МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри Є. О. Давиденко

підпис

«___» _____20____ р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ ТА АНАЛІЗ ВПЛИВУ РОЗПОДІЛЕНИХ СИСТЕМ НА ЕФЕКТИВНІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Спеціальність «Інженерія програмного забезпечення»

121 – КРМ.1 – 608м.21810808

Студент _____ Я. Р.Гагін

підпис

«__» _____ 20__ р.

Керівник канд. техн. наук, доцент _____ Г. В. Гобрань

підпис

«__» _____ 20__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри_____Є. О. Давиденко

«_____» _____ 20__ р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

Видано студенту групи 608м факультету комп'ютерних наук

Гагін Ярослав Русланович
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Дослідження та аналіз впливу розподілених систем на ефективність програмного забезпечення

Затверджена наказом по ЧНУ від «__» _____ 20____ р. № _____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Обґрунтування необхідності удосконалення існуючого об'єкта проєктування базується на комплексному аналізі сучасного стану проблеми в галузі обробки великих обсягів даних у розподілених інформаційно-обчислювальних системах. Основні проєктні рішення спрямовані на розробку модифікацій зважених архітектурних рішень, математичного забезпечення для інтелектуальних систем та розподілених даних у великих інформаційних інфраструктурах.

4. Перелік питань, що підлягають розробці _____

1. Аналіз сучасного стану та перспектив розвитку розподілених систем в інформаційних інфраструктурах. 2. Розробка методики визначення функціональної специфікації типової розподільної системи в інформаційних інфраструктурах. 3. Спеціалізовані обчислювальні структури для оптимізації розподілених реєстрацій і обробки даних по вузлах комп'ютерної мережі. 4. Практична реалізація методу оптимізації розподілених реєстрацій підключення розміщення та обробки даних у системах управління базами даних

5. Перелік графічних матеріалів
Презентація

Керівник роботи канд. техн. наук, доцент, Горбань Гліб Валентинович
_____
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Гагін Ярослав Русланович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Дослідження та аналіз впливу розподілених систем на ефективність програмного
забезпечення_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРМ	10.11.2023р.	11.11.2023р.	виконано
2.	Огляд літератури за темою роботи	11.11.2023р.	18.11.2023р.	виконано
3.	Складання календарного плану КРМ	18.11.2023р.	24.11.2023р.	виконано
4.	Аналіз предметної області	24.11.2023р.	28.11.2023р.	виконано
5.	Аналіз сучасного стану та перспектив розвитку розподілених систем в інформаційних інфраструктурах	28.11.2023р.	03.12.2023р.	виконано
6.	Розробка методики визначення функціональної специфікації типової розподільної системи в інформаційних інфраструктурах	03.12.2023р.	15.12.2023р.	виконано
7.	Спеціалізовані обчислювальні структури для оптимізації розподілених реєстрацій і обробки даних по вузлах комп'ютерної мережі	15.12.2023р.	07.01.2024р.	виконано
8.	Практична реалізація методу оптимізації розподілених реєстрацій підключення розміщення та обробки даних у системах управління базами даних	07.01.2024р.	15.01.2024р.	виконано
9.	Відгук керівника КРМ	16.01.2024р.	17.01.2024р.	виконано
10.	Оформлення КРМ та презентації	17.01.2024р	19.01.2024р	виконано
11.	Попередній захист	10.02.2024р	16.02.2024р	виконано
12.	Рецензування	16.02.2024р	17.02.2024р	виконано
13.	Завершення оформлення КРМ та презентації	17.02.2024р	18.02.2024р	виконано
14.	Захист кваліфікаційної роботи	27.02.2024р.	27.02.2024р.	виконано

Розробив студент _____Гагін Ярослав Русланович_____ _____
(*прізвище, ім'я, по батькові)* *(підпис)*
«__» _____ 20__ р.

Керівник роботи канд. техн. наук, доцент, Горбань Г. В._____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Дослідження та аналіз впливу розподілених систем на ефективність програмного забезпечення»

Студент 608м гр.: Гагін Ярослав Русланович

Керівник: канд. техн. наук, доцент, Горбань Г. В.

Актуальність роботи — незважаючи на досягнення у сфері систем обробки надвеликих інформаційних баз, залишаються проблеми, які потребують додаткових наукових досліджень – подальший розвиток способів обробки великих масивів даних за допомогою розподілених інформативних систем (РІС) та розробка нових засобів створення високопродуктивних структур мереж, спрямованих на роботу в системах БД (СБД), що мають можливість забезпечити системну масштабованість, безвідмовність та високу доступність інформації за збереження або незначного збільшення ціни. Забезпечення зростання продуктивності цих мереж за рахунок збільшення ефективності обробки даних в інформаційних базах пов’язане з вирішенням питань щодо раціонального розміщення файлів з даними розподілених БД (РБД) по мережевих вузлах, розпаралелювання алгоритмів управління інформацією та ін.

Об'єкт дослідження. Процес обробки розподілених даних в ІОС.

Предмет дослідження. Способи та засоби підвищення продуктивності при розподілені даних в ІОС.

Метою роботи є дослідження та аналіз впливу розподілених систем на ефективність програмного забезпечення та підвищення продуктивності розподіленої системи за рахунок обробки великого обсягу даних при раціональному розміщенні інформаційних файлів у комп’ютерній мережі, а також отримання апаратно-програмних засобів, орієнтованих на розпаралелювання операцій та призначених для обробки великих обсягів даних з огляду на розподіл реєстрацій даних.

Досягнення поставленої мети у роботі вирішувалися такі основні завдання:

1.Систематизувати уявлення знань про технології обробки великих масивів

даних у розподілених інформаційно-обчислювальних системах (ІОС) та провести їх аналіз.

2.Розробити модифікації зважених архітектурних рішень для підвищення продуктивності ІОС, ефективність яких залежить від технологій обробки даних системою в режимі реального та від часу відгуку в умовах розподілу даних.

3.Формування математичного забезпечення типової інтелектуальної системи розподілених даних в інформаційних інфраструктурах

Обґрунтування необхідності удосконалення існуючого об’єкта проєктування базується на комплексному аналізі сучасного стану проблеми в галузі обробки великих обсягів даних у розподілених інформаційно-обчислювальних системах.

Практично вирішені задачі вказують на необхідність нового підходу, оскільки існуючі рішення частково розв’язують проблеми, але не враховують високий ступінь розподіленості та великий обсяг даних.

Аналіз показав, що існуючі методи розміщення файлів в розподілених базах даних мають обмеження і не завжди ефективні для великих обсягів даних, тому потрібен новий підхід.

Світові тенденції підкреслюють постійне зростання вимог до продуктивності розподілених систем та необхідність вирішення завдань обробки великих обсягів даних, що підкреслює актуальність нового підходу.

Основні проєктні рішення спрямовані на розробку модифікацій зважених архітектурних рішень, математичного забезпечення для інтелектуальних систем та розподілених даних у великих інформаційних інфраструктурах.

Результати дослідження можуть бути застосовані для підвищення продуктивності та ефективності роботи розподілених інформаційних систем у різних галузях економіки та науки.

Робота складається з 111 сторінки, містить 27 рисунків, 2 таблиці, 64 використаних джерел посилання та 1 додаток.

Ключові слова: розподілені бази даних, системи керування базами даних, оптимізація розподілених реєстрацій, інформаційна інфраструктура, комп’ютерна мережа, безпека підключення.

ASTRACT

of the Master's Thesis

" Research and analysis of the impact of distributed systems on software efficiency"

Student of group 608m: Hahin Yaroslav Ruslanovych

Supervisor: Candidate of Technical Sciences, Associate Professor Horban G. V

The relevance of the work is that despite the achievements in the field of systems for processing large information bases, there are no problems that require additional scientific research and further development of methods for processing large amounts of data through additional divisions of information systems (DIS) the development of new technologies and the creation of highly productive structures, direct to work in database (DB) systems, which can ensure system scalability, security and high availability of information for savings or a slight increase in price. Ensuring increased productivity of these measures due to increased efficiency of data processing in information bases is associated with the highest levels of nutrition and rational placement of files with data from divisional databases (RDBs) along the network nodes, roses parallelization of information management algorithms and so on. Object of investigation. The process of processing separate data in iOS. Subject of investigation. Ways to increase productivity when distributing data in iOS.

The method of work is to study and analyze the flow of distributed systems on the effectiveness of software and increase the productivity of a distributed system for processing large-scale data with rational placement of information files on the computer These boundaries, as well as the removal of hardware and software features, are oriented towards paralleling operations and purposes for processing of great obligations of data with a look at the distribution of data registration. The main objectives of the robot were as follows:

1. Systematize the existing knowledge about technologies for processing large amounts of data from separate information and computing systems (IOS) and conduct their analysis.

2. Develop modifications to important architectural solutions to increase the productivity of iOS, the effectiveness of which lies in the technology of data processing by the system in real time and in the hour of processing the data in the minds.

3. Formation of mathematical support for a standard intelligent system for sharing data in information infrastructures.

The justification of the need to improve the existing design object is based on a comprehensive analysis of the current state of the problem in the field of processing large volumes of data in distributed information and computing systems. Practically solved problems indicate the need for a new approach, since existing solutions partially solve problems, but do not take into account the high degree of distribution and large volume of data. The analysis showed that the existing methods of placing files in distributed databases have limitations and are not always effective for large volumes of data, so a new approach is needed.

World trends emphasize the constant growth of requirements for the performance of distributed systems and the need to solve the tasks of processing large volumes of data, which emphasizes the relevance of a new approach. The main project solutions are aimed at the development of modifications of weighted architectural solutions, mathematical support for intelligent systems and distributed data in large information infrastructures. The results of the research can be applied to increase the productivity and efficiency of distributed information systems in various fields of economy and science.

The work consists of 111 pages, contains 27 figures, 2 tables, 64 reference sources and 1 appendix.

Keywords: distributed databases, database management systems, optimization of distributed registrations, information infrastructure, computer network, connection security.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – Бази даних;

БПЕ – Багатофункціональні процесорні елементи;

ЗЗП – Зовнішній запам’ятовуючий пристрій;

ЗП – Запам’ятовуючий пристрій;

ІОС – Інформаційно-обчислювальна система;

КП – Керуючий процесор;

МПМ – Модуль пошуку максимуму;

ОЗП – Оперативний запам’ятовуючий пристрій;

ОКМД – Архітектура з одиночним потоком команд та множинним потоком даних;

ОС – Обчислювальна система;

ПАК – Програмований активний комутатор;

ПЕ – Процесорний елемент;

ПЗ – Програмне забезпечення;

ПЗП – Постійний запам’ятовуючий пристрій;

ПК – Пристрій керування;

ПЛІС – Програмована логічна інтегральна схема;

ППР – Послідовно-подібний розклад;

ПРА – Процес інтерпретації операторів реляційної алгебри;

ПС – Прискорювач серверів;

РБД – Розподілені бази даних;

РІС – Розподілені інформаційні системи;

РР – Розподілений розклад;

СБД – Системи баз даних;

СПО – Селектор провідної одиниці;

СКБД – Системи керування базами даних;

СКРБД – Система керування розподіленими базами даних;

СПЕ – Спеціалізовані процесорні елементи;

СРК – Схема порозрядної кон’юнкції;

СФО – Схема формування ознак;

CORBA – Common Object Request Broker Architecture, архітектура брокера запитів об'єктів;

DCOM – Distributed Component Object Model, розподілена компонентна об'єктна модель;

DII – Dynamic Invocation Interface, Динамічний інтерфейс виклику;

EJB – Enterprise JavaBeans;

FAP – Завдання розподілу файлів (File Allocation Problem);

FIFO – Дисципліна обслуговування запитів “Перший прийшов – перший пішов” (First input – first output);

IP – Internet-Протокол;

OMG – Object Management Group, Архітектура управління об'єктами;

OSI – Еталонна модель взаємодії відкритих систем (Open system interconnection);

SOAP – Simple Object Access Protocol;

UDDI – Universal Description, Discovery and Integration – Протокол пошуку вебсервісів в Internet;

UDP – Протокол передачі дейтаграм користувача (User datagram protocol);

URL – уніфікований покажчик ресурсів;

WSDL - Web Service Description Language.

ВСТУП

Актуальність роботи полягає у тому, що незважаючи на досягнення у сфері систем обробки надвеликих інформаційних баз, залишаються проблеми, які потребують додаткових наукових досліджень – подальший розвиток способів обробки великих масивів даних за допомогою розподілених інформативних систем (РІС) та розробка нових засобів створення високопродуктивних структур мереж, спрямованих на роботу в системах БД (СБД), що мають можливість забезпечити системну масштабованість, безвідмовність та високу доступність інформації за збереження або незначного збільшення ціни. Забезпечення зростання продуктивності цих мереж за рахунок збільшення ефективності обробки даних в інформаційних базах пов’язане з вирішенням питань щодо раціонального розміщення файлів з даними розподілених БД (РБД) по мережевих вузлах, розпаралелювання алгоритмів управління інформацією та ін.

Об’єкт дослідження – процес обробки розподільних даних в ІОС.

Предмет дослідження – способи та засоби підвищення продуктивності при розподілені даних в ІОС.

Метою роботи є дослідження та аналіз впливу розподілених систем на ефективність програмного забезпечення та підвищення продуктивності розподіленої системи за рахунок обробки великого обсягу даних при раціональному розміщенні інформаційних файлів у комп'ютерній мережі, а також отримання апаратно-програмних засобів, орієнтованих на розпаралелювання операцій та призначених для обробки великих обсягів даних з огляду на розподіл реєстрацій даних.

Досягнення поставленої мети у роботі вирішувалися такі основні **завдання**:

1. Систематизувати уявлення знань про технології обробки великих масивів даних у розподілених інформаційно-обчислювальних системах (ІОС) та провести їх аналіз.

2. Розробити модифікації зважених архітектурних рішень для підвищення продуктивності ІОС, ефективність яких залежить від технологій обробки даних

системою в режимі реального та від часу відгуку в умовах розподілу даних.

3. Формування математичного забезпечення типової інтелектуальної системи розподілених даних в інформаційних інфраструктурах.

Розвиток інформаційних та комунікаційних технологій призвів до поширення розподілених систем обробки даних на основі комп'ютерних мереж. Існують науково-технічні завдання обробки надвеликих масивів даних, реалізації яких недостатньо одного комп'ютера з однопроцесорною архітектурою. Прикладом надвеликої бази даних (БД) є БД системи спостереження Землі EOS/DIS [1], яка включає дані безлічі супутників, що збирають інформацію для вивчення довгострокових тенденцій стану атмосфери, океанів і земної поверхні з обсягом надходження інформації 1 Пбайт на рік. Подібною системою має і Стендфордський центр лінійного прискорювача, що має БД, створену для експерименту BABAR (вивчення зіткнення субатомних частинок з метою визначення впливу поведінки матерії та антиматерії на формування всесвіту) та, за даними [2], що має обсяг 1492.0 Тбайт.

Літературний огляд показав, що відомі способи та обчислювальні структури оптимізування розміщення інформаційних файлів розподілених баз даних по комп'ютерним мережевим вузлам ефективні для завдань малої розмірності, носять єдиний або приватний характер, не враховують властиві для комп'ютерних мереж особливості, або складні для реалізації.

У своїй формальній постановці задача оптимального розміщення файлів полягає в мінімізації цільової функції, яка визначає залежність рівномірних експлуатаційних витрат від розміщення файлів. Цільовою функцією є вартість зберігання копій файлів і передачі довідкових і коригувальних транзакцій через канали зв'язку. Використовується пошук рішення без урахування обмежень на продуктивність комп'ютера шляхом редукції нелінійної дискретної задачі програмування з булевими змінними. Вибір конфігурації мережі та розміщення файлів з урахуванням можливих обмежень підключення до мережі та наявності хоча б 1 копії кожного файлу здійснюється по частинах евристичними методами.

Виходячи з проведеного аналізу, можна зробити такі висновки. Критерії для

оцінки ефективності розподілу інформаційних ресурсів у комп’ютерній мережі слід вибрати з набору якості обслуговування еталонну модель BOS/MOS, розширену показниками, що характеризують вартість досягнення належного рівня якості. Модель оптимізації для задачі розміщення файлів RDB в комп’ютерній мережі з топологією спільної шини може бути використана у випадках, коли вихідні дані характеризуються великим ступенем невизначеності. Евристичні або точні алгоритми вирішення навіть простої задачі можуть дати необ’єктивні оцінки ефективності розподілу інформаційних ресурсів.

Таким чином, тематика даної магістерської роботи, присвяченої розробці засобів організації розміщення та забезпечення безпеки іформаційних файлів у розподіленій обчислювальній системі, є актуальною та є науковим та практичним інтересом.

Обґрунтування необхідності удосконалення існуючого об’єкта проєктування базується на комплексному аналізі сучасного стану проблеми в галузі обробки великих обсягів даних у розподілених інформаційно-обчислювальних системах (ІОС).

Практично вирішені задачі вказують на необхідність нового підходу, оскільки існуючі рішення частково розв'язують проблеми, але не враховують високий ступінь розподіленості та великий обсяг даних.

Аналіз показав, що існуючі методи розміщення файлів в розподілених базах даних мають обмеження і не завжди ефективні для великих обсягів даних, тому потрібен новий підхід.

Основні проєктні рішення спрямовані на розробку модифікацій зважених архітектурних рішень, математичного забезпечення для інтелектуальних систем та розподілених даних у великих інформаційних інфраструктурах.

Результати дослідження можуть бути застосовані для підвищення продуктивності та ефективності роботи розподілених інформаційних систем у різних галузях економіки та науки.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ТА ПЕРСПЕКТИВ РОЗВИТКУ РОЗПОДІЛЕНИХ СИСТЕМ В ІНФОРМАЦІЙНИХ ІНФРАСТРУКТУРАХ

1.1 Огляд методів і структур обробки даних у сучасних інформаційних системах

На сьогоднішній день реальні комп'ютерні мережі характеризуються деякими недоліками, особливо, розподілена мережа є дуже неоднорідним середовищем інформаційної передачі: одні ділянки можуть бути побудовані за методиками ATM або FDDI, інші – на основі повільних протоколів X.25. Реальна швидкість інформаційної передачі безпосередньо залежить від пропускної спроможності найбільш повільної мережі. Так, доступ віддаленого користувача до корпоративної бази може бути значно ускладнений. З іншого боку, чи завжди потрібний віддаленому користувачеві абсолютний доступ до всієї інформаційної бази? У багатьох випадках запитується лише інформація, яка безпосередньо належить до його середовищі діяльності.

