею, оскільки сьогодні велика частина застосувань розташована в центрах обробки даних (ЦОД), а розподілені застосування розташовуються одразу у декількох з них. Також такі застосування можуть використовувати не одну СЗД, а декілька, що додатково ускладнює розбиття усієї множини СЗД на рівні.

Кожен ЦОД має свої СЗД, сервери застосувань та інші компоненти IT-інфраструктури, що дозволяє забезпечувати швидкий доступ застосувань до інформації, що розташована безпосередньо у СЗД цього ЦОД. Але якщо система розподілена, то застосування має отримувати інформацію, що розміщена у різних ЦОД, а швидкість доступу до даних сильно залежить від відстані до інших ЦОД, швидкості, відгуку і інших параметрів мережі.

Говорячи про ЦОД, важливо зазначити його основні властивості, можливості та структуру. Узагальнена структура СЗД з використанням декількох ЦОД наведена на рисунку 1.

ЦОД може складатися з великої кількості різнотипних сховищ (в тому числі готових рішень від відомих компаній, таких як EMC). Сховища, в свою чергу, складаються з стійок дисків та окремих накопичувачів.

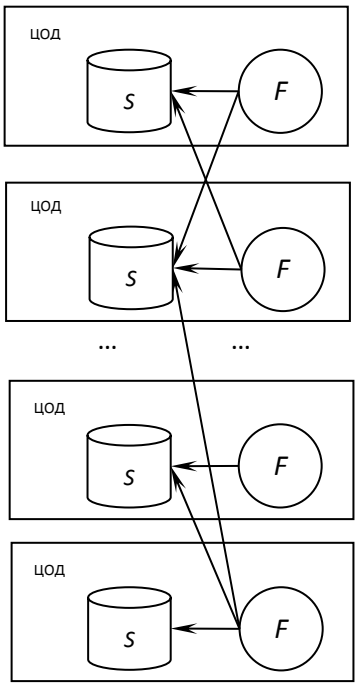

Рис. 1. Схема IT-інфраструктури. S - сховище, F - застосування

Однією з найважливіших характеристик СЗД, ЦОД і сховищ є їх живучість і надійність. Основним з відомих методів підвищення цієї характеристики лишається реплікація даних в кілька незалежних сховищ. В ідеалі, місця збереження дубльованих даних мають бути залежні від різних локальних чи глобальних мереж, а також різних джерел живлення. Важливо розуміти, що дублюватися, в залежності від потреб, можуть як окремі дані, так і диски, стійки, сховища в цілому.

Загальновідомо, що всі бізнес-рішення мають акцент на зменшенні витрат без втрат якості. Ще одним способом економії, а значить і оптимізації, є знеструмлення окремих сховищ (стійок, дисків) з непотрібними в певний час даними. Наприклад нічне виключення пристроїв, що зберігають «холодні» дані, або дані, які точно не будуть потрібні у неробочий час. Маємо ще одну проблему – аналіз і подальше налаштування системи таким чином, щоб відключення в цілях економії не завадили бізнес-процесам, а також налаштування самого процесу знеструмлення без фатальних наслідків, оскільки ЦОД, окрім самих носіїв має сервери налаштувань та іншу інфраструктуру.

Стосовно оптимізації розміщення даних у розподілених системах, очевидно, що чим швидше застосування отримує інформацію з СЗД, тим швидше воно працює. Тому для покращен-

ня роботи системи доцільно розміщувати дані на носії, до яких має швидший доступ з врахуванням конкретного застосування, що їх використовує. Тобто постає проблема оптимального розташування даних у розподіленій СЗД для покращення показників швидкодії певного застосування. І, на відміну від системи, що складається з однієї СЗД і одного серверу застосувань, виникає ситуація, коли вигідніше тримати інформацію на менш швидких носіях, але менших за часом доступу до серверу застосувань. В такій системі також важливо враховувати інтенсивність використання даних окремими серверами застосувань, щоб правильно розрахувати оптимальне розташування.

Для серверів застосувань розташованих в одному ЦОД, можливо розбити СЗД з різних ЦОД на рівні за такими критеріями:

- Швидкість доступу до сховища. Сховище виступає як одиниця.
- Продуктивність носія. Диск виступає як одиниця.
- Швидкість доступу до носія. Диск виступає як одиниця.

Під одиницею мається на увазі нероздільна частина рівня.

Для серверів застосувань, що розташовані в різних ЦОД:

- Сума добутку показників швидкості доступу до сховища і інтенсивності звернень серверу застосувань з яким виконується обмін. Сховище – одиниця.
- Сума добутку показників швидкості доступу до носія і інтенсивності звернень серверу застосувань з яким виконується обмін. Сховище – одиниця.

