УДК 621.3

В.Ф. Третяк, А.А. Пашнєва

Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків
Харківський національний університет радіоелектроніки, Харків

ОПТИМІЗАЦІЯ СТРУКТУРИ СХОВИЩА ДАНИХ У ВУЗЛАХ ІНФОКОМУНІКАЦІЙНОЇ МЕРЕЖІ ХМАРНОГО СЕРЕДОВИЩА

Головною ідеєю статті є аналіз методів і технології роботи з великими даними, аналіз методів інтеграції додатків на рівні даних, а також показати підхід до оптимізації структури сховища даних у вузлах інфокомунікаційної мережі хмарного середовища. Особливу увагу було приділено ранговому підходу до рішення задачі оптимізації структури сховища даних у вузлах мережі хмарного середовища.

Ключові слова: *великі дані ,реплікація, ранговий підхід, сховище даних, інфокомунікаційна мережа, інтеграція, тиражування даних, хмарне середовище, фрагментація.*

Вступ

Актуальність. Аналіз літератури. "Хмарні" технології є частиною нової мережевої інтернет-архітектури, яка базується на трьох основних принципах: інформаційно-орієнтованої мережевої архітектури (information - centric networking); "хмарних" обчисленнях які інтегровані з мережею (cloud computing integrated with networking); відкритої коннективності (open connectivity). "Хмари" відносяться до класу мережевих комп'ютерних систем, основними елементами яких є: комп'ютерна мережа з підвищеною надійністю і пропускною спроможністю; клієнт "хмари" - апаратне і програмне забезпечення, що взаємодіє з "хмарою" на основі стека протоколів TCP/IP; власне "хмара" - програмно-апаратний комплекс, що забезпечує роботу "хмарних" сервісів, взаємодію з клієнтом і динамічне управління ресурсами хмарного середовища. Особливостями "хмарних" технологій є наступні ознаки: сервісна модель обслуговування; самообслуговування; еластичність; використання поширених мережевих технологій. Зазвичай виділяють наступні базові класи "хмарних" сервісів: інфраструктура як послуга (Infrastructure as a Service, laaS); платформа як послуга (Platform as a Service, PaaS); дані як послуга (Data as a Service, DaaS); програмне забезпечення як послуга (Software as a Service, SaaS); робоче місце як послуга (Workplace as a Service, WaaS); усе як послуга (All as a Service, AaaS).

Сучасним підходом до проектування інформаційних систем є напрям хмарних обчислень (Cloud Computing), який містить спеціалізований спектр технологій обробки і передачі даних, коли комп'ютерні ресурси і потужності надаються як Інтернет-сервіси. Специфіка Cloud Computing полягає в тому, що забезпечується динамічне масштабування ресурсів хмари, його внутрішня структура прихована від споживача сервісів, використовується концепція плати у міру використання, пред'являються високі вимоги до надійності і доступності хмарної системи та ін. Вивченням питання організації та ефективної обробки великих даних активно займаються наукові групи під керівництвом провідних світових вчених: Я. Фостера [6], Е. Ділмана [3], Д. Тейна [8], Я. Гордона, Р. Буйя [1], Т. Хейя, Р. Продана [9], Ільїна В.А. [11].

Метою даної статті аналіз методів і технології роботи з великими даними, аналіз методів інтеграції додатків на рівні даних, а також показати підхід до оптимізації структури сховища даних у вузлах інфокумунікаційної мережі хмарного середовища

1. Аналіз методів і технології обробки великих даних

Аналіз показав, що при сучасних розрахунках на великих даних виникають наступні проблеми і тенденції:

- створення нових алгоритмів, які здатні масштабуватися при пошуку і обробці великих масивів даних;
- створення нових масштабованих технологій управління метаданими складних, гетерогенних і розподілених джерел даних;
- створення нових підходів в області високопродуктивних обчислювальних платформ для забезпечення рівномірного високошвидкісного доступу до мультитерабайтних структур даних;
- створення спеціалізованої комунікаційної гібридної архітектури для фільтрації і обробки потоків мультигігабайтних даних, що надходять від високошвидкісних мереж передачі цих, наукових вимірювальних систем і систем моделювання в режимі реального часу;
- розробка високонадійних високопродуктивних розподілених файлових систем, орієнтованих на обслуговування петабайтних масивів даних;
- створення нових алгоритмів для забезпечення мобільності розрахунків на вузлах, вартість передачі даних з яких на інший вузол занадто висока;

- появa гнучких і спрощених технологій, що забезпечують інтеграцію нових плагінів і програмних компонентів, що працюють на різних обчислювальних платформах;
- розвиток методів генерації підписів для даних з метою зменшення розмірності і збільшення швидкості їх обробки.