Ефективність ІОС безпосередньо залежить від активності трафіку, чим він нижчий, тим швидше окупляться кошти, які були вкладені в її побудову. Для системної реалізації потрібна правильна координація розподілу та зберігання даних. Оптимальним методом зниження трафіку в зв’язкових каналах є застосування методики «клієнт-сервер» [7, 9, 11, 26, 27].

Проектування розподілених систем має багато схожості з проектуванням програмного забезпечення. У цьому випадку також необхідно враховувати специфічні властивості [7,8,11,14].

Існує 6 основних характеристик розподілених систем.

1. Спільне ресурсне застосування як апаратних (жорстких дисків, принтерів), так і програмних (файлів, трансляторів);
2. Відкритість;
3. Паралельність;
4. Масштабованість;
5. Безвідмовність;

6. Прозорість.

Недоліки систем розподілу.

1. Складність.

2. Безпека.

При проектуванні систем розподілу існує ряд питань, які проектувальники повинні враховувати.

1. Ідентифікація ресурсу.

2. спілкування.

3. Якість обслуговування системи.

4. Архітектура програмного забезпечення. Завданням розробників системи розподілу є розробка програмного та апаратного забезпечення для забезпечення бажаних характеристик розподіленої системи. Існує 3 типи архітектур системи розподілу.

1. Клієнт – сервер. Систему можна представити як набір послуг, які сервер надає клієнтам. У цих системах сервери та клієнти відрізняються один від одного.

2. Триланкова. Сервер не надає послуги клієнтам безпосередньо, а через бізнес-логіку сервера.

3. Архітектура розподілених об'єктів. Сервери та клієнти нічим не відрізняються, і систему можна розглядати як набір взаємодіючих об’єктів, розташування яких не має значення.

Сьогодні архітектура розподілених об’єктів широко використовується і називається архітектурою веб-сервісів. Веб-сервіс – це додаток, доступний через Інтернет, що надає послуги, тип яких не залежить від провайдера (оскільки використовується універсальний формат інформації – XML) і функціональних платформ. Існує 3 відомі техніки, які підтримують концепцію розподілених об’єктних систем: методи EJB, CORBA та DCOM.

«Клієнт – сервер» є ключовою моделлю взаємодії між машинами в мережі при роботі в сучасних ІВС. Цей принцип також стосується взаємодії програмного забезпечення. Якщо один з них виконує певні функції, надаючи іншим необхідний

спектр послуг, то така програма вважається сервером. Програми, які використовують ці служби, називаються клієнтами. Таким чином, ядро реляційної SQL-орієнтованої системи управління базами даних називається сервером бази даних або SQL-сервером, а програма, яка звертається до нього для служб обробки інформації, називається SQL-клієнтом.

Відмінності в реалізаціях методики "клієнт-сервер" визначаються 4-ма факторами. Перший, це які види програмного забезпечення інтегрований кожен із цих елементів. Другий, які механізми програмного забезпечення застосовуються для реалізації функцій цих 4-х груп. Третій, як логічні елементи розподіляються між машинами у мережі. Четверте, які механізми застосовуються для зв'язку елементів між собою.

Модель сервера файлів (File Server (FS) – FS – модель) вважається основним для локальних мереж ПК. Відповідно до даної моделі 1 - н з мережевих комп'ютерів є сервером файлів і надає послуги з файлової обробки іншим комп'ютерам. Сервер файлів працює під управлінням операційної системи мережі (наприклад, Novell NetWare). На інших мережевих комп'ютерах працює додаток, коди якого з'єднані уявлення і елементи. Протокол обміну є комплектом низькорівневих викликів, які забезпечують доступ додатку до системи файлів на файловому сервері.

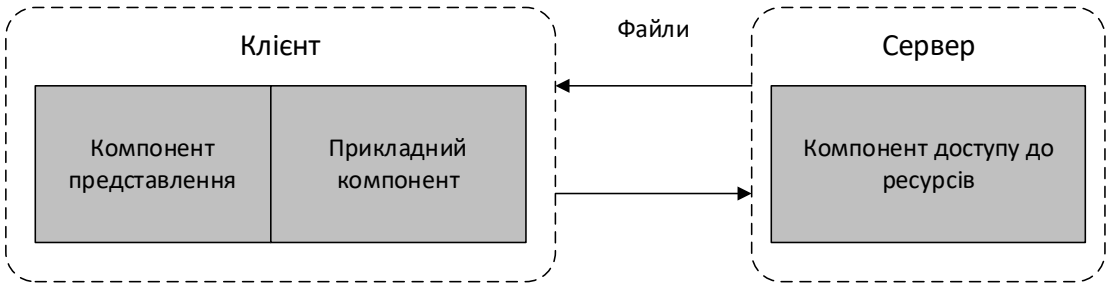


Рисунок 1.1 – Обчислювальна система розподілу з урахуванням файлового сервера

FS – модель є фундаментом для розширення можливостей СКБД у сфері підтримки розрахованого на багато користувачів режиму. У цих системах на кількох ПК виконується прикладна програма і копія СКБД, а бази даних містяться

в файлах, що розділяються, що знаходяться на файл - сервері. При зверненні прикладної програми до інформаційної бази, СКБД відправляє запит файл – сервер.

Технологічними недоліками моделі вважають високий мережевий трафік, вузький спектр операцій маніпулювання інформацією («інформація — це файли»), відсутність необхідних засобів безпечного доступу до даних (захист лише на рівні файлової системи) тощо.

Найбільш технологічна модель RDA суттєво відрізняється від моделі FS характером елемента доступу до інформаційних ресурсів. Зазвичай це SQL-сервер. У моделі RDA коди елемента презентації та елемента додатка пов’язані та виконуються на клієнтському комп’ютері. Доступ до інформаційних ресурсів забезпечується або операторами за допомогою спеціальної мови (мова SQL, якщо мова йде про базу даних), або функціями спеціальної бібліотеки (за наявності відповідного інтерфейсу прикладного програмування - API).

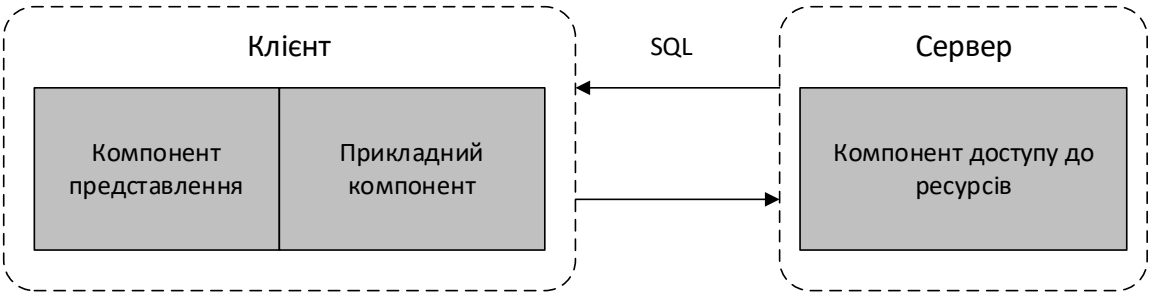


Рисунок 1.2 - Обчислювальна система розподілу з доступом до віддаленої інформації

Клієнт надсилає запити до інформаційних ресурсів (наприклад, бази даних) по мережі на віддалений ПК. На ньому працює ядро системи управління базою даних, яке обробляє запити, виконує вказані в них дії та відправляє клієнту результат у вигляді інформаційного блоку..

RDA– модель усуває недоліки, які притаманні як системам з централізованою архітектурою, так і системам з файловим сервером.

Перенесення елемента презентації та елемента програми на клієнтські комп'ютери значно розвантажує сервер інформаційної бази та мінімізує загальну

кількість процесів операційної системи.

Головною перевагою моделі RDA є уніфікація клієнт-серверного інтерфейсу у вигляді мови SQL. RDA – модель також має недоліки. По-перше, взаємодія клієнт-сервер через SQL-запити значно навантажує мережу. По-друге, задовільне управління додатками в моделі RDA практично неможливо через поєднання різних за своєю суттю функцій (функцій представлення та додатків) в одній програмі.

Переваги DBS – моделі: можливість централізованого управління прикладними функціями та зниження трафіку (замість SQL – запитів по мережі відправляються виклики процесів, що зберігаються), можливість поділу процесу між декількома додатками, економія комп'ютерних ресурсів за рахунок застосування одного разу сформованого плану виконання процесу. До мінусів можна віднести обмеженість у засобах, що використовуються для написання збережених процесів, що представляють собою різні розширення процесів SQL, що не витримують зіставлення за функціональними можливостями з мовами 3-го покоління, таким як С.

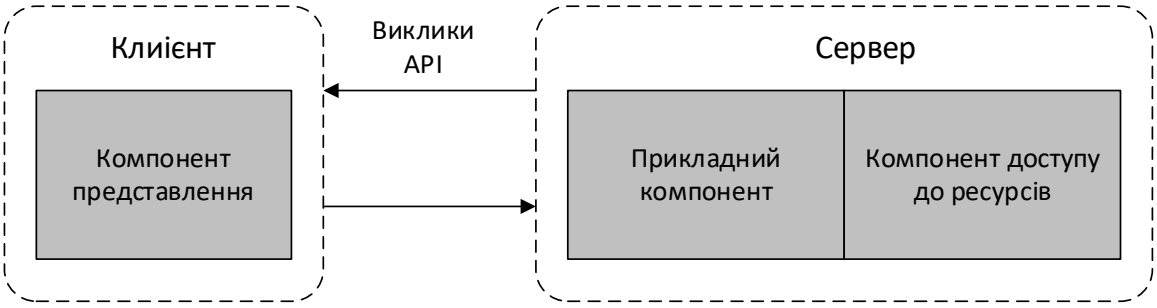


Рисунок 1.3 - Обчислювальна система розподілу та сервер інформаційних баз

У практиці застосовуються змішані моделі, коли підтримка цілісності інформаційної бази та деякі прості прикладні функції виконуються процесами, що зберігаються (DBS – модель), а найбільш складні функції здійснюються саме в прикладній програмі, що працює на клієнт – комп'ютері (RDA – модель).

У AS – моделі процедура, що виконується клієнт – комп'ютері, відповідає, переважно, за інтерфейс з користувачем (тобто. реалізує функції 1 – й групи).

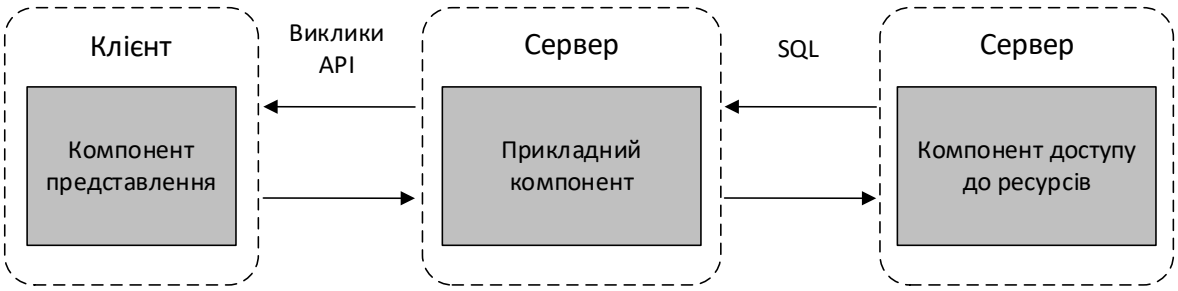


Рисунок 1.4 - Обчислювальна система розподілу та сервери додатків

Моделі RDA та DBS будуються на дволанковій схемі поділу функцій.

Сучасним рішенням проблеми СКБД для багатопроцесорних платформ є можливість запуску декількох серверів інформаційних баз, у тому числі на різних процесорах. Кожен сервер повинен бути багатопоточним. (багатопотокова інфраструктура з кількома серверами) (багатопотокова, багатосерверна архітектура).

1.2 Координація розподіленої обробки інформації

Сучасний рівень прийняття рішень та оперативного управління інформаційними ресурсами вимагає більшої децентралізації. Для вирішення проблеми використовуються наступні методи: методика розподіленої бази даних (DDB) і методика реплікації даних.

В основу взаємодії прикладних програм клієнтів з сервером інформаційних баз покладено кілька фундаментальних принципів, які визначають функціональні можливості сучасних систем управління БД у частині, що відноситься до мережевої взаємодії та розподіленої інформаційної обробки, серед яких:

- прозорість розташування;
- мережева прозорість;
- автоматичне перетворення інформаційних форматів;
- автоматична кодова трансляція;
- міжопераційність.

Під час встановлення елементів DBMS Client Net на локальних пристроях виконується процес ідентифікації вузлів. Якщо базу даних перенесено на інший

пристрій, тоді не потрібно вносити зміни до прикладної програми, а достатньо поставити ім'я нового вузла.

Глобальний інформаційний словник відстежує розташування об'єктів у розподіленій базі даних. Інформація може зберігатися на локальному пристрої, на віддаленому вузлі або на обох вузлах – їх розташування має бути чітким для кінцевого користувача та програм. Не обов'язково точно вказувати розташування даних – програма цілком може залежати від цього, яких вузлах розміщені відомості, із якими вона проводить операцію.

Друге завдання вирішується інтелектуально. Розподілений запит стосується декількох інформаційних баз на різних вузлах, при цьому обсяги вибірки можуть бути різними. Можливі ситуації, коли підсумкова таблиця запиту є об'єднання (join) 2 – х таблиць і з них перебуває на локальному пристрої, а друга – на віддаленому. Цей запит – розподілений, т.к. торкається таблиці, що належать різним інформаційним базам. Для його виконання потрібно мати дві вихідні таблиці на 1-му вузлі. Таким чином, одну з таблиць необхідно передати через мережу. Очевидно, що ця таблиця має бути меншою за розміром. Отже, оптимізатор розподілених запитів повинен враховувати табличні розміри, інакше запит виконуватиметься дуже довго.

Крім табличних розмірів, оптимізатор розподілених запитів зобов'язаний враховувати також багато додаткових параметрів. Усі дані перебувають у глобальному інформаційному словнику.

Рішення всіх 3-х завдань, покладено спеціальний елемент систем управління базами даних – сервер РБД (Distributed Database Server). Якщо база даних знаходиться на одному вузлі, а сервер та прикладна програма також виконуються там, тоді не потрібен ні комунікаційний сервер, ні сервер розподіленої інформаційної бази.

Принципова відмінність методики реплікування інформації від методики РБД (яку найчастіше для стислості називають технологією STAR) полягає у відмові від розподілених даних. Суть полягає в тому, що будь-яка база даних (для СКБД та користувачів, які з нею працюють) завжди вважається локальною; інформація

завжди розміщується локально тому мережному вузлі, де вони обробляються; усі системні транзакції завершуються локально.

Особливість методики РБД – одночасне завершення транзакцій кількох пристроях розподіленої системи, тобто. синхронне фіксування змін РБД.

Реальною альтернативою методиці STAR є методика тиражування інформації, яка не вимагає синхронного запису змін.

1.3 Використання CASE – інструментів для моделювання даних в сучасних системах обробки інформації

Створення ІОС підприємства – нелегка та багатоступінчаста процедура, що містить фазу моделювання даних. Модель даних – специфікація інформаційної структури та порядків предметної галузі. Для будівництва інформаційної моделі підприємства використовують так звані Computer Aided Software Engineering (CASE) - гроші.

CASE– методика автоматизованого проектування ІВС, яка дозволяє значно прискорити процедуру їх розробки, зменшити трудовитрати та підвищити якість проектування.

1.4 Чисельні аспекти та алгоритми обробки інформації в комп'ютерній мережі

У вимогі до РБД можна назвати управління функціонуванням БД (одноразовий доступ, захист, відновлення), які визначаються СКБД, і доступом до інформації (запити).

1.4.1 Одноразовий доступ

Особливістю РБД є багатокористувацький розподілений доступ до інформації.

Розглянемо проблему конкурентного паралельного доступу.

Взаємодія компонентів у RDB здійснюється через транзакцію, яка стає розподіленою (рис. 1.5) і, діючи на всю базу даних, залишає базу даних незайманою

після завершення. Транзакція містить команди: read R і write – update W.

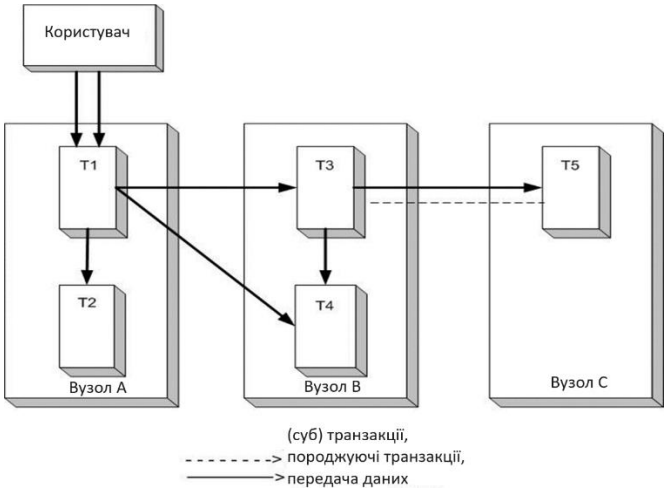


Рисунок 1.5 - Модель розподілених транзакцій: Ti (під) транзакції

У розподілених базах даних використовуються операції, синхронізації, розклади, послідовно подібні до розкладу (ППР), конфлікту, рестарту. Дві транзакції входять у конфлікт, у разі роботи з одним і тим самим єдиним інформаційним компонентом і щонайменше одна з них здійсниться операцією «писати» запис.