На жаль, формальні моделі і відповідні методи розв'язання задач, пов'язаних з підвищенням ефективності застосування багаторівневих систем збереження даних, у літературних джерелах не описані. Тому виникає нагальна потреба у таких моделях і методах, насамперед оптимального переміщення даних між сховищами з різними технічними характеристиками.

Мета дослідження

Метою дослідження є підвищення ефективності використання багаторівневих систем збереження даних за рахунок розроблення і використання моделей і методів розв'язання задачі оптимального переміщення блоків даних між дисками сховищ з різними технічними характеристиками.

Математична модель

Розглянемо випадок, коли функції застосування звертаються до блоків даних таким чином, що кожна функція може звертатись до декількох блоків, а до кожного блоку можуть звертатись декілька функцій. При цьому, кожен блок даних має бути розміщеним у одному і лише одному сховищі даних. Модель доступу до наведена на рисунку 2.

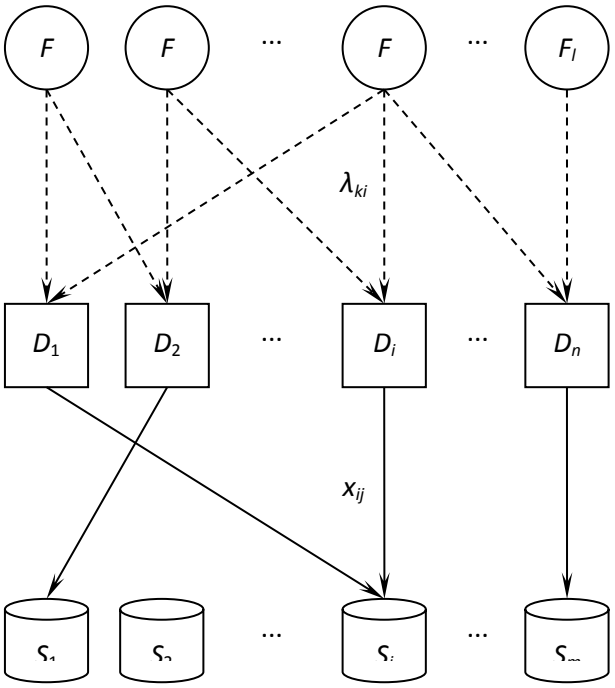

Рис. 2. Модель доступу до даних

Введемо такі позначення:

F_k – функції застосувань, $k = 1, \ldots, l$;

μ_k – інтенсивність викликів функції F_k;

D_i – блоки даних, які використовуються застосуваннями, $i = 1, \ldots, n$;

d_i – розмір блоку даних D_i;

λ_{ki} – інтенсивність звернень функції F_k до блоку даних D_i;

S_j – фізичні сховища, на яких розташовуються блоки даних, $j = 1, \ldots, m$;

t_j – час довільного доступу до сховища S_j;

s_j – розмір сховища S_j;

x_{ij} – булева змінна, що визначає, чи розташовано блок D_i даних у сховищі S_j;

Оскільки кожний блок даних може бути розташований лише у одному сховищі, має виконуватись умова:

$$\sum_{j=1}^{m} x_{ij} = 1,\ i = 1, \ldots, n. \qquad (1)$$

Крім того, сума розмірів блоків даних не повинна перевищувати розмір сховища, на якому вони розташовані:

$$\sum_{i=1}^{n} d_i \cdot x_{ij} \le s_j \,, j = 1, \ldots, m. \qquad (2)$$

Задача 1. В умовах обмежених ресурсів системи збереження даних і відсутності жорстких вимог до часу відгуку функцій, виникає задача оптимального розміщення блоків даних по дисках таким чином, щоб середня продуктивність системи була максимальною, тобто середній час доступу до даних був мінімальним.

Середня інтенсивність звернень до блоку даних D_i може бути представлена наступним виразом:

$$\sum_{k=1}^{l} \mu_k \lambda_{ki} \,. \qquad (3)$$

Тоді інтенсивність звернень до сховища S_j буде визначатись виразом:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\sum_{k=1}^{l} \mu_k \lambda_{ki}\right) \cdot x_{ij} \,, \qquad (4)$$

а критерій мінімізації середнього часу доступу можна записати наступним чином:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\sum_{k=1}^{l} \mu_k \lambda_{ki}\right) \cdot x_{ij} \cdot t_j \,. \qquad (5)$$

Тоді задачу мінімізації середнього часу доступу можна сформулювати наступним чином: мінімізувати (5) при виконанні обмежень (1), (2).

Задача 2. Розглянемо випадок, коли є обмеження на час відгуку при викликах функцій додатків. У найгіршому випадку при виконанні функції доступ до блоків даних буде виконуватись послідовно. Позначимо максимально допустимий час доступу до даних функції F_k як T_k. Вочевидь, він має перевищувати сумарний час доступу до усіх блоків даних, які використовуються даною функцією. Тобто,

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_{ki} \cdot x_{ij} \cdot t_j \le T_k \,, k = 1, \ldots, l. \qquad (6)$$

Тоді задачу мінімізації середнього часу доступу можна сформулювати наступним чином: мінімізувати (5) при виконанні обмежень (1), (2), (6).