На сьогодні існує ряд підходів (парадигм), які були розроблені для успішного вирішення завдань обробки великих даних. Передусім, слід виділити два принципово різних режими обробки: пакетний і режим реального часу (потоковий). Пакетний режим припускає обробку статичних даних і зазвичай не накладає обмежень на час виконання розрахунків, орієнтуючись в основному на результат обробки. Розглянемо найбільш відомі підходи до забезпечення цього режиму.

Підхід на основі абстракції Грід [12,10]. Представником класичної концепції обробки великих даних через Грід технології являється система, побудована в CERN, - Gfarm (Grid Datafarm). У основі Gfarm лежать декілька базових компонентів: розподілена файлова система (parallel file system), вузли ресурсів (nodes) і система виконання розрахунків. Розподілена файлова систем складається з розрахункових вузлів і сервісів метаданих, надає величезний об'єм дискового простору (який вимірюэться в петабайтах) і включає можливості масштабування пропускної спроможності на основні операції читання-запису, а також функціональність по забезпеченню відмовостійкості.

Системи, засновані на цій парадигмі, використовують як базу принцип Code-to-Data і виконують запуск обчислювальних розрахунків безпосередньо на вузлах даних, тим самим, ідеологічно не відділяючи їх від обчислювального типу вузлів. Переваги: масштабованість, що дозволяє працювати на рівні петабайтів; можливість планування з урахуванням вартості (у тому числі і за часом) передачі даних для розрахунку і запуску будь-яких пакетів усередині Грід; відмовостійкість. Мінуси: жорстка прив'язка до інфраструктурних особливостей Грід; відсутність підтримки сучасних технологічних рішень (наприклад, обчислювальних хмар).

Підхід на основі абстракції WMS [7]. WMS відділяє абстрактний опис завдання від конкретного ресурсу обчислювального середовища, саме середовище виходить за рамки конкретної системи організації інфраструктури і використовує Грід як один з можливих обчислювачів, тим самим дозволяючи уникати необхідності вивчення його внутрішньої структури і віддаючи право виконання розрахунків безпосередньо йому. Незважаючи на те що концепція, заснована на ланцюжках завдань, спочатку була створена для Грід, на даний момент вона з успіхом застосовується як в хмарних обчисленнях, так і в гетерогенних середовищах.

До основних плюсів цього підходу можна віднести незалежність від платформи, яка використовується, тобто відсутність прив'язки до конкретного обчислювального середовища; багатогранність реалізацій алгоритмів оптимізації планування, у тому числі з урахуванням розрахунків великих даних; високий рівень абстракції при створенні самих ланцюжків завдань. До недоліків відносяться відсутність безпосереднього контролю розрахунків і високорівнева абстракція, які можуть привести до втрат продуктивності; висока невизначеність в оцінці часу виконання кроків ланцюжка задач (workflow, WF) із-за різноманіття параметрів гетерогенного середовища, що враховуються і не враховуються.

Підхід на основі абстракції MapReduce [2]. Основна ідея парадигми криється в операціях Map і Reduce. Перша потрібна для виконання операції обробки на певних даних, що поступають на вхід в наступному виді: map (ключ1, значення1). Результатом застосування Map є список виду (ключ2, значення2). Результат операцій Map подається на вхід завдання Reduce як (ключ2, список(значення2)). Reduce, у свою чергу, генерує список значень. Така проста схема дозволяє вирішувати величезний клас завдань. До явних переваг MapReduce можна віднести простоту використання (досить реалізації двох операцій, Map і Reduce); широку застосовність і технічну підтримку співтовариством, що забезпечує розвиток парадигми; високу швидкість обробки даних завдяки можливості імплементації низькорівневих операцій. До мінусів можна віднести погану абстракцію - користувач не може думати інакше, як в логіці обробки MapReduce; прихильність до обчислювального середовища - традиційно MapReduce працює в єдиному кластері; наявність класів завдань (наприклад, завдань "селекції даних"), що демонструють погану продуктивність при використанні MapReduce за рахунок необхідності повного перебору; невисоку ефективність при виконанні операції Reduce в умовах значного об'єму даних і передпідготовки Shuffle.

Підхід на основі абстракції All – Pairs [5]. Ідея цього підходу полягає у використанні простої операції All - Pairs, яка отримує на вхід функцію F, дві множини A і B і будує на виході матрицю значень F(ai, bi). Проте незважаючи на уявну простоту непродумана реалізація операції може не лише не забезпечити зростання продуктивності, але і привести до її падіння порівняно з однопоточним режимом.

Реалізація All-Pairs дозволяє уникнути консервативної схеми доступу процесів обробки (завдань) до даних, коли система не знає, які дані якому процесу будуть потрібні, до схеми організації як обчислювальних потреб, так і необхідних даних для кожного процесу. Ідею підходу можна представити в чотирьох головних етапах виконання розрахунків: моделювання системи, розподіл даних, організація

процесів обробки, очищення системи. Моделювання системи полягає в оцінці розподілу даних по можливих потенційних процесах обробки з урахуванням вартості передачі даних в розподіленому середовищі, часу і об'єму самої обробки, що дозволяє говорити про наявність концепції Code-to-Data в цьому підході.