Дії щодо інформаційної зміни допускаються лише за попередньо заблокованими даними, при цьому спочатку має бути виконане оновлення всіх копій. У цьому режимі можливі безвиході, усунути які можна зняттям «молодших» (що надійшли пізніше) транзакцій.

1. Кожен кортеж t відносини r зі схемою R, у якому предикат P(t) = true, неможливо вставити R, видалений чи змінений інший транзакцією до зняття блокування.

Двофазне блокування виконується тут таким чином:

У стадії розширення (підготовки) транзакція має запросити предикат – блокування. Вузол – координатор веде блокувальну таблицю. Блокування виконується, якщо воно не входить у конфліктні ситуації з іншими блокуваннями. В іншому випадку - транзакція перенаправляється в режим очікування або їй дозволяється виконати перехоплення ресурсу.

У стадії стиснення (розблокування) виконується видалення рядка з блокувальної таблиці: перше ж розблокування повідомляє початок стадії стиснення.

Встановлення факту конфліктної ситуації тут, як і раніше, проблемно: показано, що загалом проблема рекурсивно не вирішувана.

Процес встановлення факту конфліктної ситуації визначено математично для предикатів типу [атрибут] F [константа], де як F застосовуються операції з множини($<, =, >, \neq$).

Тупикове положення може бути усунуто за допомогою трифазної транзакції (блокування, виконання, розблокування).

Блокування з основним вузлом та із застосуванням предикатів зводяться до централізації процесу синхронізації. У такому разі продуктивність інформаційної бази обмежена можливостями нестандартного центру, подальший розвиток синхронізації пішов у напрямку децентралізації. Для будь-якого інформаційного компонента одна копія є основною, яку блокує транзакція у кожному вузлі.

1.4.2 Захист інформації та застосунків у базах даних

Як і в централізованій базі даних, виділяються режими заборони та дозволу. Заборони – це пароль, ідентифікатор користувача. Вони дають можливість отримати доступ до розподілених баз даних загалом. Окремі частини розподілених баз даних може мати власний захист. Для цього АРБД встановлює повноваження (GRANT): список операцій, які користувач зможе виконувати із відповідними інформаційними компонентами.

Повноваження можна віднести до об'єктів чи системі загалом. Повноваження для певних об'єктів («начинка таблиці») надають конкретним користувачам можливість проведення операцій (INSERT, UPDATE, DELETE). Повноваження системи стосуються всієї інформаційної бази: створення, структурна зміна, видалення будь-якої області таблиці, опитування будь-якої таблиці (CREATE, ALTER, DROP, SELECT). Як і в головній БД, наведені повноваження можна скасувати (REVOKE). Для захисту АРБД може формувати (використовувати) ролі.

1.4.3 Спосіб відновлення РБД

Якщо розподілена база даних забезпечує цілісну, несуперечливу інформацію, це говорить про її коректність. Відновлення (управління відновленням) пов'язане з приведенням системи у стан коректності після (апаратного) збою.

Відновлювальна система вирішує 2 групи завдань:

1) при невеликій несправності – відмова у проведенні поточної транзакції;
2) при значній відмові – мінімізування відновлювальної роботи РБД.

Для відновлення найчастіше застосовується системний реєстраційний журнал.

Копіювання змін дозволено після закриття інформаційної бази.

У журналі може бути одна або більше груп реєстраційних файлів і членів груп для збереження змін РБД. Кожна група включає у собі 1 – н чи більше файлів, які можуть зберігатися різних дисках, як це відбувається в СКБД Oracle [24, 29, 31].

Процедура відновлення пов'язана з процесом одноразового доступу (паралельного виконання), що організовується двофазною транзакцією.

Перевагою методу головної копії вважається єдина послідовність коригування бази даних та зменшення ймовірності тупикової ситуації. У разі читання у доступних вузлах та застосування активних вузлів створюється та вводиться перелік U записів про вузли, які активні, а ключовий вузол використовується для відновлення. «Головна» транзакція під час оновлення зобов'язана вимагати обстановку для запису та корекції інформації у всіх вузлах переліку U, включаючи ключовий. Вдається уникнути деяких конфліктних ситуацій (читання – запис, запис – запис), тоді, коли система цілком працездатна. За наявності відмов та відновлення перелік U змінюється головним вузлом і можуть виникнути ускладнення.

Відновлювальна схема одного з вузлів представлена рисунку 1.6. Нехай у період t1 відмовив вузол два та його база даних зруйнувалася. Якщо вузол два не буде виправлено до періоду часу t5, тоді відмова вузла один чи два призведе до ґрунтовного збою. Вузол два виявляє свою відмову на період t2 і починає

відновлюватися, застосовуючи «знімки» інших вузлів. На інтервалі t3 – t4 робочі вузли мають продовжувати проведення транзакцій, які має запам'ятовувати вузол два (і обробляти після періоду t4). З періоду t4 вузол два примикає до інших, і система здатна витримати 2-й відмова.

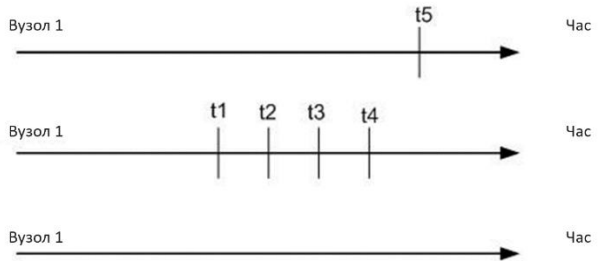


Рисунок 1.6 - Відновлювальна схема при втраті управління: t1, t5 - відмова; t2 – виявлення (само) відмови; t3 – початок відновлення; t4 - закінчення відновлення

1.4.4 Запити

Якщо структура запитів попередньо відома (стандартна), ефективність процедури запиту встановлюється на описаних раніше стадіях фрагментації і локалізації. Якщо структура запитів невідома, виникає потреба оптимізувати процедуру запитів. За наявності паралельної інформаційної обробки це актуально.

Цілеспрямованими функціями оптимізування можуть бути:

1) мінімум часу інформаційної передачі;
2) мінімум часу інформаційної обробки у вузлах;
3) збільшення паралельності під час обробки та передачі даних;
4) покращення трафіку та процесорного завантаження в мережі;
5) менше вартості (затримки) лише інформаційної передачі (якщо є низькошвидкісні зв'язкові канали);
6) менше вартості локальної обробки (якщо швидкість інформаційної передачі можна порівняти з процесорною швидкодією);
7) вирівнювання комп'ютерного навантаження.

Для характеристики витрат на запити введемо 2 параметри: ціна обробки та затримку інформації.

Ціна ТЗ інформаційної обробки об'єму x, яка пересилається у вузол i з вузлів j та пов'язаної з експлуатацією фізичних зв'язкових каналів, становить

$$TC = \sum_{i=1}^{j} C_0^{ij} + C_1^{ij} x,$$

деC_0^{ij}, C_1^{ij}– зумовлені системою матриці ціни встановлення зв'язку та одиниці інформаційної передачі($C_0^{ij} = C_1^{ij} = 0$).

Затримка TD(x) – час між початком і завершенням обробки запита:

$$TD(x) = \sum_{i=1}^{j} D_0^{ij} + D_1^{ij} x,$$

деD_0^{ij}, D_1^{ij}, - матриці часу встановлення зв'язку та одиниці інформаційної передачі.

Отримаємо припущення:

1) канали зв'язків однорідні($C_0^{ij} = C_0,\ C_1^{ij} = C_1,\ D_0^{ij} = D_0,\ D_1^{ij} = D_1$);

2) Вартість передачі висока і Витрати локальну обробку можна проігнорувати (враховуються параметри лише інформаційної передачі).

Оптимізація запиту може бути поділена на 2 етапи:

1) глобальне оптимізування, яке далі називатимемо оптимізацією;

2) локальне оптимізування [18], проведене методами діагонитики цифрових пристроїв.

Для опису суті оптимізування застосовують апарат реляційної алгебри (РА) з її перетвореннями еквівалента, які дозволяють зменшити час відповіді на запит. Змінними оптимізування можуть бути вибір фізичних копій, і навіть призначення вузлів під час проведення операцій РА.

Використовуємо дерево запитів, де всім листям відповідає відношення, кожній вершині – операція реляційної алгебри. Кінцевій вершині відповідає вузол розподіленої бази даних (локальна БД), а відповіді запит – коренева вершина.

Можливе серйозне вирішення задачі оптимізування, що найчастіше

зводиться до завдання цілісного програмування.

Застосуємо евристичний підхід до оптимізування. Ціну та час інформаційного пересилання можна скорочувати за рахунок зменшення обсягу даних, що передаються.

У зв'язку з цим можливе таке [2,26]:

- унарні операції слід виконати якомога раніше (у вузлах) (рисунок 1.7);

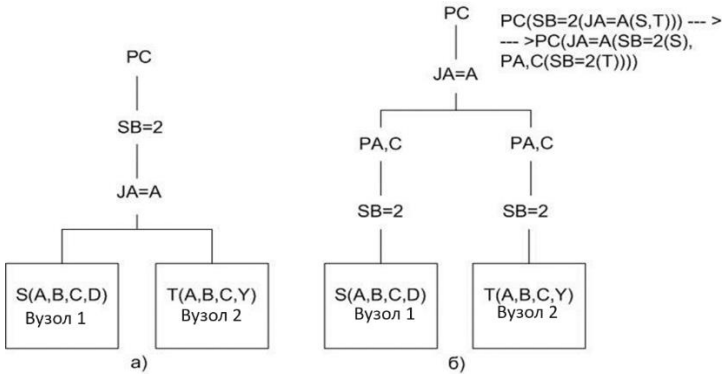


Рисунок 1.7 – Застосування унарних операцій

- вирази, що часто зустрічаються, слід виконати 1 раз. І тому у дереві запиту виробляється об'єднання листя – відношень, далі – злиття ідентичних проміжних операцій (рисунок 1.8). Тут застосовується вираз$DF(T, S_{NF}(T)) = S_F(T), NF = NOT\ F$.

- застосування напівз'єднання. Концепція напівз'єднання представлена рисунку 1.9. При традиційній схемі або R потрібно пересилати у вузол два або S - у вузол один (пунктирна лінія). Можна пересилати тільки 1 разів S, скорочену за допомогою проекції, один вузол і далі виконувати напівз'єднання;

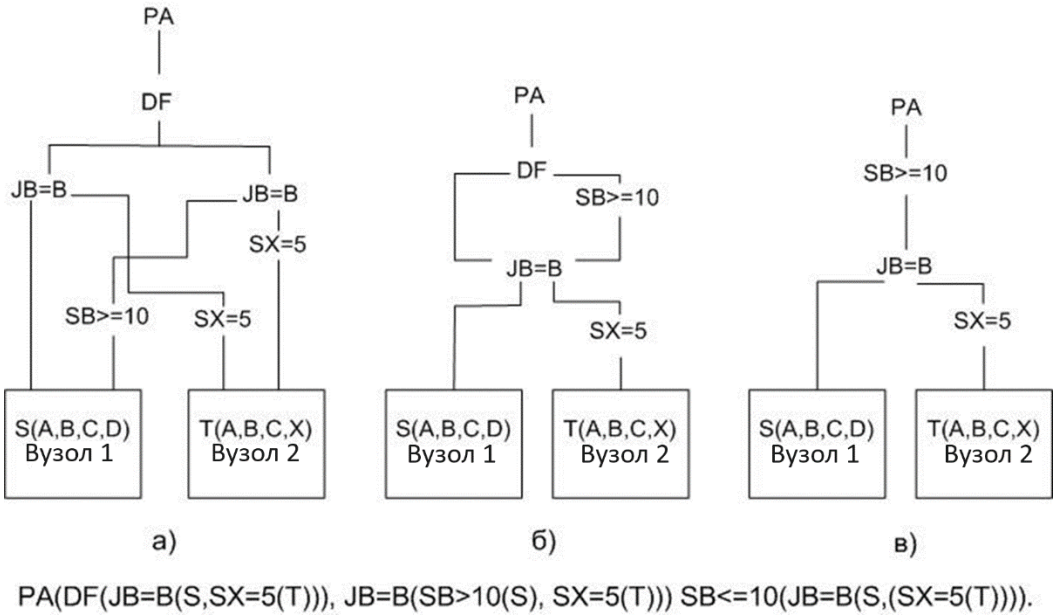


Рисунок 1.8 – Виділення єдиних виразів: а – вихідний запит; в – перетворений

– стратегія використання копій (доступу). Вона реалізується фізично. Встановити найкращий варіант за їх значної кількості можна лише з допомогою моделювання. Нехай будуть враховані лише витрати інформаційної передачі та є матриці ціни чи часу передач. Значення матриці можна встановити точно, а розміри відношень приблизно. Таким чином, слід використовувати наближені способи.

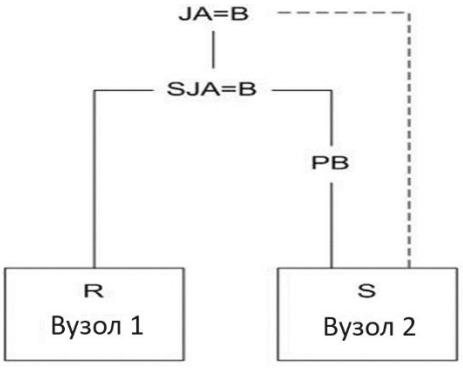


Рисунок 1.9 - Застосування напівз'єднання

Щоб вибрати копії та призначення виконавчих вузлів, доречно будувати граф оптимізації, що не містить унарних операцій, які виконуються у вузлах. Операції декартового добутку та різниці дуже рідко зустрічаються, найчастіше спостерігаються операції об'єднання та з'єднання.

Завдання оптимізації запиту відрізняється спірністю рішення, воно має бути регулярним та оперативним. Від розробника потрібна побудова спеціалізованих програм. Систематизовано термінологію, дуже багато уваги приділено сучасним тенденціям розвитку методики БД. Сюди відносяться автоматизування проєктування баз даних із застосуванням об'єктно - орієнтованого методу створення додатків, інтерфейсів користувача і вибудовування самих БД [2,26].

1.5 Критерії ефективності функціонування інформаційної інфраструктури

Найбільш часто використовувані аспекти продуктивності мережі можна розділити на дві групи. Одна група визначає продуктивність мережі, інша – надійність.

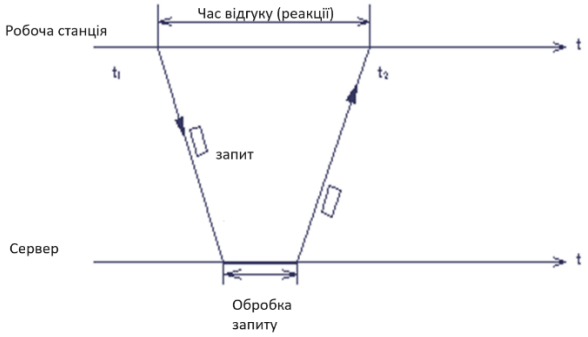
Продуктивність комп'ютерної мережі визначається за допомогою двох типів індикаторів - індикаторів часу, які оцінюють затримку, яку вносить мережа під час обміну інформацією, і індикаторів пропускної здатності, які відображають кількість даних, що передаються мережею в одиницю часу. Ці 2 види показників вважаються взаємно оберненими, і якщо відомий один з них, то можна розрахувати другий [3,26].

1.5.1 Час відгуку

Показник «час відгуку» використовується як тимчасова оцінка продуктивності комп'ютерної мережі.

Відповідно, час відповіді задається як часовий інтервал між виникненням запиту користувача на деякий мережевий сервіс і отриманням відповіді на цей запит (рисунок 1.10). Суть і значення цього показника залежить від типу сервісу, до якого здійснюється доступ, від того, який користувач і який конкретно сервер звертається, а також від поточного стану інших компонентів мережі - навантаження на сегменти, через які здійснюється запит, навантаження на сервер тощо.

Розберемо приклади встановлення показника «час реакції», представлені на рисунку 1.11 [3,26].



Риснуок 1.10 – Налаштування часу відповіді при обробці запиту сервером

Перший приклад. Час відповіді — це поняття часу, що мине з моменту доступу користувача до служби FTP для передачі файлу з сервера один на клієнтський комп’ютер 1 до кінця цієї передачі.

Для кінцевого клієнта, встановлений час реакції є ясною і найбільш природною ознакою продуктивності комп'ютерної мережі (файловий обсяг, який вносить певну невизначеність у показник, можна зафіксувати, оцінивши час реакції під час передачі, наприклад, 1 – го Мбайта даних). Фахівця мережі цікавить продуктивність безпосередньо мережі, для найчіткішої оцінки доцільно обчислити з часу реакції складові, які відповідають стадіям немережевої обробки інформації – пошуку потрібних даних на диску, записи їх у диск тощо. Отримане в результаті цих скорочень час можна вважати другим мережним визначенням часу реакції на прикладному рівні.