Задача 3. У випадках, коли сховище даних побудовано за концепцією MAID (massive array of idle disks) [5], яка передбачає автоматичне відключення живлення накопичувачів, які тривалий час не використовуються, природньою постає задача розміщення блоків даних у сховищах якнайбільш щільно, при цьому саме таким чином, щоб вивільнити частину накопичувачів і за рахунок їх вимкнення мінімізувати витрати на їх живлення та охолодження.

Позначимо енергоспоживання сховища S_j як e_j, $j = 1, \ldots, m$.

Ознакою того що у сховищі S_j не розміщено жодного блоку даних буде визначати наступний вираз:

$$\prod_{i=1}^{n} \overline{x_{ij}} = 1, \; j = 1,\ldots,m \,. \qquad (7)$$

Для мінімізації енергоспоживання потрібно розмістити блоки даних таким чином, щоб можна було вимкнути частину накопичувачів з максимальним енергоспоживанням, тобто:

$$\max \sum_{j=1}^{m} e_j \prod_{i=1}^{n} \overline{x_{ij}} \,. \qquad (8)$$

Тоді задачу мінімізації енергоспоживання можна сформулювати наступним чином: максимізувати (8) при виконанні обмежень (1), (2), (6).

Наведені задачі являють собою лінійні задачі булевого програмування і при невеликій кількості незалежних змінних можуть бути ефективно розв'язані шляхом застосування точних методів [6].

Якщо ж розмірність задачі висока, то для її розв'язання найбільш доцільно буде запропонувати використання різних варіантів генетичного алгоритму (ГА) [7]. При цьому пропонується наступна схема кодування генів.

Оскільки кожний блок даних може бути розташований не більше ніж у одному сховищі, для кодування генів перейдемо від матриці $n \times m$ булевих змінних x_{ij} до вектора довжини n дискретних змінних $y_i \in [1, m]$, кожний елемент якого містить номер сховища S_j, (j=1, ... , m), у якому розміщений відповідний блок даних (рис. 3).

Такий спосіб кодування генів дозволяє, по-перше, скоротити розмірність задачі в m разів, а по-друге, забезпечити автоматичне виконання обмежень (1), оскільки така схема кодування гарантує, що кожний блок даних буде розташований у одному і лише одному сховищі. При цьому операція мутації фактично буде відповідати переносу блоку даних з одного сховища у інше, а операція кросинговеру – обміну декількох блоків даних між сховищами.

$$x_{ij} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \longrightarrow y_i = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 2 \\ 1 \\ 3 \\ 1 \\ 1 \end{bmatrix}$$

Рис. 3. Приклад кодування генів

Для розв'язання подібного класу задач добре себе зарекомендував керований генетичний алгоритм (КГА) [8].

Висновки

У статті розглянуто підхід до підвищення продуктивності багаторівневої системи збереження даних, що базується на періодичному переміщенні даних між носіями різної продуктивності. Запропоновані три моделі оптимального розміщення блоків даних за різними критеріями, що враховують статистику звернень до інформації, обмеження на ресурси і бізнес-вимоги до системи. Ці моделі можуть бути використані для підвищення швидкодії систем збереження даних. Розв’язання поставлених задач виконується на основі відомих точних методів або запропонованого варіанта генетичного алгоритму в залежності від кількості вільних змінних.

Перелік посилань

1. Mayer-Schönberger V. Big Data: A Revolution That Will Transform How We Live, Work, and Think / V. Mayer-Schönberger, K. Cukier. – Houghton Mifflin Harcourt, 2013. – 240 p.
2. Marz N. Big Data: Principles and Best Practices of Scalable Realtime Data Systems / N. Marz, J. Warren. – Manning Publications Company, 2015. – 328 p.
3. Schulz G. Cloud and Virtual Data Storage Networking / G. Schulz. – CRC Press Taylor & Francis Group, 2011. – 370 p.
4. Information Lifecycle Management [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: http://i.bnet.com/whitepapers/sun00.pdf
5. Colarelli D. Massive arrays of idle disks for storage archives / D. Colarelli, D. Grunwald. // Proceedings of the 2002 ACM/IEEE conference on Supercomputing. – 2002. – С. 1-11.
6. Зайченко Ю.П. Дослідження операцій / Ю.П. Зайченко – [7-е видання, перероблене та доповнене.] – К.: Видавничий Дім «Слово», 2006. – 816 с.
7. Goldberg D.E. Genetic algorithms in search, optimiza-tion, and machine learning / D.E. Goldberg. – Addison-Wesley, 1989. – 412 p.
8. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплекси та системи. – 2010. – № 1(25). – С. 106-120.