Головною перевагою підходу можна назвати "глибоку" участь системи на етапі планування розподілених обчислень, мінусами - вузьку спрямованість вирішуваних завдань, незрілість і відсутність широкої практичної реалізації. На даний момент наукового розвитку цього підходу не спостерігається.

Підходи до обробки великих даних в режимі реального часу [4]. На відміну від пакетного режиму, обробка великих даних в режимі реального часу - відносно новий напрям розвитку ІКТ, і тут можна виділити традиційні підходи, засновані на реалізації загальної черги завдань (task queue) і процесів обробників (workers), і рішення, які подібні Storm і Spark streaming.

Перший підхід широко застосовується внаслідок простої ідеї організації єдиної черги завдань (можливо, і розподіленою) і підключення масштабованого числа розподілених процесів обробки, завдання яких - у міру виконання забрати з черги наступне завдання.

Головним недоліком, і перевагою подібного підходу являється його свобода в реалізації і самостійність в уточненні деталей, що призводить як до вдалих, так і невдалих рішень.

2. Аналіз методів інтеграції додатків на рівні даних

Слід зазначити, що нині жодне велике підприємство не може обходитися без системи, що забезпечує функцій сховища даних. Все більше організацій прагнуть до активних операційних сховищ, тому оперативна обробка транзакцій є найважливішим засобом взаємодії з інформацією, що знаходиться в сховищах даних. Бізнес процеси сховища даних представлено на рис. 1 – 5.

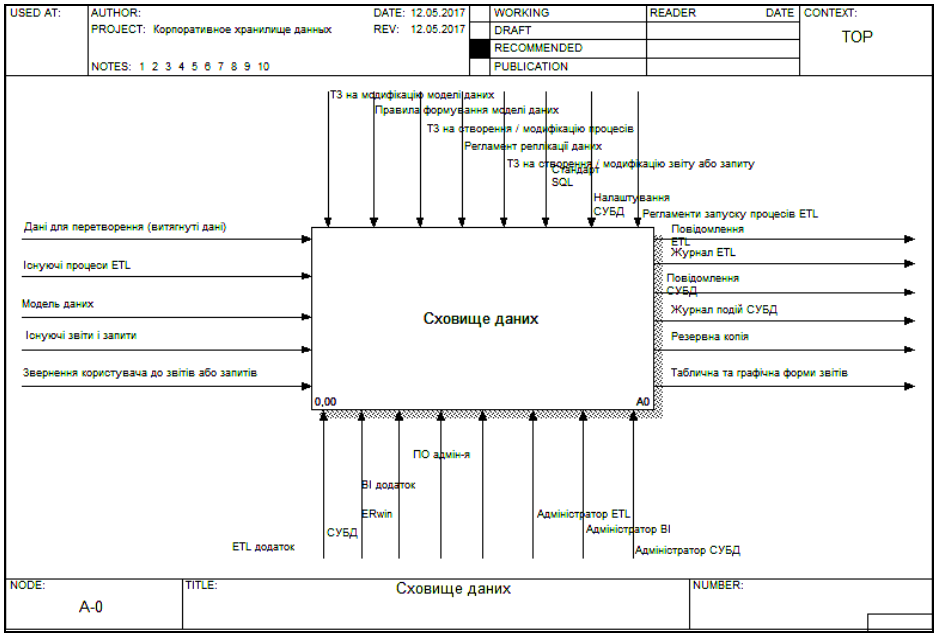

Рис. 1. Контекстна діаграма «Сховище даних»

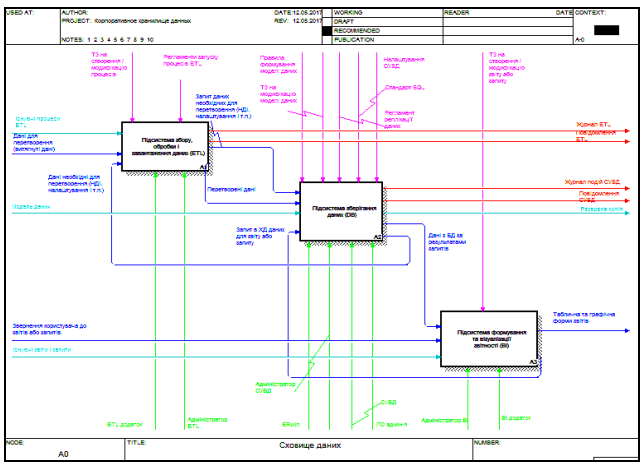

Рис. 2. Декомпозиція задачі «Сховище даних»