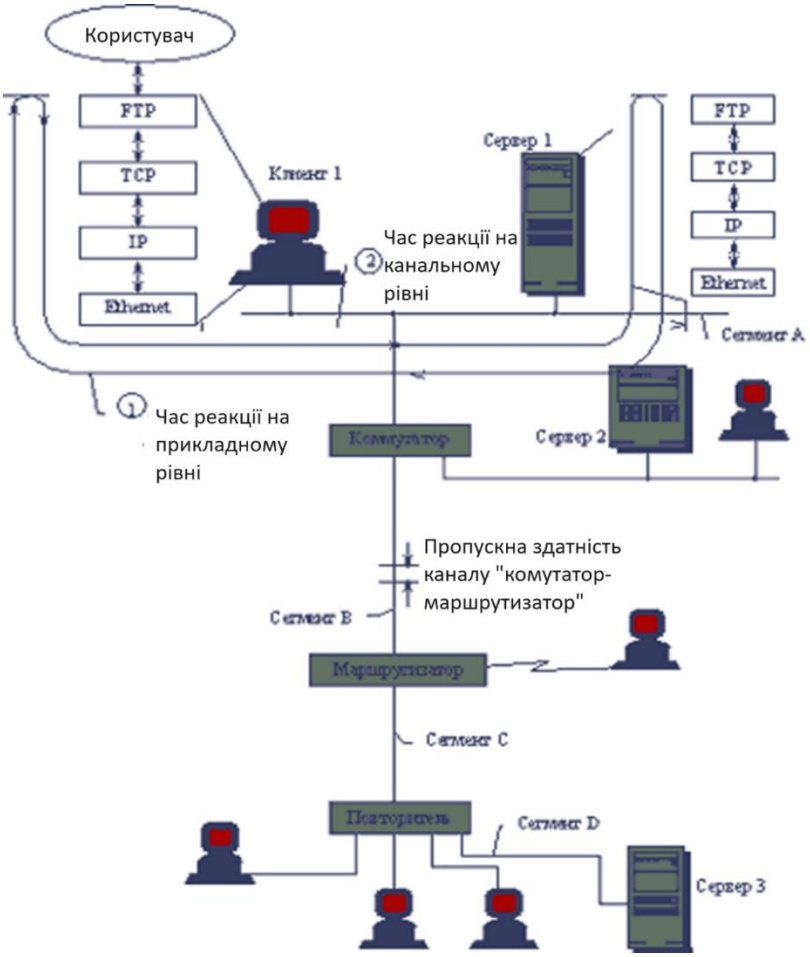


Рисунок 1.11 – Час реакції в мережі

Варіантами цього аспекту можуть бути час реакції, вирахований при різних, але фіксованих мережевих станах:

Повністю розвантажена мережа. Час реакції визначається, коли до сервера один звертається лише клієнт один, тобто на мережевому сегменті, що об'єднує сервер 1 клієнт 1, немає жодної іншої активності – на ній є лише кадри сесії FTP, ефективність якої вимірюється.

Завантажена мережа. При вимірі аспекту продуктивності, як у мережі працюють й інші вузли, і сервіси, з'являються свої складнощі – в мережі може бути велика кількість варіантів навантаження, тому головне під час визначення критеріїв такого сорту – виконання вимірювань за певних типових умов роботи в мережі. Т. к. трафік у мережі має пульсуючий характер, і особливості трафіку значно змінюються залежно від часу дня і дня тижня, тоді встановлення типового навантаження – процес складний, що вимагає тривалих вимірювань у мережі.

Оцінюючи вироблення комп'ютерної мережі щодо конкретним вузловим парам, а всім вузлам загалом використовуються аспекти 2 – х типів: середньо – зважені і порогові.

Середньозважений аспект є сумою часів реакції всіх або деяких вузлів при взаємодії з усіма, або кількома мережевими серверами по певному сервісу, тобто суму виду

$$(\sum i \sum jT_{ij})/(n \cdot m),$$

де Тij – час реакції і-го користувача при зверненні до j-го сервера; n- кількість клієнтів; m– кількість серверів.

Якщо усереднення виконується і з сервісів, тоді представленому вираженні додасться ще 1 – про підсумовування – за кількістю сервісів, що враховуються. Оптимізація мережі з цього аспекту полягає у пошуку значень параметрів, у яких аспект має мінімальне значення чи крайньому разі вбирається у певне задане число.

Пороговий аспект відображає найгірший час реакції по всіх можливих комбінаціях клієнтів, серверів та сервісів:

$$max_{ijk}\, T_{ijk},$$

де і та j мають те саме значення, що й у попередньому випадку, а k означає тип послуги. Оптимізацію також можна виконати для мінімізації аспекту або досягнення певного цільового значення, яке вважається логічним з практичної точки зору.

1.5.2 Продуктивність

Ключове завдання, для вирішення якої вибудовується будь-яка мережа, – швидка інформаційна передача між машинами. Тому аспекти, пов'язані з продуктивністю мережі чи частини мережі, відображають якість виконання мережею головної функції.

Існує багато варіантів визначення аспектів цього типу, як і у випадку з

аспектами класу «час реакції». Варіанти можуть відрізнятися один від одного: обрана одиниця вимірювання кількості переданих даних, характер враховуваної інформації - тільки користувач або користувач разом із сервісними, кількість точок для вимірювання переданого трафіку, спосіб усереднення результатів мережі в цілому.

Аспекти, які відрізняються числом та розташуванням точок виміру. Продуктивність можна вимірювати між будь-якими двома - вузлами або мережевими точками, наприклад, між клієнтом - комп'ютером 1 і сервером 3, як представлено на рисунку 1.2. При цьому одержувані значення продуктивності будуть змінюватися при одних і тих самих мережевих робочих умовах залежно від того, між якими 2-ма точками проводяться вимірювання. Оскільки в мережі одночасно працює велика кількість комп'ютерів користувачів і серверів, то повну характеристику мережевої продуктивності надає комплекс пропускних здібностей, виміряних для різних поєднань машин, що взаємодіють (матриця трафіку мережевих вузлів). Відомі спеціалізовані вимірювальні засоби, що фіксують матрицю трафіку для будь-якого вузла комп'ютерної мережі.

Слід визначити єдину продуктивність мережі як середню кількість даних, переданих між усіма мережними вузлами за одиницю часу. Загальна мережна продуктивність може вимірюватися і в пакетах за секунду, і в бітах за секунду. При поділі мережі на сегменти або підмережі єдина мережна пропускна здатність дорівнює сумі продуктивності підмереж плюс продуктивність міжсегментних, або міжмережевих зв'язків.

1.5.3 Показники надійності та безвідмовності

Значною особливістю комп'ютерної мережі вважається надійність - здатність бездоганно функціонувати протягом тривалого часу. Дана властивість має 3 складові: надійність, готовність та сервісну зручність.

Підвищення готовності передбачає стримування у певних межах впливу відмов та збоїв на системну роботу за допомогою засобів контролю та корекції похибок, а також засобів автоматичного відновлення циркуляції даних у мережі

після виявлення несправності.

Аспектом оцінки готовності вважається коефіцієнт готовності, який дорівнює частині часу системного перебування у працездатному стані і може пояснюватись як ймовірність знаходження системи в робочому стані. Показник готовності вираховується як співвідношення середнього часу виробітку на відмову до суми тієї ж величини та середнього періоду відновлення. Системи з високою готовністю називають безвідмовними.

Основним методом підвищення готовності вважається надмірність, з урахуванням якої здійснюються різні варіанти безвідмовних архітектур. Тому для координації безвідмовності роботи мережеві компоненти, через які проходять ці маршрути, необхідно зарезервувати: повинні бути запасні кабельні зв'язки, якими можна буде скористатися при відмові 1-го з головних кабелів, всі пристрої комунікацій на магістральних шляхах повинні або самі реалізуватися за стійкою до відмови схемою збереженням всіх основних елементів, або для кожного комунікаційного приладу повинен бути аналогічний резерв.

Існують градації безвідмовних комп'ютерних мереж [21]:

- *висока готовність* (high availability)-;
- *відмовостійкість* (fault to learance)– ;
- *безперервна готовність (continuous availability)*-.

Оскільки у локальних мережах відновленням втрачених даних займаються, зазвичай протоколи транспортного чи прикладного рівня, які працюють із тайм – аутами кілька 10 – ов секунд, то втрати пропускну здатність через – за низької мережевої надійності можуть становити 100 – і %.

1.6 Файлове розміщення на вузлах комп'ютерної мережі

Завдяки техніці комутації пакетів були проведені глибокі та прикладні наукові дослідження в 3 напрямках. Перший напрямок пов'язаний з розробкою основ теорії комутації пакетів у системах розподілу [19, 33].

Другий напрямок поглиблених досліджень пов’язаний з математичною

теорією оптимізації потоків у мережах та вибору вигідних мережевих маршрутів з комутацією пакетів [34].

Третій напрямок – здійснення науково-прикладних досліджень щодо розробки сучасних апаратно-програмних засобів техніки комутації пакетів [19]. Загалом дослідження обумовлені необхідністю підвищення продуктивності та надійності системи, зниження загальних витрат і розширення спектру послуг, що надаються [19, 33, 34].

– Закінченням сеансу передачі є оповіщення від комутатора цьому маршруті про закінчення передачі останнього пакета вихідного масиву даних.

На рисунку 1.12 представлена проста схема транспортної системи [33], де обрані 4-і маршрути з відповідною кількістю комутаторів. Ефективність операції буферного заповнення дізнаватимемося у відносних тимчасових одиницях – у байт – тактах. Один байт – такт відповідає часу виконання 1 – ї операції PUSH чи POP; іншими словами, 1 - н байт - такт дорівнює часу прийому в буфер 1 - го байта інформації. І якщо пакет міститься D+h байтів, тоді час передачі (прийому) даного пакета з (в) буфера дорівнює D+h байт – тактам.

Так, ключовими параметрами транспортної системи з пакетною комутацією є число комутаторів (r) і довжина заголовка (h), від яких залежить час доставки інформаційного масиву. Безпосередньо величини r і h визначають можливість адаптації часу доставки інформаційного масиву D до певних умов роботи бізнес-додатків. Тому процедура вибору необхідного часу доставки інформаційного масиву обсягом D байтів з заданих показників величин r і h і з обмежень, які є як даних величин, так деяких технічних особливостей транспортної системи з пакетної комутацією, називають rh – оптимізацією.

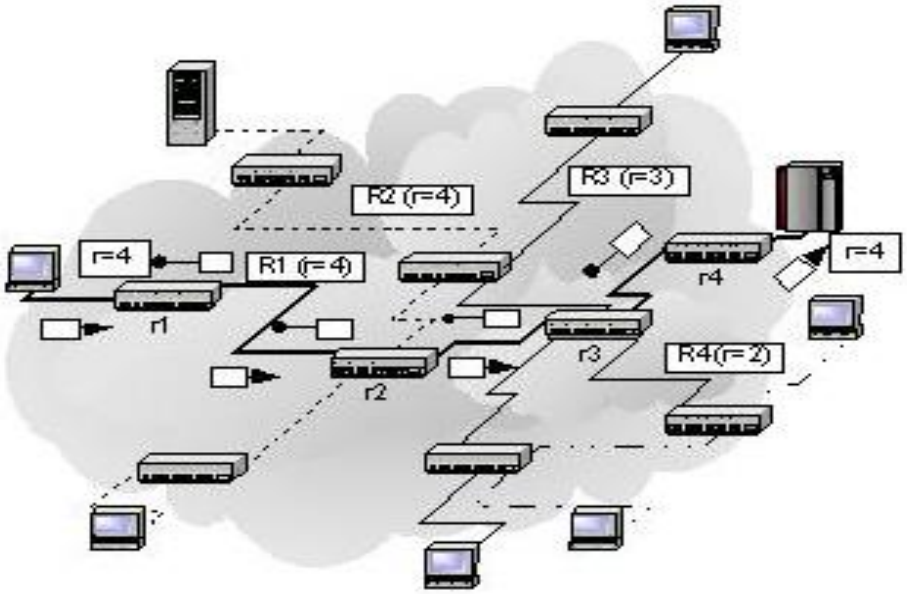


Рисунок 1.12 – Транспортна система з 4-ма маршрутами

Нехай пакет інформації, який містить D байтів корисних відомостей та h_A байтів заголовка, повинен пересилатися в транспортну систему для транспортування $D + h_A$ байтів до інформаційного приймача. Тоді час транспортування інформаційного пакета [33] від джерела A до приймача B, r комутаторів, кожен з яких вносить припинення пакета на $h_i \geq 0$ байт - тактів, буде одно.

$$T_{A\to B}^{D} = D(r+1) + \sum_{i=1}^{r+1} h_i \text{ байт-тактам,}$$

де $h_1 = h_A$, у протилежному випадку $h_A := h_1$.

Найменший час передачі D байтів корисних даних та h_A байтів заголовка [33] між хостом – джерелом і хостом – приймачем n пакетами через підібраний маршрут R з r комутаторами, кожен із яких вносить припинення пакета на$h_i \geq 0$байт - тактів, одно

$$min\, T_n^D = D + 2\sqrt{Dr^* max\{h_i\}} - max\{h_i\} + \sum_{i=1}^{r+1} h_i \text{ байт-тактам.}$$

Розглянемо нерідкий випадок передачі інформаційного масиву від A до B – коли$h_i = h_A \geq 1 \ \forall i = \overline{1,r}$. Це означає, що розмір заголовка пакетів, що пересилаються, по всьому маршруту R залишається незмінною величиною. Ця ситуація більш характерна для сучасних мереж із пакетною комутацією. Тому що при цьому

$$\sum_{i=1}^{r+1} h_i = (r+1)h_A; \quad max\{h_i\} = h_A, T_n^D = \frac{D}{n}(r+n) + h_A(r+n)$$

$$= \left(\frac{D}{n} + h_A\right)(r+n).$$

$$min\, T_n^D = D + 2\sqrt{Drh_A} - h_A + \sum_{i=1}^{r+1} h_A = D + 2\sqrt{h_A rD} + h_A r = \left(\sqrt{h_A r} + \sqrt{D}\right)^2.$$

Наступний аналіз відбуватиметься з урахуванням,$h_i = h_A \geq 1 \forall i = \overline{1,r}$, хоча подібні підсумки можна отримати і у випадку. Так само можна вважати, що$h_A = h$. Тоді$T_n^D = \left(\frac{D}{n} + h\right)(r+n) \ \ min\, T_n^D = \left(\sqrt{hr} + \sqrt{D}\right)^2$.

Співвідношення величин h та r вважається визначальним значенням транспортної системи з пакетною комутацією. Адаптація цього співвідношення, до конкретної комутаційної системи, дозволяє досягти необхідних показників показників продуктивності за мірками часу доставки інформаційних масивів.

На рисунку 1.13 представлені графіки функції $T_n^D = (D+h)(r+n)$, для ситуацій коли $r > h$,$r < h$ і $r = h$. Фігурально величини h і r – інь та янь мереж пакетної комутації. При оптимізуванні транспортної системи можуть бути використані властивості функціоналу T_n^D, пов'язані з можливістю досягнення рівної продуктивності при різних значеннях$r(h)$.

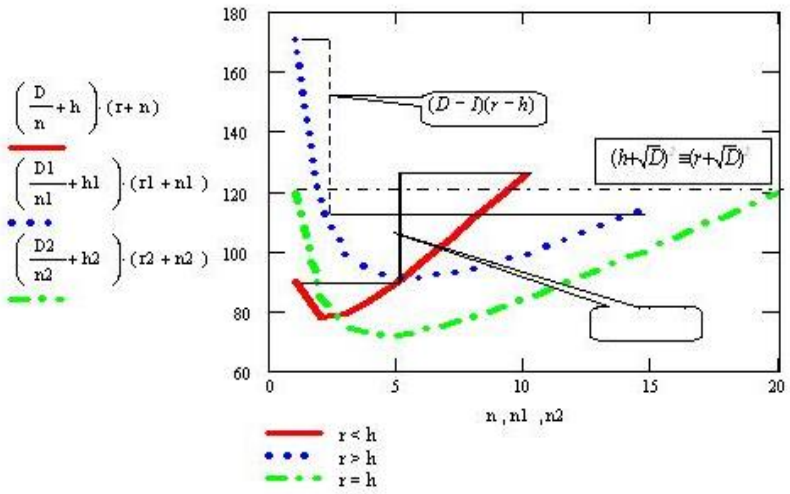


Рисунок 1.13 – Співвідношення r, h и $T_n^D(n)$

Для цього потрібно дізнатися$n_1 і n_2$, за яких$T_{n_1}^D = T_{n_2}^D$.

$$(n_2)_1 = \frac{Dr + hn_1^2 \pm (Dr - hn_1^2)}{2hn_1} = \frac{Dr}{hn_1};\ (n_2)_2 = n_1.$$

Так, отримуємо, що$T_{n_2}^D = T_{\frac{Dr}{hn_1}}^D$. Для показників$D = 1024,\ h = 5$ и$r = 20$дізнаємося$n_1 = 32$.

Потім$T_{32}^D = \left(\frac{1024}{32} + 5\right)(20 + 32) = 1924$. При$n_1 = 32\ \ n_2 = \frac{Dr}{hn_1} = \frac{1024*20}{5*32} = 128$.

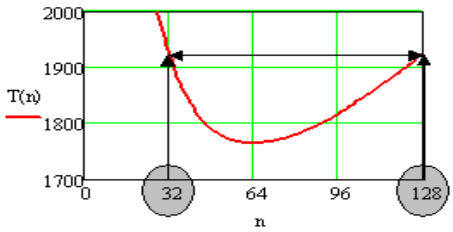


Рисунок 1.14 – Графік функції$T(n)$. $T_{32}^D = T_{128}^D$

Для показника величини$n_2 = 128$вирахуємо$T_{128}^D = \left(\frac{1024}{128} + 5\right)(20 + 128) = 1924$. Так, отримали, що$T_{128}^D = T_{32}^D$. Цей випадок представлений рисунку 2.14.

Показник часу доставки інформаційних масивів може бути основним

критерієм розробки багатьох інтелектуальних методик з архітектурних якостей додатків.

Уявімо, що комутатори транспортної системи обслуговують десять каналів пакетної передачі і мають розмір буферів, що входять і виходять, по 256 Бт. По кожному каналу мають передаватися інформаційні масиви обсягом 1024 байт. При$n = 64$гарантується мінімум часу доставки. Прикладні завдання наводять, що ця транспортна система має забезпечувати час доставки інформаційних масивів за 1900 – 2000 байт – тактів. Так як$T_{32}^{D} = 1924$при$n = 32$Тоді доцільно доставляти масив інформації обсягом 1024 байтів 32 пакетами по тридцять сім байтів. Але при цьому приблизно 30% пакетів буде втрачено, що призведе до збільшення трафіку за рахунок передач, що повторюються. Для вирішення необхідної задачі можна застосувати вирази$T_{n_2}^{D} = T_{\frac{Dr}{hn_1}}^{D}$. В цьому випадку$T_{128}^{D} = T_{32}^{D}$. Тому через комутатори, які обслуговують десять каналів масиви інформації, можна передавати без втрат за допомогою 128 пакетів, кожен з яких міститиме тринадцять байтів даних. Загалом спектр показників величиниT_n^D(от T_1^D до T_D^D) можливо адаптувати до параметрів конкретної системи за допомогою коригування значень величини$n = \overline{1, D}$.

Теоретично, час доставки інформаційного масиву буде мінімальним, якщо між джерелом та приймачем інформації немає транзитних вузлів (комутатори, маршрутизатори) та пакет без заголовка. У цій ситуації час доставки інформації обсягом D байтів, дорівнюватиме D байт – тактам. Якщо інформаційний пакет містить заголовок, що складається з h байтів, мінімальний час доставки складе D+h байт – тактів.

p align="justify"> При формуванні ресурсів інтелектуального управління мережами пакетної комутації з оптимізуванням часу доставки інформаційних масивів потрібно створити підсистему корекції параметрів враховуючи необхідності округлення показників величини n у вищенаведених випадках.

У багатьох сучасних мережах з комутацією формати пакетів, що пересилаються, передбачають наявність фіксованого службового поля (h) та інформаційного поля (D). Зазвичай, довжина інформаційного поля визначає

найменший і найбільший розмір пакета, що пересилається. Отже, у мережах можливості забезпечення найменшого часу доставки інформаційних масивів можуть бути обмежені.

Висновки до розділу 1

1. Проведено огляд літературних джерел, який показав, що ефективність використання відомих методів оптимізації розміщення інформаційних файлів по вузлах комп'ютерної мережі значною мірою залежить від структури мережі. Відомі методи ефективні для завдань невеликої розмірності або мають загальний характер.

2. Досліджено тенденції розвитку сучасних СБД. Показано, що при проектуванні СБД найважливішим завданням є вибір таких системотехнічних, алгоритмічних та програмних рішень, які дозволили б реалізувати систему апаратно-програмної підтримки операцій над даними на основі перспективної мікропроцесорної техніки та методів сучасних мов програмування високого рівня.

3. Досліджено методи підвищення продуктивності СБД – апаратна та програма реалізації операцій реляційної алгебри, вертикальний та горизонтальний паралелізм.

4. Відомі апаратно-програмні засоби важко застосовні при цілісному проектуванні нових РБД. Тому для обробки великих масивів інформації необхідно використовувати нові методи та засоби оптимізації, оптимізації розміщення інформаційних файлів по вузлах комп'ютерної мережі.

5. Огляд літературних джерел показав, що при переході від централізованих систем обробки інформації та управління відсутня методологія побудови РБД, що забезпечує можливість передачі даних з контрольованою та прогнозованою величиною затримок, практично відсутні прикладні методи розрахунків, тому існує необхідність у створенні таких методів.

2 ДОСЛІДЖЕННЯ МЕТОДИКИ ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНОЇ СПЕЦИФІКАЦІЇ ТИПОВОЇ СИСТЕМИ РОЗПОДІЛЬНОЇ СИСТЕМИ В ІНФОРМАЦІЙНИХ ІНФРАСТРУКТУРАХ

2.1 Оптимізація розподілу реєстрацій по вузлам локальної комп'ютерної мережі на основі чисельного критерію якості обслуговування

Оптимізація комп'ютерних мереж передбачає визначення:

- аспекти ефективності мережної роботи;
- безлічі змінних мережевих властивостей, які безпосередньо чи опосередковано впливають аспекти ефективності;
- порога чутливості показників аспекту ефективності.

Методи забезпечення якості обслуговування базуються на підтримці певної смуги пропускання; скорочення ймовірності втрати кадрів; виключення чи керованості мережних перевантажень; можливості конфігурування мережного трафіку; встановлення кількісних характеристик трафіку та ін. [17 – 20].

Якість обслуговування комп'ютерної мережі, зазвичай, є інтегральним показником, складові якого змінюються залежно від вимог до мережі.

Для отримання чисельних параметрів, формованих комп'ютерних мереж, можна використовувати засоби моделювання, з яких створюються моделі, відтворюють інформативні процеси, які у мережах. Можна застосовувати методи натурного (фізичного) прогнозування, а також математичного: моделі імітації та системні моделі масового обслуговування (СМО) та ін. [21 – 23].

У структурі комп'ютерної мережі виділяються підмережі:

- традиційного мережевого управління (конфігурацією, продуктивністю, безпекою);
- моделювання функціонування мережі, включаючи аналіз навантажень на окремі ділянки та підтримку прийняття рішень щодо перепланування.

У роботах [4, 8] досліджено питання розподілу інформаційних ресурсів за вузлами комп'ютерних мереж. Розглянуто алгоритми розміщення інформаційних файлів, орієнтовані оптимізацію. Як критерії оптимальності розглядалися середній

обсяг даних, що пересилаються каналами зв'язку в одиницю часу; загальний час обробки запитів; загальна вартість трафіку мережі та ін.

Виникає необхідність вибору числового критерію оптимальності, який визначає середній час виконання запитів користувачів і є зручним для оптимального розміщення файлів. Вибір такої характеристики СМО обумовлений тим, що користувачі, як правило, зацікавлені не в мінімізації розміру черги або якихось інших характеристик СМО, а в тому, щоб їхні запити оброблялися за якомога менший час.

При визначенні середнього часу очікування запитів W у черзі обслуговування рекомендується [24] використовувати таку формулу:

$$W = \frac{\rho^2}{\lambda(1-\rho)},$$

деρ - Коефіцієнт завантаження обслуговуючого пристрою $(0 \leq \rho < 1)$;

λ - Інтенсивність потоку запитів (середня кількість пакетів, які претендують на передачу за одиницю часу).

При постановці задачі оптимізації розміщення файлів серед вузлів мережі з метою отримання високої якості обслуговування можна величину b залишайте середній час обслуговування запиту (за винятком часу очікування обслуговування) постійним і не залежить від розміщення файлу. Величинаρзалежить відλі пропускна здатність обслуговуючого пристроюμ: $\rho = \frac{\lambda}{\mu}$. Слід зазначити, що параметрρграє ключову роль освіти черзі.

Зазвичай максимально допустимий час очікування запитів у черзі обслуговування M – постійно, тому максимально допустимий коефіцієнт завантаження обслуговуючого пристрою визначається з виразу:

$$\rho \frac{M}{M + b_{max}}$$

При розподілі запитів між N обслуговуючі пристрої повинні мінімізувати значення W, тоді як маршрут запиту заздалегідь невідомий, тобто запит може

оброблятися як одним обслуговуючим пристроєм, так і послідовно декількома обслуговуючими пристроями.

Цільову функцію (критерій якості) вибирають у вигляді комбінації параметрів трафіку каналами зв'язку в мережі:

$$Q(\rho) = \sum_{i=1}^{N} C_i \rho_i, \quad (2.1)$$

де C_i - Вагові коефіцієнти, що враховують середній час обслуговування пакетів каналом зв'язку: $C_i = b_i;\ \ b = \frac{\rho}{\lambda}$.

На відміну від застосовуваної процедури мінімізації середнього часу очікування запитів у черзі на обслуговування по всій мережі, пропонується мінімізувати функцію визначення параметрів трафіку по каналах зв’язку (залежно від ρ_i). Давайте встановимо якесь значення $\varepsilon \geq O(\rho^2)$, де $O(\dots)$– функція, що відображає часову складність алгоритму.

Для вирішення цієї проблеми необхідно визначити граничні умови. Для цього введемо функцію $W(\rho)$ у вигляді

$$W(\rho) = b\rho + O(\rho^2).$$

Звідси $\frac{b\rho^2}{1-\rho} \leq \varepsilon$ і, отже, отримаємо обмежуючу умову:

$$\rho \leq \frac{-\varepsilon + \sqrt{\varepsilon^2 + 4b\varepsilon}}{2b}.$$

Розглянемо приклад визначення обмежує умови для базового варіанта найбільш популярної на сьогоднішній день (понад 90% ринку) мережі Ethernet з такими характеристиками: стандарт IEEE 802.3, специфікація 10 BASE-T; швидкість передачі $-10\,\frac{\text{Мбит}}{c}$; мінімальна довжина пакета – 512 біт; максимальна кількість абонентів – до 1024; початковий кадровий інтервал - 96 мс.

Час обслуговування зазвичай не залежить від характеристик заявок (довжини пакета). Тому вважатимуться, що пропускна спроможність у кожному каналі

зв'язку однакова і постійна:$\mu_i = \mu = const$.

На рис. 2.1 представлено залежність зміниρвідε. Аналіз показує, що на завантаження каналів зв'язку основний вплив має величинаε. Тому під час вирішення завдання оптимізації середнього часу очікування запитів у черзі обслуговування каналами зв'язку можна мінімізувати функцію (2.1). При цьому практично точність не втрачається. Якщо необхідно враховувати обмеження на середнє завантаження каналів зв'язку, можна збільшити коефіцієнти цільової функції.

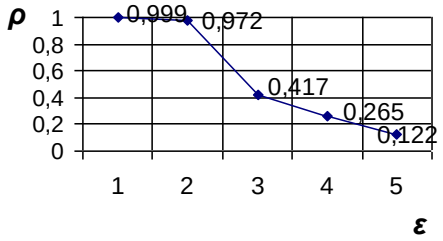


Рисунок 2.1 – Залежність зміни коефіцієнта завантаження обслуговуючого пристроюρвід величиниε

Таким чином запропоновано новий підхід до побудови математичних моделей оптимального розподілу файлів серед вузлів комп'ютерної мережі.

Використання таких моделей дозволяє отримувати квазіоптимальні рішення, послідовно коригуючи структуру та характеристики комп'ютерної мережі. З практичної точки зору отримані результати дозволяють істотно підвищити ефективність функціонування комп'ютерних мереж.

2.2 Метод оптимізації розподілених реєстрацій розміщення файлів у комп’ютерній мережі на основі часу очікування

Проблема створення комутаційних систем, призначених для аналізу стану мережі в кожний момент часу та оптимальне транспортування даних, повністю не вирішена. Системи, що виконують порівняно просту оптимізацію розподілу передачі даних каналами мережі, залишаються надзвичайно дорогими, а

ефективність використання можливостей універсальних комутаторів при передачі великих обсягів мультимедійної інформації кількома каналами одночасно – порівняно низькою.

При забезпеченні розрахованого на багато користувачів доступу до інформаційних ресурсів, що зберігаються у вигляді РБД, потрібно раціонально розмістити файли РБД у вузлах комп'ютерної мережі.

Відомо кілька відповідних математичних моделей, що відрізняються видом цільової функції та комплексом обмежень, що враховуються при пошуку оптимального методу [5, 6].

На сьогоднішній день для пошуку оптимальних рішень реальне застосування знаходять лише пакети прикладних програм, а саме Matlab [27].

Після виявлення оптимального рішення, зазвичай проводиться аналіз модельної стійкості та чутливості.

2.3 Організація оптимізації розподілених реєстрацій розміщення файлів у комп’ютерній мережі на основі теорії масового обслуговування

Розглянемо спосіб побудови моделі раціонального файлового розподілу РБД по вузлах комп'ютерної мережі, суть якого математичний апарат концепції масового обслуговування. Теорія черг полягає в основі вибудовування моделі комп'ютерної мережі в багатьох роботах з оптимізації файлового розподілу в РБД [21, 22, 32, 34], однак, як математична модель локальної мережі з шинною топологією береться СМО з 1 обслуговуючим приладом - єдиною шиною.

Проаналізуємо мережу, як багатоприладну СМО, тобто як систему, де кілька ідентичних приладів обслуговування обробляють 1 чергу заявок. Заявка на самому початку черги йде на обслуговування до одного з вільних приладів. Багатоприладна черга, представлена на рисунку 2.3, а відрізняється від координації черги на рисунку 2.3 б, де показано кілька одноприладових черг, що працюють паралельно. Якщо у всіх випадках прилади обслуговування та вхідний потік заявок ідентичні, а в одноприладових чергах заявки надходять випадково і, потрапивши в неї, залишаються там (інакше, перехід в іншу чергу заборонено), то виявляється, що

робота багатоприладової черги переважно одноприладових, що функціонують паралельно .

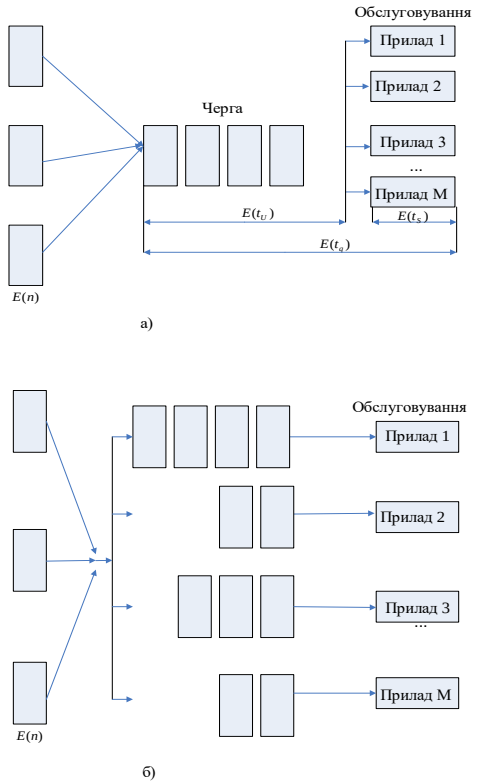


Рисунок 2.3 – Пакетна обробка при використанні черги з рядом обслуговуючих приладів (а) та кілька паралельних черг, з 1-м приладом обслуговування (б)

Встановимо оптимальну кількість копій для кожного файлу розподілених баз даних, розглядаючи комп'ютерну мережу, як ряд багатоприладових СМО, що перетинаються, з 1-ю чергою запитів до певного файлу.

Під час проектування СМО часто виконуються розрахунки на найгірший випадок. У цій ситуації оцінки не дуже точні, але принаймні похибки забезпечують запас стабільності. У реальних системах обслуговуючий час коливається. Розкид може бути викладений розрахунком середнього показника та середнього відхилення для обслуговуючого часу конкретних видів обладнання. Найкращий випадок має місце, коли час обслуговування незмінно, тобто стандартне відхилення =0 (тобто. немає відхилення від середнього показника). Найгірший випадок має також місце, коли час обслуговування відповідає експоненційному розподілу, тобто коли стандартне відхилення = середньому показнику (для стандартного

відхилення це занадто максимальна величина, яка показує, що є великий розкид значень обслуговуючого часу). Варто наголосити, що насправді експоненційний розподіл не завжди найгірший випадок; Наприклад, середній показник величин 5, 10, 20 і 200 = 58.75, а стандартне відхилення – приблизно 81 [32].

Позначимо середній показникxчерез$E(x)$ (математичне очікування), а типове відхиленняx -σ_x. Тоді

$$\sigma_x = \sqrt{\frac{\sum[x-E(x)]^2}{N}}. \quad (2.7)$$

Керуючись досвідченими відомостями, а не конкретними показниками, у виразі σ_x замість N використовується N-1,

де N – кількість дослідів.

Звідси

$$\sigma_x = \sqrt{\frac{\sum[x-\bar{x}]^2}{N-1}}, \quad (2.8)$$

де $\bar{x}$- Середній показник дослідної величини.

Оцінка середнього показника та типового відхилення тим більше точна, чим більше приймається досвідчених значень. Коли N має більшу величину, різниця між рівняннями (2.7) і (2.8) мізерна.

Інший метод визначення типового відхилення:

$$\sigma_x = \sqrt{E(x^2) - E^2(x)}.$$

У роботі [39] представлено, що 95% показників часу відповіді на запит не перевищують середнього значення часу у відповідь плюс 2 стандартних відхилення. Іншими словами, близько 5% відповідей тривають упродовж більшого часу, зазначеної величини.

З метою спрощення математичних розрахунків системне навантаження, як правило, виражено у відносних величинах у порівнянні з максимальним навантаженням, яке здатна обробити система. Як правило, значення позначається

буквоюρ. Як показано у поданих визначеннях, повністю зайняте обладнання має$\rho = 1$, а не завантажене $-\rho = 0$. Так, коефіцієнт застосування обладнання знаходиться в межах від нуля до одного, зрідка його виражають у відсотках.

Є правило [30], відповідно до якого крива часу у відповідь різко йде нагору, коли застосування обладнання стає вище вісімдесяти відсотків.

При проектуванні системи масового обслуговування ставиться мета, щоб її використання при постійних навантаженнях було в межах шістдесяти-сімдесяти відсотків [34].

Точний метод визначення коефіцієнта використання обладнання в СМО з 1 сервісним пристроєм задається виразом

$$\rho = E(n)E(t_s),$$

де$E(n)$– середня кількість звернень, що надходять за одиницю часу на обслуговування;

$E(t_s)$– середній час обслуговування 1-го запиту.

Уявімо, що є M службових пристроїв одного типу. Отже, на будь-який пристрій за одиницю часу він надсилає $E(n)/M$ запити. Отже, коефіцієнт застосування конкретного пристрою

$$\rho = \frac{E(n)E(t_s)}{M}. \quad (2,9)$$

Коефіцієнтрмає бути менше одиниці. Як зазначалося раніше, при прагненніρКрім того, стрімко зростає розмір черги та час перебування в ній.

Нехайwкількість запитів, які очікують на обслуговування в певний час, іq– кількість системних запитів, що очікують і обслуговуються в цей час.