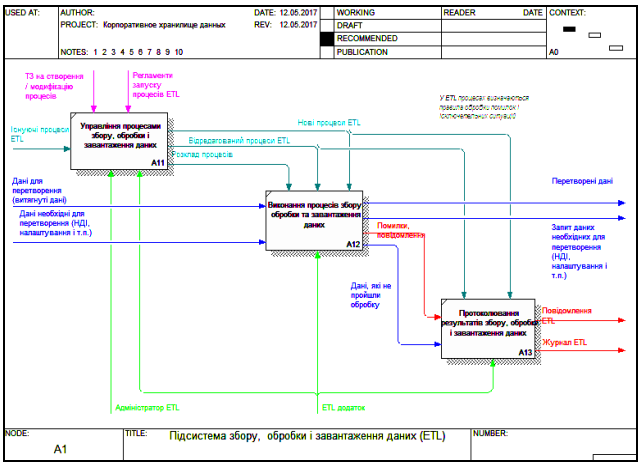

Рис. 3. Підсистема збору, обробки і завантаження даних

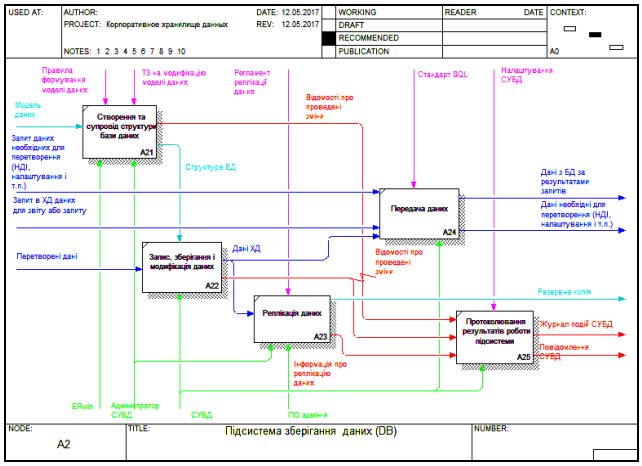

Рис. 4. Підсистема обробки даних

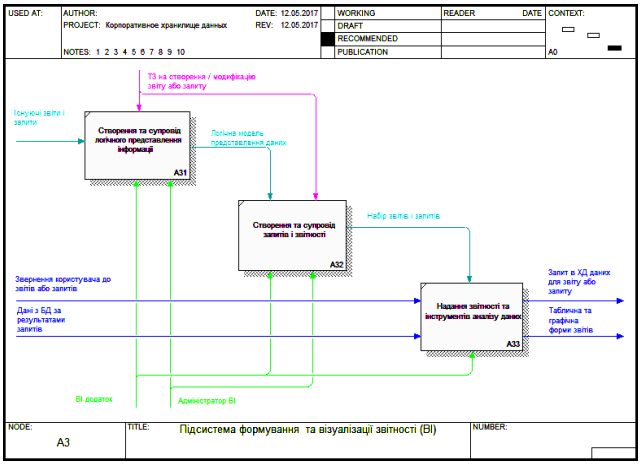

Рис. 5. Підсистема формування звітності

Аналіз публікацій дозволив виділити наступні існуючі методи інтеграції додатків на рівні даних:

– консолідація даних. Технологія, що застосовується при такому методі, має назву ETL (Extract-Transform-Load, тобто Витягування-Перетворення-Завантаження). Цей метод призначений для вилучення необхідної інформації з різноманітних систем, перетворення між вихідним і цільовим форматом і завантаження в цільову систему (наприклад, в сховище даних). Основними недоліками цього методу є: затримка поновлення даних, оскільки дані копіюються з систем з певною періодичністю; підвищені вимоги до потужності цільового місця зберігання;

– федералізація даних. При такому методі кожен з n джерел містить n-1 фрагментів коду, що забезпечують трансляцію запитів до інших джерел федерації і перетворення результатів. Це забезпечує єдину віртуальну картину різнорідних джерел даних. Цей метод позбавляє від необхідності копіювати дані (наприклад, в сховище даних) і дозволяє використовувати дані безпосередньо з джерела. Основним недоліком цього методу є нелінійно зростаюча складність забезпечення віртуальної картини при збільшенні кількості джерел даних;

– поширення даних. За допомогою спеціальних програмних компонентів здійснюється копіювання даних між різними додатками. Копіювання може відбуватися в синхронному або асинхронному режимі. До основних недоліків цього методу можна віднести: підвищені вимоги до потужності споживача даних; обов'язкову присутність кожної програми в мережі при синхронному режимі, а при асинхронному режимі може виникнути ситуація, коли дані в додатках, що синхронізуються не будуть співпадати;

– системи з медіатором. Медіатор - це програмний компонент, який забезпечує єдину точку входу для користувача запитів і єдиний віртуальне бачення різнорідних джерел даних. Медіатор транслює запит користувача до джерел даних на основі загальної схеми і перетворює результати від джерел в єдину форму подання. Кожне джерело даних має адаптер, який перетворює запит із загальної схеми медіатора в схему джерела, а, потім, результати запиту перетворює назад в загальну схему. При підключенні нового джерела даних потрібно створити відповідний адаптер. Основним недоліком такого методу є те, що дані доступні, як правило, тільки для читання;