Нехай іде далі t_w– час очікування послуги, і t_q – час перебування запиту в системі, тобто час, який він витрачає як на очікування, так і на обслуговування. Середні значення$w,\ q,\ t_w$ іt_qдавайте встановимо це відповідно через$E(w),\ E(q),\ E(t_w)$ і$E(t_q)$. Рівність завжди об’єктивна

$$t_q = t_w + t_s.$$

Засоби,

$$E(t_q) = E(t_w) + E(t_s).$$

Т.к.$E(n)$– середня кількість вхідних запитів, а потім аналізується стабільний стан

$$E(w) = E(n)E(t_w), \tag{2.10}$$

і

$$E(q) = E(n)E(t_q).$$

Кількості$w,\ q,\ t_w it_q$посилаються на запити, які очікують на будь-який із серверів M. Підставляючи в

$$E(q) = E(n)E(t_q) = E(n)E(t_w) + E(n)E(t_s),$$

відповідних величин із рівнянь (2.9) і (2.10), отримуємо

$$E(q) = E(w) + M\rho.$$

Імовірність наявності N запитів у системі в заданий момент часу [22]

$$P(q = N) = \frac{(M\rho)^N}{N!} P_0, \quad \text{если} \quad N < M,$$

і

$$P(q = N) = \frac{(M\rho)^N}{M!\, M^{N-M}} P_0, \quad \text{если} \quad N \geq M,$$

де

$$P_0 = \left[\sum_{N=0}^{M-1} \frac{(M\rho)^N}{N!} + \frac{(M\rho)^M}{(1-\rho)M!}\right]^{-1}.$$

Імовірність того, що всі службові пристрої зайняті в даний момент часу, дорівнює

$$B = P(q \geq M) = \sum_{N=M}^{\infty} P(q = N),$$

і розраховується за формулою

$$B = \frac{1 - \frac{\sum_{N=0}^{M-1} \frac{(M\rho)^N}{N!}}{\sum_{N=0}^{M} \frac{(M\rho)^N}{N!}}}{1 - \rho \frac{\sum_{N=0}^{M-1} \frac{(M\rho)^N}{N!}}{\sum_{N=0}^{M} \frac{(M\rho)^N}{N!}}}.$$

Рівняння зводиться до $B = \rho$, коли $M = 1$. Фактор B присутній у всіх інших рівняннях для систем із кількома сервісними пристроями. Для визначення його кількісних показників описано функцію fact_B, яка визначає ймовірність завантаження всіх пристроїв залежно від числового значення коефіцієнта використання обладнання.ρі кількість сервісних пристроїв M.

У QS з кількома сервісними пристроями середня кількість запитів, які очікують на обслуговування, становить

$$E(w) = B\frac{\rho}{1-\rho}.$$

Так,

$$E(q) = B\frac{\rho}{1-\rho} + M\rho.$$

Типове відхилення для w =

$$\sigma_w = \frac{1}{1-\rho}\sqrt{B\rho(1+\rho-B\rho)}.$$

Середній період очікування перед обробкою

$$E(t_w) = \frac{BE(t_s)}{M(1-\rho)}.$$

Отже, середній час перебування в черзі

$$E(t_q) = \frac{BE(t_s)}{M(1-\rho)} + E(t_s).$$

Типовий період очікування перед відхиленням обробки

$$\sigma_{t_w} = \frac{E(t_s)}{M(1-\rho)}\sqrt{B(2-B)}\,,$$

і типове відхилення терміну перебування в черзі

$$\sigma_{t_q} = \frac{E(t_s)}{M(1-\rho)}\sqrt{B(2-B)+M^2(1-\rho)^2}\,.$$

Імовірність того, що час очікування перевищує t, визначається такою формулою

$$P(t_w \geq t) = Be^{-M(1-\rho)t/E(t_s)}\,.$$

Справжню систему можна скоординувати так, щоб кілька запитів взагалі не чекали на обслуговування($t_w = 0$), і невелика частина з них затримається на тривалий період часу. У цьому випадку середній час очікування відкладеного запиту значно перевищує E(t_w).

Задамо середній час затримки E(t_d) як середній період часу для запитів, які повинні чекати.

Ймовірність того, що запит опиниться в черзі, дорівнює B. Отже, середній час очікування становить

$$E(t_w) = BE(t_d) + (1 - B)0 = BE(t_d) \ .$$

Але

$$E(t_w) = \frac{BE(t_s)}{M(1-\rho)}.$$

Тому

$$E(t_d) = \frac{E(t_s)}{M(1-\rho)}.$$

Попередні рівняння для черг із кількістю серверів базуються на припущенні, що час обслуговування має експоненціальний розподіл. Немає простих виразів, які описують багатоінструментальні системи QS, які мають кращий час обслуговування, ніж експоненціальний розподіл, але для оцінки в таких ситуаціях було б корисно використовувати апарат математичної апроксимації [22].

Відзначимо кілька випадків, коли описана теорія невірна. Наведені вище формули служать для наближення найскладніших ситуацій, які існують в реальності. Причина полягає в припущенні довільності надходження заявки та (іноді) індикативності часу обслуговування. Насправді може бути більш сприятливий запит, ніж випадковий.

Але є 2 типи ситуацій, коли черги та затримки набагато гірші, ніж отримані з наведених вище формул.

По-перше, через короткий проміжок часу може надійти максимальна кількість запитів.

У деяких випадках не можна вважати, що значення часу прибуття відповідає розподілу Пуассона. Варто підкреслити, що більшість програм прогнозування для цих систем розроблені для пуассоновського вхідного потоку подій

Для вибору більш підходящої моделі з тих, що пропонуються, необхідно провести оцінку вимог користувачів таблиця 2.1 демонструє, передбачені чи ні в моделях такі особливості комп'ютерних мереж, інформаційних баз і додатків.

Таблиця 2.1 - Зіставлення моделей

	1	2	3	4	5	6
Модель I	+	+	-	-	-	-
Модель II	+	-	+	+	+	-
Модель III	-	+	-	+	+	+

Вивчаючи чисельні результати реалізації моделей I і II, побудованих з урахуванням єдиного підходу, можна підкреслити такі особливості моделей:

1) отримані приклади тестів матриці оптимального розміщення розподілених баз даних показують величезну залежність між обраним аспектом оптимальності та кінцевою матрицею розподілу;

2) в отриманих матрицях раціонального файлового розподілу при мінімізуванні середнього обсягу даних, що відсилаються, і при мінімізації єдиного часу обробки абсолютно всіх запитів, які надійшли в систему в одиницю часу, чітко порушується припущення рівномірної завантаженості мережних вузлів. Зрозуміло, що найбільше будуть навантажені 1-і вузли.

3) Щоб підвищити пропускну здатність системи, як допоміжну умову можна застосувати обмеження на тривалість очікування запиту від будь-якого вузла. Нехай a_{ijs} – час очікування, який потрібний для виконання запиту, розпочатого у вузлі K_j до файлу F_i, що міститься в s-му вузлі; T_{ij} – максимальний час виконання запиту для файлу F_i, ініційований у вузлі K_j. Між значеннями a_{ijs} і T_{ij} є відносини

$$a_{ijs}(1 - x_{ij})x_{is} \leq T_{ij},$$

для $j \neq s,\ 1 \leq i \leq m$. Для того, щоб з цього співвідношення отримати обмеження, потрібні величини a_{ijs} виразити через змінні x_{ij}. Це зробити дуже важко;

4) Наведена схема обробки запитів практично не вписується в паралелізм обробки інформації в мережі, а також не враховує дуже поширену ситуацію складних запитів (одночасний доступ до кількох файлів з 1 вузла). Наприклад, локальна база даних хоста K_j містить файли F_i і F_{i+1}, а локальна інформаційна база вузла K_{j+1} файли F_{i+1} і F_{i+2}. На вузлі K_j починається складний запит файлу F_i і F_{i+1}. За заданою схемою обидва ці файли будуть оброблятися у вузлі K_j по черзі. Однак, було б логічніше під час пошуку файлом F_i на вузлі K_j надіслати запит на обробку файлу F_{i+1} у вузол K_{j+1} (якщо він не завантажений);

5) Але якщо питання вирішується сукупно, тобто як оптимізацією програмного забезпечення, і навіть апаратним оновленням, тоді навантаження деяких мережевих вузлів не вплине на швидкість роботи. Тому, незважаючи на перераховані вище недоліки, дані моделі оптимізаційної задачі можна застосувати на практиці при проектуванні певних РБД.

Так, запропоновані математичні моделі оптимізаційної задачі файлового розміщення РБД по вузлах локальної мережі можна успішно застосувати під час проектування певних розподільчих баз даних, використовуючи попередню оцінку вимог користувачів та програмний комплекс з метою статистичного збору та оптимального перерозподілу запитів.

2.4 Координація оптимізації файлового розміщення РБД за єдиним часом запитового обслуговування

Вирішенню задачі раціонального розміщення файлів інформації по вузлах локальної мережі присвячено кілька робіт [1, 5, 6, 10 – 14, 32 – 34], які відрізняються і постановкою задачі, і методами її вирішення.

Досліджуємо мережу із топологією єдиної шини. Локальні мережі з шинною топологією розрізняє відносна легкість управління, малі витрати часу на арбітраж, простота розширення і досить висока надійність (через паралельні підключення вузлів до каналу) [34].

Нехай запит, який надходить на будь-який мережевий вузол, передбачає

доступ до файлу РБД. Розрізнятимемо 2 типи запитів: запити пошуку та запити на виправлення. Запити обслуговуються у вузлі порядку надходження. Щоб заощадити ресурси, систему пріоритетів не запроваджуємо. Запит пошуку ініціюється у конкретному вузлі. Якщо копія необхідного файлу міститься в локальній вузловій базі, з якого надійшов запит, він обробляється. Якщо копії необхідного файлу немає у локальній базі цього вузла, запит пошуку надсилається у вільний вузол, який містить копію необхідного файлу, там обробляється і результат пересилається у вихідний вузол.

Як аспект раціональності приймається єдиний час, необхідний обслуговування запитів, які надійшли до системи протягом одиниці часу. Топологія шини, однотипність зв'язкових ліній та їх мала довжина в локальних мережах роблять час відправки незалежним від вузла запиту та вузла передачі.

Нехай

n - кількість мережевих вузлів;

m –кількість незалежних файлів розподілених баз даних;

K_j - j-й паровий вузол;

F_i - і-файл розподілених баз даних;

L_i – обсяг файлу F_i;

b_j - об'єм пам'яті вузла K_j який призначений для розміщення файлів;

s - кількість типів запитів пошуку;

λ_{kij} - інтенсивність запитів пошуку k-типу до файлу F_i з вузла K_j;

t_{kij} - час обробки запиту пошуку k-типу до файлу F_i у вузлі K_j;

$T_{ki}^{(1)}$ - час надсилання запиту пошуку k-типу до файлу ;

$T_{ki}^{(2)}$ - час надсилання відповіді на запит пошуку k-типу до файлу F_i;

r – кількість видів виправлень;

λ'_{lij} - інтенсивність виправлень l-типу файлу F_i з вузла K_j;

t'_{lij} - час обробки виправлення l-типу файлу F_i у вузлі K_j;

T'_{li} - час надсилання виправлення l-типу файлу F_i;

$x_{ij}(i = 1, \dots, m; \quad j = 1, \dots, n)$ – величини, що визначаються за формулою

$$x_{ij} = \begin{cases} 1, & \text{якщо копія файлу } F_i \text{ знаходиться в вузлі } K_j; \\ 0, & \text{якщо копія файлу } F_i \text{ не знаходиться в вузлі } K_j. \end{cases}$$

Час, який витрачається на відправлення із вузла K_j даних під час виконання запиту пошуку k–типу до файлу F_i, одно $(T_{ki}^{(1)} + T_{ki}^{(2)})(1 - x_{ij})$. Тоді єдиний час, який необхідний для відправки даних зв'язковими каналами між вузлами при виконанні запитів пошуку, що надійшли в систему протягом одиниці часу, встановлюється як

$$T = \sum_{k=1}^{s} \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda_{kij} (T_{ki}^{(1)} + T_{ki}^{(2)})(1 - x_{ij}).$$

Прийнявши

$$a_{ij} = \sum_{k=1}^{s} \lambda_{kij} (T_{ki}^{(1)} + T_{ki}^{(2)}),$$

отримаємо

$$T = T_0 - \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij},$$

де $T_0 \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}$.

Час, який потрібний для відправки з вузла K_j інформації при виконанні виправлення l-типу файлу F_i, одно

$$\sum_{\substack{p=1 \\ p \neq j}}^{n} T'_{li} x_{ip}.$$

Тоді єдиний час, необхідний для відправки інформації зв'язковими каналами між вузлами при виконанні запитів коригування, які надійшли до мережі протягом

одиниці часу, встановлюється як

$$T' = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{\substack{p=1\\p\neq j}}^{n}\lambda'_{lij}T'_{li}x_{ip}.$$

Якщо покласти

$$a'_{ij} = \sum_{l=1}^{r}\lambda'_{lij}T'_{li},$$

то

$$T' = \sum_{i=1}^{m}\sum_{j=1}^{n}a'_{ij}\sum_{\substack{p=1\\p\neq j}}^{n}x_{ip} = \sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}\sum_{\substack{p=1\\p\neq j}}^{n}a'_{ip} = \sum_{i=1}^{m}\sum_{j=1}^{n}\hat{a}_{ij}x_{ij}\,,$$

де $\hat{a}_{ij} = \sum_{\substack{p=1\\p\neq j}}^{n} a'_{ip}$.

Час обробки запиту пошуку k-типу до файлу F_i з вузла K_j можна уявити формулою

$$t_{kij}x_{ij} + \hat{t}_{kij}(1 - x_{ij}) \widehat{=} \hat{t}_{kij} + (t_{kij} - \hat{t}_{kij})x_{ij},$$

де $\hat{t}_{kij} = \frac{1}{n-1}\sum_{\substack{p=1\\p\neq j}}^{n} t_{kip}$.

Єдиний час, який необхідний для обробки всіх пошукових запитів, що надійшли в мережу протягом одиниці часу, встановлюється як

$$t = t_0 + \sum_{i=1}^{m}\sum_{j=1}^{n}b_{ij}x_{ij}\,,$$

де

$$t_0 \equiv \sum_{k=1}^{s}\sum_{i=1}^{m}\sum_{j=1}^{n}\lambda_{kij}\hat{t}_{kij}\,,\ b_{ij} = \sum_{k=1}^{s}\lambda_{kij}\left(t_{kij} - \hat{t}_{kij}\right).$$

Час, який потрібний для обробки виправлення l-типу файлу F_i, одно

$$\sum_{q=1}^{n} t'_{liq} x_{iq}.$$

Уніфікований час, необхідний для виконання всіх запитів на виправлення, які надходять у мережу протягом одиниці часу, встановлюється як

$$t' = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{q=1}^{n} \lambda'_{lij} t'_{liq} x_{iq} = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{q=1}^{n} \lambda'_{liq} t'_{lij} x_{ij}.$$

Покладання

$$b'_{ij} = \sum_{l=1}^{r}\sum_{q=1}^{n} \lambda'_{liq} t'_{lij},$$

отримаємо

$$t' = \sum_{i=1}^{m}\sum_{j=1}^{n} b'_{ij} x_{ij}.$$

Відзначаючи $f_0 \equiv T_0 + t_0$, $C_{ij} = -a_{ij} + \hat{a}_{ij} + b_{ij} + b'_{ij}$, отримуємо точну модель задачі оптимального розподілу копій файлів між вузлами мережі з точки зору мінімального рівномірного часу, необхідного для обслуговування всіх запитів, що надходять в систему протягом одиниці часу, пов'язану з типом задач дискретного програмування з булевими змінними:

$$f(X) = f_0 + \sum_{i=1}^{m\Sigma}\sum_{j=1}^{n\Sigma} C_{ij} x_{ij} \to min, \quad (2.11)$$

під обмеженнями

$$\sum_{j=1}^{n} x_{ij} \geq 1 \quad (i = 1,2,\dots,m); \quad (2.12)$$

$$\sum_{i=1}^{m} L_i x_{ij} \leq b_j \quad (j = 1,2,\dots,n); \quad (2.13)$$

$$x_{ij} \in \{0,1\} \quad (i = 1,2,\dots,m;\ \ j = 1,2,\dots,n). \tag{2.14}$$

Алгоритмічна реалізація моделі

Для реалізації моделі (2.11) – (2.14) запропоновано алгоритм, який створює модель [6] для подальшого порівняння.

Блок-схема алгоритму наведена на рисунку 2.6. На першому етапі алгоритму знаходиться вихідний розподіл файлів, який буде раціональним, якщо не враховувати умови (2.13). На другому етапі перерозподіл файлів здійснюється за наявності хоча б індексу 1-n для вихідного розповсюдження. *j* така, що умова (2.13) не виконується. Другий етап алгоритму виконується до тих пір, поки не буде знайдено розподіл, що відповідає умові (2.13). Розглянемо етапи рекомендованого алгоритму.

Перший етап. Визначення початкового розподілу.

1. Визначення значень $C_{ij}(i = 1,2,\dots,m;\ \ j = 1,2,\dots,n)$ і обчислити матрицю $C = \{C_{ij}\}_{mn}$. Структурна схема створення матриці $C(m,n)$ показано на рисунку 2.7.

2. Якщо для $i\ \exists C_{ij} < 0,$

то

$$x_{ij}^{*} = \begin{cases} 1, C_{ij} < 0; \\ 0, C_{ij} \geq 0. \end{cases}$$

3. Якщо для $i\ \forall C_{ij} \geq 0$, то визначимо $\min\limits_{1 \leq j \leq n} C_{ij}$. Нехай $\min\limits_{1 \leq j \leq n} C_{ij} = C_{ij_i}$.

Потім

$$x_{ij}^{*} = \begin{cases} 1, j = j_i; \\ 0, j \neq j_i. \end{cases}$$

Алгоритм визначення початкового розподілу файлів по вузлах мережі в залежності від обраного аспекту оптимальності наведено на блок-схемі на рисунку 2.4.

Другий етап. Перерозподіл файлів.

1. Створіть вектор значень $E = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$, де $\varepsilon_j = 0$ $(j = 1,2, \dots, n)$. Під час роботи алгоритму, після перерозподілу файлу з певного заповненого вузла K_j відповідний компонентε_jвекторEнадається значення 1, і цей вузол закривається для перерозподілу.

2. Для всіх індексів j, де $\varepsilon_j = 0$, перевіряємо виконання умови (2.13). Якщо ця умова для всіх індексів j виконується, тоді алгоритм завершується. Якщо напевно $j = r$ маємо

$$\sum_{i=1}^{m} L_i x_{ir} > b_r,$$

Потім переходимо до третього пункту.