– системи з посиланням на масив. При такому методі тиражуються в єдине місце зберігання не всі дані з кожного запису, а тільки частина, що використовуються для пошуку джерел даних, в яких містяться необхідні записи. До основних недоліків цього методу відноситься відсутність історичності даних і складна процедура емпіричного формування багатогранної структури єдиного довідкового масиву, зокрема, при додаванні нових джерел даних. Якщо контрольний масив оновлюється з деякою затримкою, то це негативно позначається на актуальності даних. Якщо ж оновлюється без затримки, то це може привести до нестачі ресурсів, необхідних для стабільного функціонування всієї системи (особливо при великій кількості джерел даних).

Розглянуті методи інтеграції даних є варіаціями двох основних механізмів підтримки розподілених СД:

– фрагментація даних - це розбиття СД або будь-якій її таблиці на фрагменти, які фізично зберігаються в різних БД, розташованих на різних вузлах комп'ютерної мережі і, можливо, управляються різними СУБД. Фрагментація даних дозволяє користувачам сприймати ці фрагменти так, як ніби вони працюють з локальною БД. Виділяють два основних види фрагментації таблиць: горизонтальна і вертикальна - це, відповідно, коли рядки і стовпці однієї логічної таблиці розподілені по декільком вузлам.

– реплікація даних - це процес копіювання даних з вихідного СД в цільову БД. При цьому дані можуть копіюватися інтенсивним або інертним способом. Інтенсивний спосіб передбачає, що зміни даних у вихідному СД будуть синхронно внесені в цільову БД як частина однієї транзакції. Інертний

спосіб передбачає, що зміни даних з вихідної БД будуть асинхронно внесені в цільову БД в рамках вже іншою транзакцією. Практично перевага віддається інертному способу, щоб підвищити надійність роботи розподілених ІС, оскільки можна вносити зміни в вихідну БД без необхідності чекати внесення змін до цільової БД, але, оскільки зміни переносяться з певною затримкою, то в якийсь момент дані можуть відрізнятися.

Спосіб оптимізації структури сховища даних у вузлах мережі хмарного середовища.

Розглядаються: n – кількість вузлів мережі з довільною структурою; m – кількість незалежних фрагментів розподіленої бази даних (РБД); K_j – j-й вузол мережі, $j=\overline{1,n}$; F_i – i-й фрагмент РБД, $i=\overline{1,m}$; L_i – об'єм i-го фрагмента; b_j – об'єм пам'яті вузла K_j призначеного для розміщення фрагментів; s – кількість класів запитів (наприклад, читання, додавання, оновлення, видалення записів БД); λ_{ij}^k – інтенсивність запитів k-го класу $\left(k=\overline{1,s}\right)$ до фрагмента F_i ініційованих у вузлі K_j; α_{ij}^k – обсяг запиту k-го класу $\left(k=\overline{1,s}\right)$ до фрагмента F_i, ініційованого у вузлі K_j; β_{ij}^k - об'єм даних по запиту при виконанні запиту k-го класу $\left(k=\overline{1,s}\right)$ до фрагмента F_i, що поступив у вузол K_j [14].

Таким чином об'єм даних, що пересилаються, при виконанні запиту k-го класу до фрагмента F_i, ініційованого у вузлі K_j, визначається таким чином $\left(\alpha_{ij}^k+\beta_{ij}^k\right)\left(1-x_{ij}\right)$. При цьому $x_{ij}\left(i=\overline{1,m};j=\overline{1,n}\right)$ визначається таким чином:

$$x_{ij}=\begin{cases}1, \text{якщо фрагмент F находиться у вузлі } K_j;\\ 0, \text{в інакшому випадку.}\end{cases} \quad (1)$$

Оскільки інтенсивність λ_{ij}^k породжує об'єм даних $\lambda_{ij}^k\left(\alpha_{ij}^k+\beta_{ij}^k\right)\left(1-x_{ij}\right)$, що потребують пересилки, то загальний об'єм даних, які необхідно переслати по каналам зв'язку між вузлами внаслідок функціонування розподіленої системи впродовж одиниці часу, визначається:

$$S=\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{s}\lambda_{ij}^k\left(\alpha_{ij}^k+\beta_{ij}^k\right)\left(1-x_{ij}\right). \quad (2)$$

Якщо покласти, що $\lambda=\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{s}\lambda_{ij}^k$, то цільова функція задачі оптимального розподілу фрагментів по вузлах ОМ буде мати вигляд:

$$V=\frac{1}{\lambda}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{s}\lambda_{ij}^k\left(\alpha_{ij}^k+\beta_{ij}^k\right)\left(1-x_{ij}\right). \quad (3)$$

Очевидно, чим менше значення середнього об'єму даних V, що пересилаються в одиницю часу, тим вище швидкість обслуговування запитів в системі. Усі повідомлення, що поступають у вхідні черги вузлів, розподіляються на два типи:

тип 1 – повідомлення, складові запити, для обробки яких необхідні фрагменти які не зберігаються в БД вузла, і відповіді на ці запити;

тип 2 – повідомлення, що становлять запити, для обслуговування яких потрібні фрагменти які зберігаються в БД відповідного вузла. При цьому вважатимемо, що запит типу 1, для свого обслуговування у віддалений вузол, перетворюється на запит типу 2.