3. Якщо$\exists s \neq r$такий як

$$\sum_{i=1}^{m} L_i x_{ir} \leq b_s \,, \sum_{i=1}^{m} L_i x_{is} \leq b_r,$$

потім ми міняємо місцями пам'ять r-го і s-го вузлів і повертаємося до другої точки. В іншому випадку переходимо до четвертого пункту.

4. Для тих i, де $\exists j \neq r$ такий як $x_{ij} = 1$, визначте $\min_i(-C_{ir})$. Нехай

$$\min_i(-C_{ir}) = -C_{kr}.$$

Для тих i, де $x_{ij} = \begin{cases} 1, & j = r, \\ 0, & j \neq r, \end{cases}$ визначаємо $\min_j(C_{ij} - C_{ir})$, куди мало взято, відповідно до тих показників $j \neq r$, де $\varepsilon_j = 0$. Нехай

$$\min_j(C_{ij} - C_{ir}) = C_{ij_i} - C_{ir}.$$

Потім визначаємо $\min_j(C_{ij} - C_{ir})$. Нехай

$$\min_i(C_{ij_i} - C_{ir}) = C_{lj_l} - C_{lr}.$$

Якщо $min(-C_{kr}, C_{lj_l} - C_{lr}) = -C_{kr}$, потім у матриці X ми припускаємо $x_{kr} = 0$. Це означає, що файл F_k виключено з вузла K_r. Якщо $min(-C_{kr}, C_{lj_l} - C_{lr}) = -C_{lr}$, потім у матриці X забезпечити $x_{lr} = 0$, $x_{lj_l} = 1$. Це означає, що файл F_l з вузла K_r перерозподілено на вузол K_{j_l}. Такий перерозподіл файлів відповідає мінімальному збільшенню показника цільової функції.

5. Перевіряємо умову (2.13) $j = r$. Якщо не виконується, то переходимо до третього пункту. Якщо умова виконується, то елементу ε_r вектора E присуджуємо показник 1 і переходимо до другого пункту.

Таким чином, алгоритм дає змогу знайти оптимальний або майже раціональний розподіл файлів між вузлами мережі за скінченну кількість кроків. Результатом роботи алгоритму є матриця X.

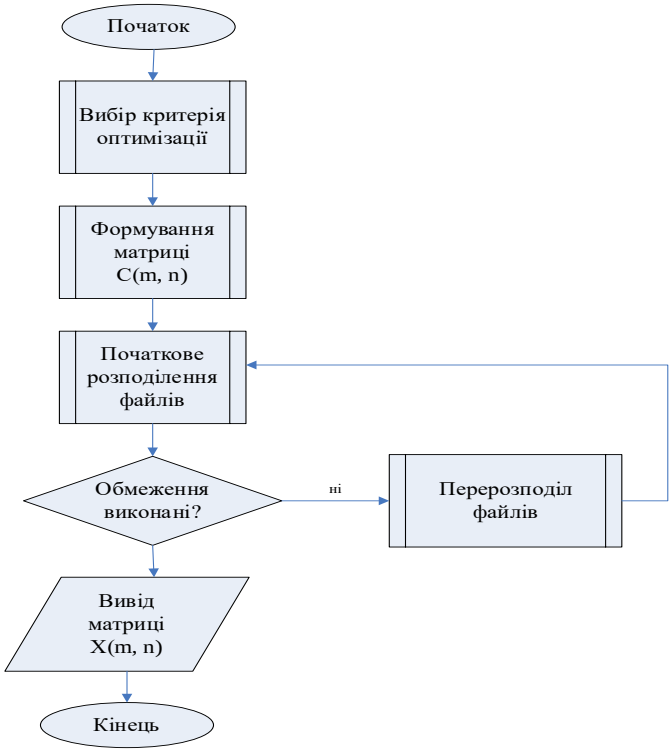


Рисунок 2.4 – Основна алгоритмічна блок-схема

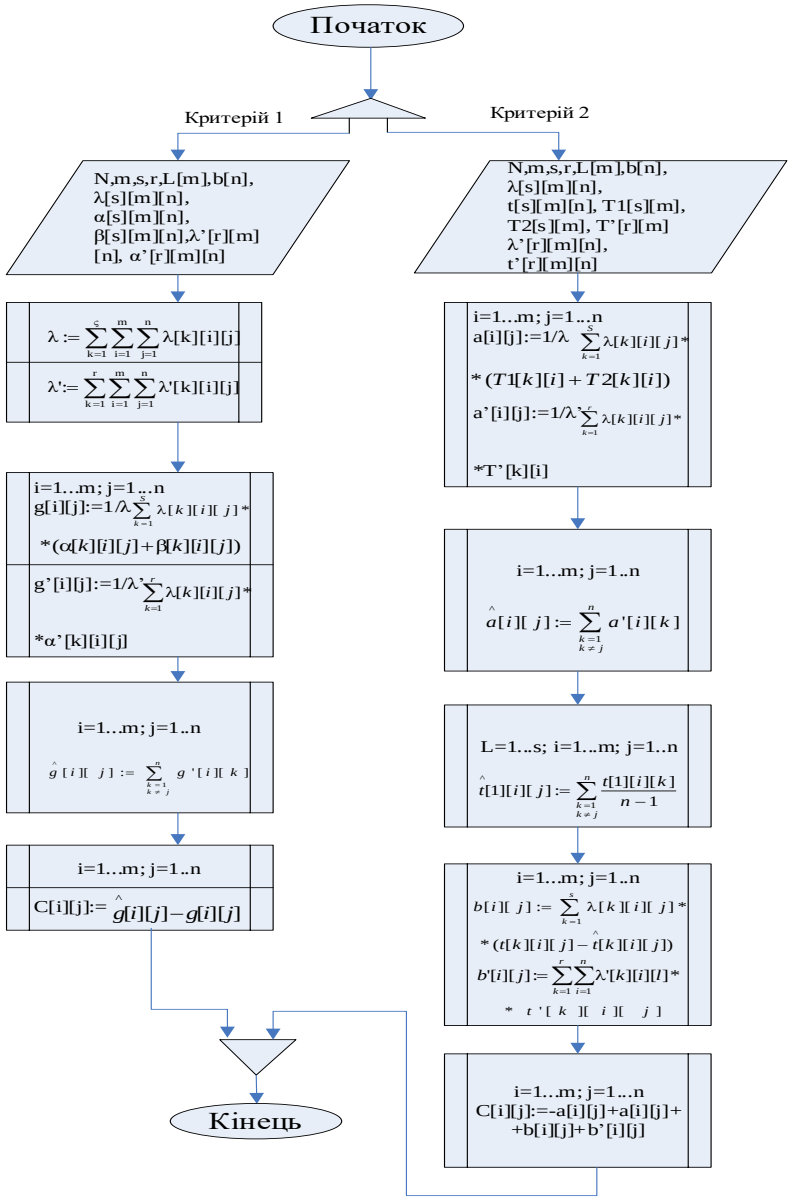


Рисунок 2.5 – Створення матриці цільової функції

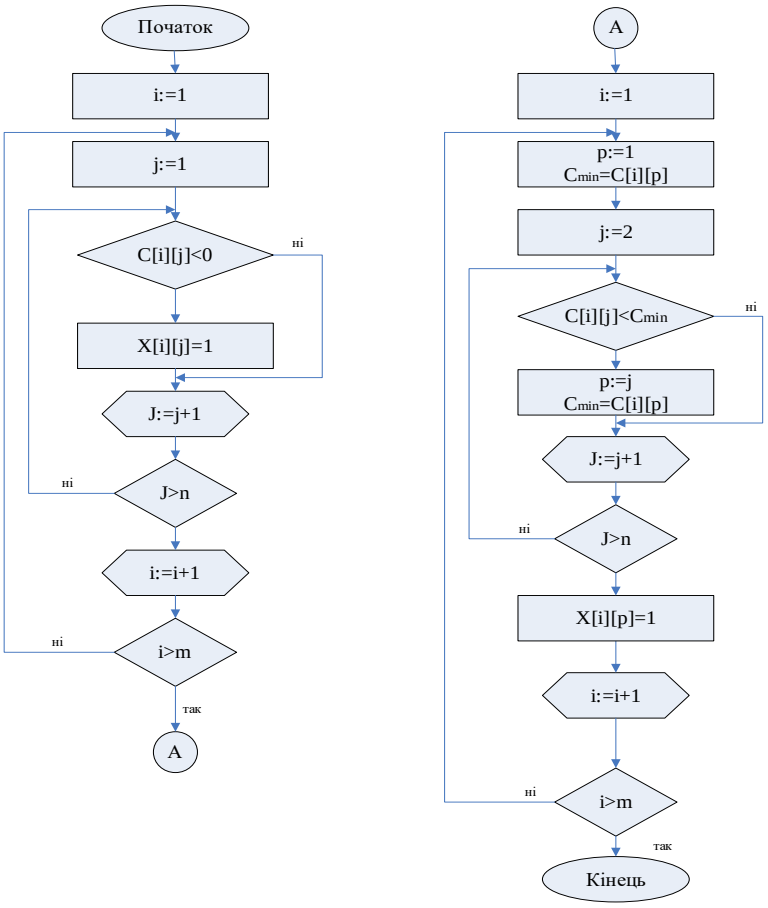


Рисунок 2.6 –Блок-схема – вихідна схема розподілу

Для реалізації описаного вище алгоритму ми зібрали програму мовою $C(opt1)$. Для 1 з 2 аспектів оптимізації вводиться відповідна вихідна інформація, створюється матриця С, визначається початковий розподіл файлів, перевіряються обмеження, файли перерозподіляються, якщо необхідно, і знаходиться раціональна матриця розподілу X.

Висновки до розділу 2

1. Було описано оптимізацію розподілу реєстрацій по вузлам локальної комп'ютерної мережі на основі чисельного критерію якості обслуговування Досліджено тенденції розвитку сучасних СБД. Показано, що при проектуванні СБД найважливішим завданням є вибір таких системотехнічних, алгоритмічних та програмних рішень, які дозволили б реалізувати систему апаратно-програмної підтримки операцій над даними на основі перспективної мікропроцесорної техніки та методів сучасних мов програмування високого рівня.

2. Розглянуто метод оптимізації розподілених реєстрацій розміщення файлів у комп’ютерній мережі на основі часу очікування. Виявлена проблема створення комутаційних систем, призначених для аналізу стану мережі в кожний момент часу та оптимальне транспортування даних для пошуку оптимальних рішень. Реальне застосування знаходять лише пакети прикладних програм, а саме Matlab

3. Досліджено мережу із топологією єдиної шини. Описано рішення задачі раціонального розміщення файлів інформації по вузлах локальної мережі .

3 СПЕЦІАЛІЗОВАНІ ОБЧИСЛЮВАЛЬНІ СТРУКТУРИ ДЛЯ ОПТИМІЗАЦІЇ РОЗПОДІЛЕНИХ РЕЄСТРАЦІЙ І ОБРОБКИ ДАНИХ ПО ВУЗЛАХ КОМП'ЮТЕРНОЇ МЕРЕЖІ

3.1 Організація розподіленої комп'ютерної мережі для розподілених реєстрацій розміщення файлів

Для прискорення обробки бази даних застосовують запитовий розподіл до БД кількома великими серверами, які з'єднані між собою комп'ютерною мережею [7, 19 – 21, 25]. Головне завдання систем управління розподіленими базами даних полягає у забезпеченні засобу інтеграції локальних інформаційних баз, які розташовуються в деяких вузлах обчислювальної мережі, так, щоб користувач, який працює в будь-якому мережному вузлі, мав доступ до всіх баз даних, як до загальної бази даних. Причому повинні забезпечуватися елементарність системного використання, можливості автономної роботи при порушеннях мережної зв'язності або за адміністративних потреб і високий рівень ефективності.

Для досягнення таких властивостей СКБД потрібний механізм розподілу запитів між серверами. Обробка запиту до БД здійснюється зверненням до кількох неподільних операцій, розкладання та послідовність яких залежить від конкретного запиту, а також план виконання, обраний оптимізатором запиту. Операції БД виконуються в суворій послідовності, причому вихід однієї операції є входом наступної. Для підтримки паралельної обробки запитів стосовно кількох серверів РБД вибирають неподільні операції, які можуть бути скопійовані і далі одночасно проводять обробку фрагментів даних і операції, результати яких можуть бути представлені так, ніби єдиний ланцюжок обчислень виконав операцію. З цією метою запити розбивають та обробляють незалежно від способів комунікації. Паралельні обробки різних ділянок БД на серверах дозволяють СКБД ефективно використовувати багатовимірні таблиці та розбиття таблиць. При правильній реалізації паралельних обробок значно зростає ефективність роботи з БД та управління ресурсами [19].

Для реалізації алгоритмів оптимізації розміщення запитів на серверах

комп'ютерної мережі необхідні достовірні вихідні дані [21]. Для визначення інтенсивності звернень до різних таблиць БД використовуєтьсярезидентна програма "Аналізатор запитів". Програма перехоплює всі запити до бази даних, перерозподіляє їх для обробки вузлами RDB і реєструє дату, час, назву таблиці, довжину запиту, час до відповіді, довжину відповіді. За допомогою модуля Query Optimizer аналізується інформація та визначаються часові характеристики звернень до таблиць, виправляються спотворення в обробці найбільш часто використовуваних даних. Алгоритм, вбудований у програму Query Optimizer, є евристичним і характеризується такими властивостями: зазвичай знаходить хороші, але не обов’язково оптимальні рішення; його можна реалізувати швидше та легше, ніж будь-який відомий точний алгоритм. Перший етап алгоритму знаходить початковий розподіл файлів на основі статистики, зібраної програмою Query Analyzer. На другому етапі відбувається перерозподіл файлів. Другий етап складається з серії кроків, на кожному з яких один файл перерозподіляється з повного вузла таким чином, щоб наполовину зменшити навантаження на вузол. Виконання другого етапу алгоритму триває до тих пір, поки не буде знайдено розподіл, що дозволяє досягти максимальної швидкості обробки запиту.

У запропонованій структурі сервери РБД, які використовуються для обробки великих обсягів даних, об’єднані в єдиний кластер за допомогою високошвидкісної комп’ютерної мережі та програмованого активного комутатора (ПАК). Для управління комутатором до мережі підключається комп’ютер адміністратора (рис. 3.1).

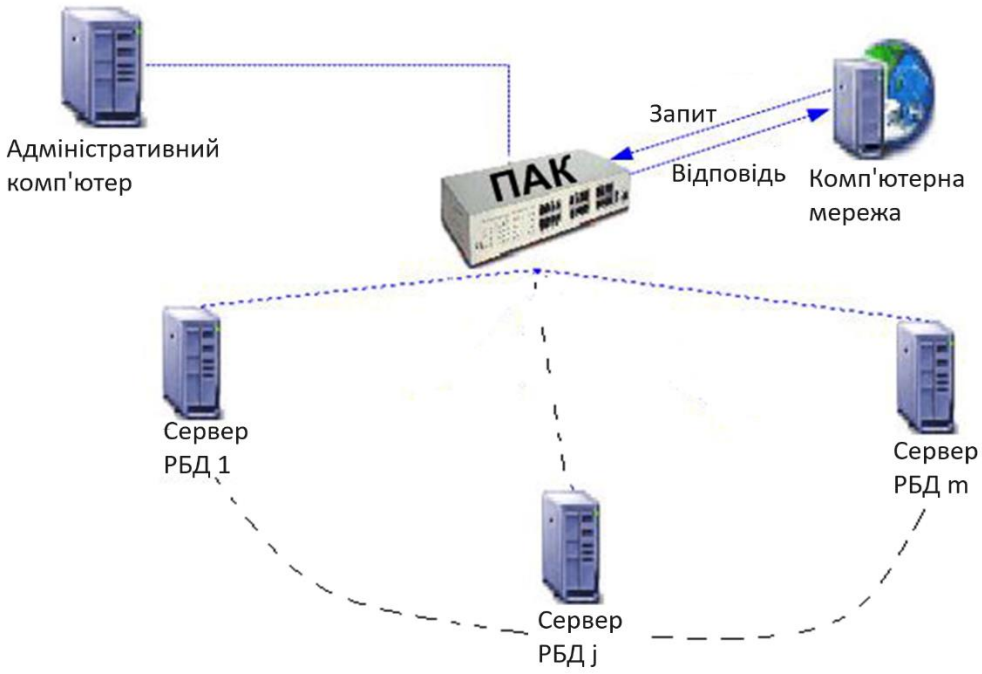


Рисунок 3.1 – Обчислювальна структура для оптимізації розміщення та обробки даних у вузлах комп’ютерної мережі

Архітектура ПАК пропонується для забезпечення балансування навантаження вузлів у мережі РБД, яка використовується для обробки великих обсягів даних. Програмований комутатор використовується для реалізації запропонованого методу та алгоритмів оптимізації розміщення запитів у вузлах комп’ютерної мережі [9, 14, 15]. Програмований комутатор орієнтований на роботу з великими обсягами даних за допомогою ПЛІС. Програмований комутатор надсилає запит на обробку даних, отриманий від користувачів РБД, на головний комп'ютер - перший етап обробки даних. Далі запит або обробляється на головному комп'ютері, якщо його обробка займає небагато часу, або відправляється через ПАК на один з допоміжних серверів. Основними функціями ПАК є управління обслуговуванням викликів, тобто встановлення та руйнування з’єднань із серверами, керування транспортними шлюзами та шлюзами доступу, координація обміну сигнальними повідомленнями між серверами чи іншим активним мережевим обладнанням, тобто підтримка функцій SG (Signaling Gateway). Комутатор координує дії допоміжних серверів і забезпечує зв'язок з комутаційним

обладнанням мереж, до яких підключені клієнти РБД. Крім того, у разі взаємодії між різнорідними мережами ПАК перетворює пакети із запитами на вибірку даних і відповіді на запити.