Оскільки кожен фрагмент $F_i\left(i=\overline{1,m}\right)$ повинен знаходитися в одному з вузлів ОС, тоді

$$\sum_{j=1}^{n}x_{ij}\ge 1, i=\overline{1,m}. \quad (4)$$

Щоб наблизити модель до реальних систем, необхідно ввести коефіцієнт реплікації фрагментів RC. Цей параметр визначає кількість копій кожного фрагмента, розподілених по вузлах мережі. При цьому можливі два варіанти застосування цього коефіцієнта :

- коефіцієнт реплікації фрагментів RC визначає точну кількість копій кожного фрагмента (строга умова), тобто $\sum_{j=1}^{n}x_{ij}=RC, i=\overline{1,m}.$
- коефіцієнт реплікації фрагментів, який визначає максимальну кількість копій кожного фрагмента (нестрога умова), тобто $\sum_{j=1}^{n}x_{ij}\le RC, i=\overline{1,m}.$

Тоді обмеження по кількості реплік фрагментів виглядатиме таким чином:

для строгої умови: $1\le\sum_{j=1}^{n}x_{ij}=RC, i=\overline{1,m}.$

для нестрогої умови: $1\le\sum_{j=1}^{n}x_{ij}\le RC, i=\overline{1,m}.$

Крім того, об'єм локальної БД кожного вузла $K_j\left(j=\overline{1,n}\right)$ не повинен перевищувати об'єм пам'яті цього вузла, призначений для розміщення фрагментів. Тому

$$\sum_{i=1}^{m}L_i x_{ij}\le b_j, j=\overline{1,n}. \quad (5)$$

Таким чином, завдання оптимального розподілу фрагментів по вузлах ОМ полягає в тому, щоб визначити значення змінних x_{ij}, де $x_{ij}=\{0;1\}$ $\left(i=\overline{1,m};j=\overline{1,n}\right)$, які задовольняють умовам і дають

мінімум лінійної функції. Отримана математична модель є задачею цілочисельного лінійного програмування з булевими змінними. Сутність запропонованого способу полягає у наступному [13, 15]. В блоці сортування даних по відношенню значення коефіцієнтів функціонала до різниці між максимальним та мінімальним значенням ваги матриці обмежень 1 виконується сортування:

$$\Psi_j = c_j \Big/ \left(\max_i a_{ij} - \min_i a_{ij} \right), \qquad (6)$$

де $a_{ij} = a_{ij} / b_i$.

Обчислювальний пристрій 3 здійснює обчислення локальних екстремумів при заданому функціоналі та обмеженні, а також визначення (обчислення) номеру вершини, у якої локальний екстремум (ЛЕ) визначений за правилом

$$d_c\left(\mu_{sp}^{r}\right) + \gamma_p < \max_{\{c_j\}} \left\{ d_c\left(\mu_{sp}^{*r}\right) \right\}. \qquad (7)$$

З вершини *s* графа ΔD будується множина шляхів $m_{sj}^{r=1}$, $j = \overline{(1,n)}$ першого рангу r, що задовольняє властивості, і в множинах $m_{sj}^{r=1}$ визначаються шляхи максимальної довжини $\left\{\mu_{sj}^{*r}\right\}$ за вагою функціонала c_j. Для кожної вершини j визначається вага:

$$\gamma_j = c_{j+1} + c_{j+2} + \ldots + c_n, \quad \gamma_n = 0; \quad j = \overline{(1,n-1)}. \qquad (8)$$

Виключаються шляхи $\left\{\mu_{sp}^{r}\right\}$, $p = \overline{(r,n)}$ у множині m_{sj}^{r} поточного рангу r, довжини якої $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівності

$$\sum_{j=1}^{n} a_{ij} x_j \le b_1, \qquad (9)$$

$$x_j \in \{0,1\}, \quad i = 1, \quad j = \overline{(1,n)}, \quad a_{1j} > 0, \; c_j > 0. \qquad (10)$$

Формується множина шляхів $m_{sp}^{r=r+1}$, $p = \overline{(1,n)}$ наступного рангу, що задовольняє властивості, на базі множини шляхів m_{sj}^{r} попереднього рангу на основі правила відсікання неперспективних варіантів рішень по вибору мінімального значення довжини шляху в графі за вагою обмеження на основі принципу оптимізації за напрямком

$$\mu_{sp}^{r=r+1} = \min_{\{\alpha_j\}} \{ \mu_{sj}^{r} \cup (j,p) \} \quad p = \overline{r+1,n}, \; j = \overline{r,n}, \; j \ne p \; . (11)$$

У визначених множинах $m_{sp}^{r=r+1}$ виділяються щонайдовші шляхи $\left\{\mu_{sp}^{*r=r+1}\right\}$. Якщо визначиться декілька шляхів мінімальної довжини за вагою обмеження, то серед них вибирається шлях з найбільшим значенням довжини за вагою функціонала c_j.

Перевіряється, чи вся множина шляхів наступного (r+1)-го рангу порожня. Якщо умова виконується, то в множинах виділяється шлях максимальної довжини за вагою функціонала і алгоритм закінчує роботу. Якщо умова не виконується, то перевіряється r = (n – 1). У разі виконання рівності в множині виділяється шлях максимальної довжини за вагою функціонала і алгоритм закінчує роботу, інакше r збільшується на 1 і виконується обчислення.

Кожен процесорний елемент 4 обчислювального пристрою 3 виконує обчислення паралельно та здійснює обмін даними між сусідніми процесорними елементами після завершення обчислень. Блок регістрів 5 кожного процесорного елементу 4 зберігає і забезпечує мікрооперації передачі даних між регістрами блока регістрів сусідніх процесорних елементів. Арифметичний обчислювач 6 обчислює локальні екстремуми на підставі даних, що надходять з блока регістрів, вибирає локальний екстремум за правилом (10) і пересилає його в обчислювальний пристрій формування вектора шляху 8 для обчислення глобального екстремуму та формування вектора шляху. Блок ідентифікації 7 визначає номер вершини, у якій локальний екстремум визначений (рис. 6).

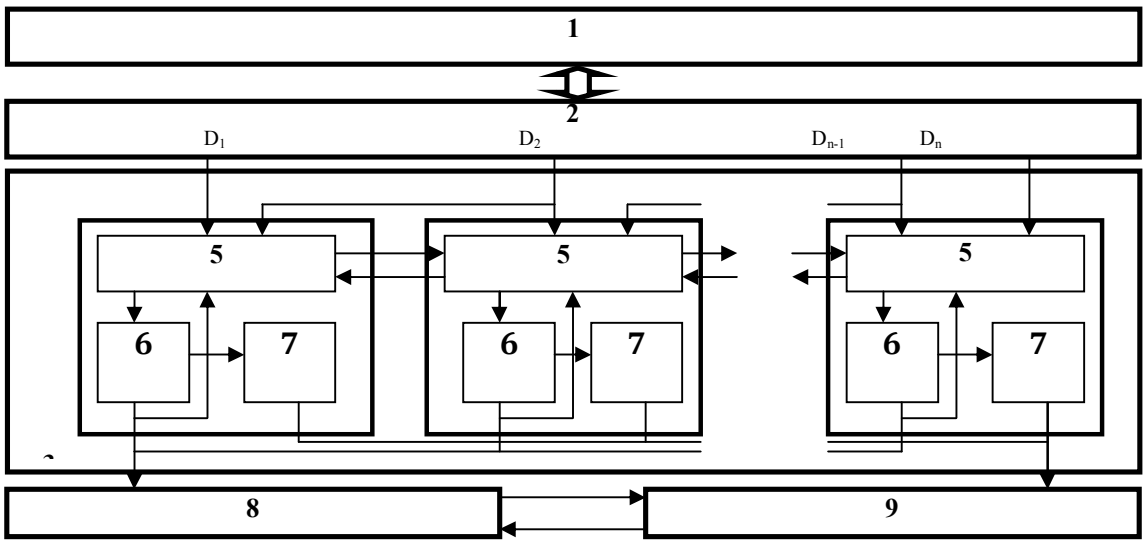

Рис. 6. Пристрій для рішення задач на графах

Модуль пам'яті 9 зберігає номери вершин локальних екстремумів на кожному рангу обчислень. Дані $D_1, D_2, ...D_n$ надходять одночасно в кожну систолічну комірку обчислювального пристрою, в яких здійснюється обчислення. Введення даних здійснюється за допомогою блока управління систолічним процесом 2 із блока сортування даних по відношенню значення коефіцієнтів функціонала до різниці між максимальним та мінімальним значенням ваги матриці обмежень 1.

Висновки

В ході проведення досліджень виконано аналітичний огляд технологій зберігання і обробки великих даних, аналіз методів інтеграції додатків на рівні даних, а також показано підхід до оптимізації структури сховища даних у вузлах інфокумунікаційної мережі хмарного середовища.

Список літератури

1. *Buyya R. et al. Big Data Analytics-Enhanced Cloud Computing: Challenges, Architectural Elements, and Future Directions //Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st Int. Conference on. - IEEE, 2015. - С. 75-84.*
2. *Dean J., Ghemawat S. MapReduce: simplified data processing on large clusters //Communications of the ACM. - 2008. - Т. 51. - №. 1. - С. 107-113.*
3. *Deelman E., Chervenak A. Data management challenges of data-intensive scientific workflows //Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on. - IEEE, 2008. - С. 687-692.*
4. *Hadoop Cluster Setup [Электронный ресурс]: руководство пользователя. - Режим доступа: http://hadoop.apache.org/docs/r1.2.1/cluster_setup.html. - Загл. с экрана (дата обращения:30.05.2017. [Electronic resource].*
5. *Phillips P. J. et al. Overview of the face recognition grand challenge //Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on. - IEEE, 2005. - Т. 1. - С. 947-954.*
6. *Ranganathan K., Foster I. Decoupling computation and data scheduling in distributed data-intensive applications // High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on. - IEEE, 2002. - С. 352-358.*
7. *Szabo C. et al. Science in the cloud: Allocation and execution of data-intensive scientific workflows //Journal of Grid Computing. - 2013. - С. 1-20.*
8. *Tian Y. et al. From "think like a vertex" to "think like a graph" //Proceedings of the VLDB Endowment. - 2013. - Т. 7. - №. 3.*
9. *Wieczorek M., Prodan R., Fahringer T. Scheduling of scientific workflows in the ASKALON grid environment //ACM SIGMOD Record. - 2005. - Т. 34. - №. 3. - С. 56-62.*
10. *Афанасьев А. П. и др. Программный комплекс для решения задач дискретной оптимизации на распределенных вычислительных системах //Труды Института системного анализа Российской академии наук. - 2006. - Т. 25. - С. 5-17.*
11. *Ильин В. А. и др. Способ запуска и обработки в гриде заданий, подготовленных для различных сред исполнения //Вычислительные методы и программирование. - 2008. - Т. 9. - №. 2*
12. *Кореньков В. В., Кутовский Н. А., Семенов Р. Н. Опыт адаптации прикладных программных пакетов для работы в грид-средах //Компьютерные исследования и моделирование. ISSN. - 2012. - С. 2076-7633.*
13. *Місюра О.М., Третяк В.Ф., Быльчук В.М. Метод оптимізації структури розподіленої бази даних у вузлах мережі хмарного середовища. /Наука і техніка Повітряних Сил Збройних Сил України, №1 (26). - 2017. - С. 92-96*
14. *Патент на корисну модель № 92968, Україна, МПК G06 F15/00. Спосіб обробки та захисту інформації в розподілених сховищах даних / В.Ф. Третяк, В.В. Бараннік та ін. – № u201403994; заяв. 14.04.2014; опубл. 10.09.2014; Бюл. № 17. – 5 с.*
15. *Третяк В.Ф., Корнієнко А.А. Метод оптимізації структури розподіленої бази даних у вузлах мережі хмарного середовища. Наука. Економіка. Інновації / Матеріали Міжнародної науково-практичної конференції, Чернівці, 15-16 січня 2017 р. - Т. 1. - Київ: Науково-видавничий центр «Лабораторія думки», 2017. - С. 7-9.*

Надійшла до редколегії 16.05.2017

Рецензент: д-р техн. наук, проф. В.І. Бараннік, Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків.

ОПТИМИЗАЦИЯ СТРУКТУРЫ ХРАНИЛИЩА ДАННЫХ В УЗЛАХ ИНФОКУМУНИКАЦИОННОЙ СЕТИ ОБЛАЧНОЙ СРЕДЫ

В.Ф. Третьяк, А.А. Пашнева

Главной идеей статьи является анализ методов и технологии работы с большими данными, анализ методов интеграции приложений на уровне данных, а также показать подход к оптимизации структуры хранилища данных в узлах инфокумуникационной сети облачного среды. Особое внимание было уделено ранговому подходу к решению задачи оптимизации структуры хранилища данных в узлах сети облачного среды.

Ключевые слова: *большие данные, репликация, ранговый подход, хранилище данных, инфокоммуникационная сеть, интеграция, тиражирование данных, облачная среда, фрагментация.*

OPTIMIZATION OF REPOSITORIES UNITS INFORMATION AND COMMUNICATION NETWORK CLOUD ENVIRONMENTS

V.F. Tretiak, A.A. Pashnyeva

The main idea of the article is to analyze the methods and technologies for working with large data analysis methods of integration applications data and see approach to optimize the structure of data storage nodes in the network cloud environment. Particular attention was paid Ranked approach to solving the problem of optimizing the structure of data storage network nodes in the cloud environment.

Keywords: *big data replication rank approach, data storage, information and communication network, integration, replication of data cloud environment, fragmentation.*