Перевага запропонованої структури перед класичними, що мають високу вартість і в загальній структурі поєднують комутаційні функції, управлінські функції обслуговуванням викликів, послуги та додатки, являють собою монолітну, закриту структуру системи, яка, в більшості випадків, не допускає розширення або модернізації на базі обладнання інших виробників, оскільки запропонована архітектура застосовує принципи компонентного мережного побудови та відкритих типових інтерфейсів між трьома основними функціями: комутації, керівництво обслуговуванням викликів, сервісу та додатків. У цій відкритій розподіленій структурі можуть використовуватися функціональні елементи різних виробників. Комутатор містить процесор мережі, шістдесят підмодулів, комутуючу матрицю, програмований прикладний пристрій (ППП) та пристрій інтерфейсу мережі (ПІМ).

3.2 Модулі програмованого комутатора для розподілу запитів у розподіленій базі даних

Для управління розподілом запитів між серверами РБД запропоновано використання ПАК, структурну схему якого показано на рис. 3.2. Комутатор базується на ПЛІС і, залежно від завантаженого в нього, дозволяє здійснювати обробку пакетів, що мають різні властивості і призначення відповідно до певних алгоритмів. Як правило, комутатори, які використовують виключно ПЗ, мають меншу продуктивність, ніж апаратні, але комутатори, що використовують ПЛІС, можуть забезпечити бажаний баланс між продуктивністю та використанням нестандартних алгоритмів обробки пакетів [35, 36]. Використання таких комутаторів дозволяє здійснювати швидке переналаштування системи та швидку зміну ПЗ [29].

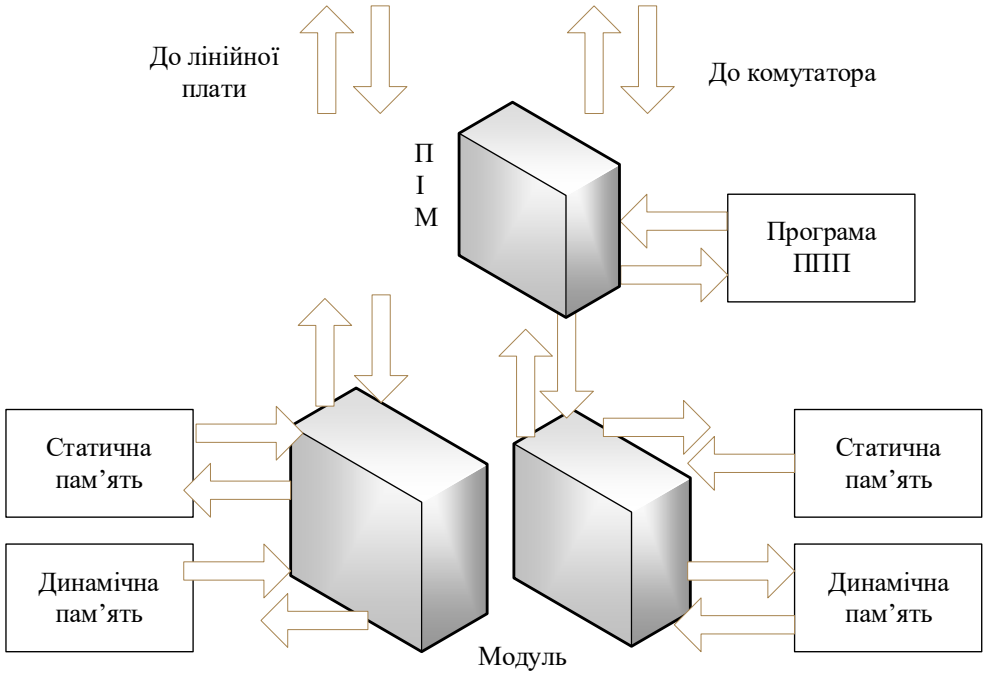


Рисунок 3.2 – Структурна схема запропонованого програмованого комутатора

Логіка ПАК базується на двох програмованих вентильних матрицях. Використання ПАК дозволяє досягти пропускної спроможності комутатора кілька Гбайт сек. ПАК складається з двох модулів, що базуються на ПЛІС Xilinx Virtex FPGA: XCV600Efg676 та XCV100Efg680. Мікропрограми та дані ПАК зберігаються у Flash та SRAM пам'яті. У першому модулі – в універсальному мережевому інтерфейсі (УМІ) здійснюється основна функція ПАК – перерозподіл пакетів із запитами. УМІ містить у собі логіку для отримання та передачі пакетів по мережі, маршрути індивідуальних потоків даних, прийому та динамічного перепрограмування ПАК відповідно до зібраної та обробленої статистичної інформації. Елементи, з яких складається УСІ, мають повний контроль над чотирма незалежними сховищами SRAM.

Пам'ять використовується виконання програм, використовують матрицю оптимального розподілу залежно від типу запиту. Другий модуль - програмований прикладний пристрій (ППП) служить для управління першим модулем відповідно до алгоритму оптимізації розподілу запитів по серверах мережі та даних, отриманих при аналізі попередніх запитів.

Структурна схема УМІ показано на рис. 3.3. УМІ контролює правильність

потоків даних між модулями ПАК і забезпечує механізм динамічного завантаження прошивки в комутатор по мережі. Інтерфейс містить чотирипортовий комутатор, який служить для передачі даних між серверами комп'ютерної мережі. Кожен порт комутатора пов'язаний з віртуальною схемою таблиць пошуку (VC) для вибору та комутації потоків. У кожному з чотирьох мережевих інтерфейсів УМІ є невеликий буфер, щоб уникнути короткочасних перевантажень. У випадку, коли кілька пакетів із запитами відправляються на один порт, УМІ перерозподіляє запити в залежності від зайнятості інших інтерфейсів і таблиць розподілу, вбудованих в РАС. У разі тривалого перевантаження інтерфейсу УМІ надсилає сигнал призупинення на попередній модуль або мережевий інтерфейс. УМІ підтримує пересилання як потоків даних, так і одиночних пакетів. Для налагодження модуля УМІ містить з’єднувач налагодження (ЕС), за допомогою якого можна тестувати та перепрограмувати РАС.

УМІ включає Control Cell Processor (CCP), призначений для управління роботою ПАК та зв'язку по мережі. CCP обробляє команди модифікація потокової маршрутизації вхідних даних, команди читання та запису даних про стан пристрою, команди читання та запису конфігурації пам'яті. Разом з ПАК є можливість використовувати мережевий вузол управління (ВУ) на базі процесора Pentium. У конфігурацію вузла входить чіп маршрутизації APIC, що керує ядром NetBSD. Зв'язок між ПАК і УУ здійснюється спеціальними пакетами, які розпізнаються інтерфейсами ПАК і передаються в модуль ППП. Об'єднавши ВУ з ПАК, можна досягти одночасного аналізу запитів, обробки статистичної інформації та введення цієї інформації в таблиці ПАК.

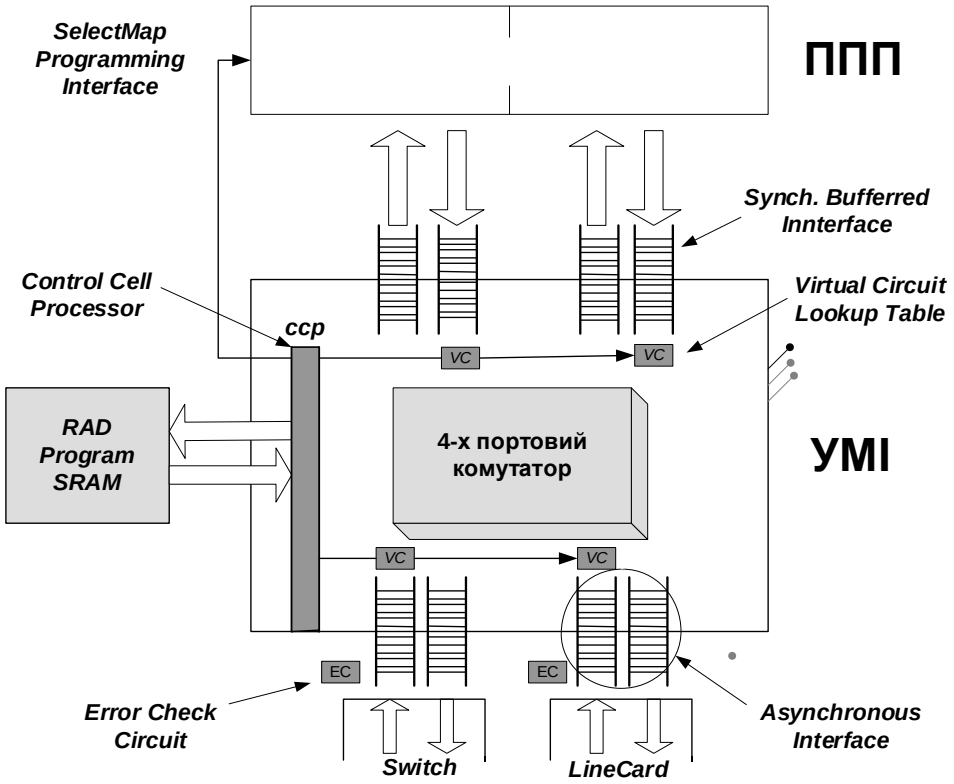


Рисунок 3.3 – Структурна схема універсального мережевого інтерфейсу

Запропоновано використовувати РБД та механізм оптимізації розподілу запитів по серверах мережі, в інформаційній системі реєстрації та обробки документів, що використовується на авіаційних підприємствах [9]. Система дозволяє інтегрувати інформацію з кількох інформаційних систем, що працюють в авіакомпанії, та створювати на основі цієї інформації системи реєстрації фінансових документів. Документи, що містяться в цій системі, можна аналізувати, створювати між ними зв'язки та реалізувати на основі цих документів звіти довільної структури.

Вхідні потоки інформації надходять до системи у вигляді бухгалтерських документів, банківських звітів, інформації про переміщення коштів та ін. Після введення та збереження у системі первинних бухгалтерських документів система автоматично формує бухгалтерські проводки. Це дозволяє передавати інформацію бухгалтерсько-аналітичній системі, формувати різноманітні оборотно-сальдові відомості, реєстри, журнали-ордери та ін.

З метою прискорення обробки БД великих обсягів запропоновано програмну

та апаратну реалізацію методу розподілу запитів по серверах РБД. Результати тестування інформаційної системи реєстрації та обробки документів свідчать про можливість значного збільшення швидкості обробки даних великого обсягу за рахунок використання РБД та механізмів оптимізації завантаження серверів, що використовуються у РБД.

3.3 Використання програмованого комутатора для розподілу реєстрацій

Програмований комутатор запропонований при розробці інтерфейсу між серверами РБД та host-комп'ютером, який займається розподілом запитів до великих та наддовших масивів інформації.

Основні функції ПАК:

1) управління обслуговуванням викликів, тобто встановлення та розірвання з'єднання між серверами;

2) керування транспортними шлюзами та шлюзами доступу за протоколом H.248 та подібними;

3) організація обміну сигнальними повідомленнями між мережами, тобто підтримка функцій SG (Signaling Gateway). ПАК організовує дії, які забезпечують з’єднання з логічними мережами в різних мережах, і перетворює дані в повідомленнях таким чином, щоб вони були зрозумілі з двох сторін різнорідних мереж.

Дорогі класичні комутатори поєднують у загальній структурі опції комутації, функції керування дзвінками, сервіси та програми, а також функції білінгу. Ця архітектура є монолітною замкнутою структурою системи, яка зазвичай не допускає розширення або модернізації на основі обладнання інших виробників.

Пропонований ПАК застосовує принципи побудови компонентів комп’ютерної мережі та відкритих класичних інтерфейсів між 3 основними функціями: комутація, керування викликами, обслуговування та програми. Ця відкрита розподілена структура може використовувати функціональні елементи від різних виробників (рис. 3.4).

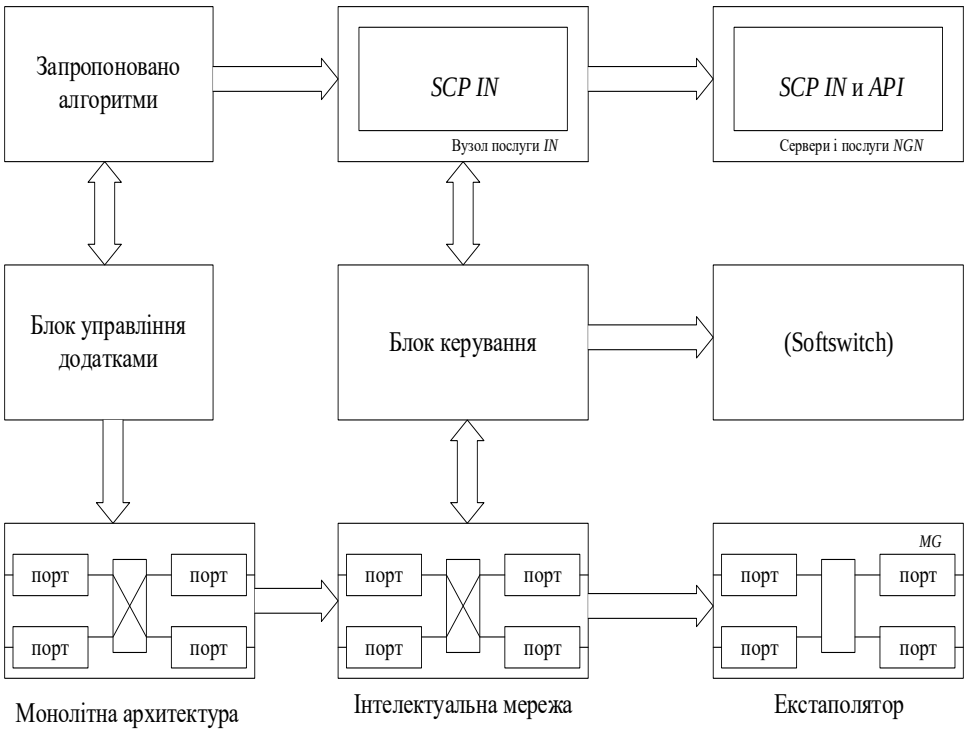


Рисунок 3.4 – Структурна схема програмованого перемикача

Відповідно до архітектури стандарту ПАК він передбачає 4 функціональні області, показані на рисунку 3.5:

- транспортна;
- керівництва обслуговуванням виклику та сигналізації;
- сервісу та додатків;
- управління експлуатацією.

Комутаторне програмування. Оновлення містить у собі такі кроки:

1. Оновлення програмного забезпечення контролера MSC-1000 – 350ds*C*.bin.

2. Оновлення зовнішнього вигляду контролера MSC-1000 – 350ds*C*.rom.

3. Оновлення програмного забезпечення карток ліній ALC-1024 – 204DV*C*.bin.

4. Оновлення зовнішнього вигляду карток ліній ALC-1024 – 204DV*C*.cfg.

Рисунок 3.5 – Схема керування даними з використанням запропонованого комутатора

Оновлення програмного забезпечення контролера виконується таким чином:

– У BootP вкладення Normal Boot.

– *NEW*:

MAC-адреса (параметри xx:xx:xx:xx:xx:xx);

HOST IP– IP адреса MSC-1000;

Server IP– адреса комп'ютера, на якій завантажено BootP;

NetMask– маска мережі;

FileName– повна назва файлу з програмним кодом (наприклад, C:TMP350ds*C*.bin).

– *Update Database*.

3.4 Використання екстраполятора у структурі програмованого комутатора для розподілу реєстрацій

Запропоновано обчислювальну структуру екстраполятора [16], що дозволяє прогнозувати інтенсивність інформаційних потоків у комп'ютерній мережі.

Екстраполятор містить блок 1 спостереження, вхід якого з'єднується з каналом передачі інформації. На вхід блоку спостереження також подається

значення періоду спостереження t_{nc}, протягом якого в блоці здійснюється усереднення інтенсивності Таким чином, блок формує послідовність моментів вибірки, що є середніми на проміжках спостереження, та відповідних усереднених значень інтенсивності. Блок містить таймер, що синхронізує процес вибірки. Інформація про моменти часу та усереднені інтенсивності з боку спостережень надходить на блоки 2 та 3, що здійснюють відповідно обчислення допоміжної функції та її градієнта за коефіцієнтами полінома. Обчислення цими блоками ведуться паралельно прискорення розрахунків. Виходи блоків 2 та 3 з'єднані з блоком 4, який виконує функцію чисельного інтегрування рівнянь налаштування для коефіцієнтів полінома. На окремі входи блоку 4 подається показник загасання допоміжної функції значення порядку полінома. На виході блоку 4 одержуємо коефіцієнти полінома. Вони разом із заданим часом екстраполяції t_e подаються на блок 5, який здійснює обчислення екстраполірованого значення інтенсивності, що є вихідною величиною екстраполятора.

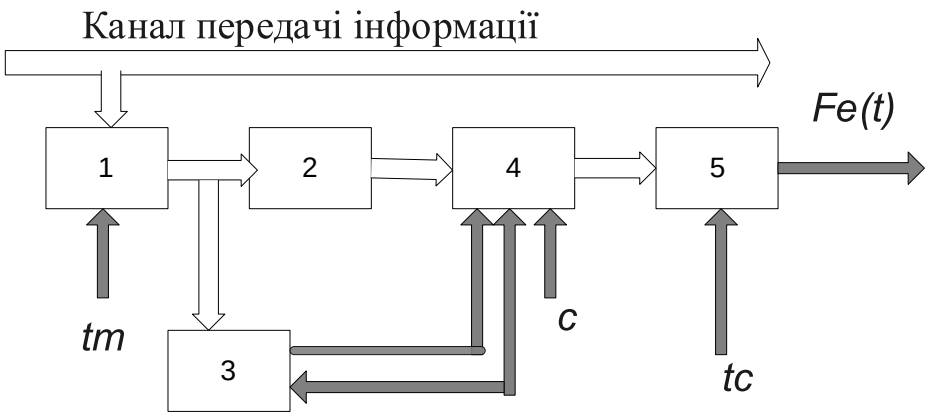


Рисунок 3.6 – Обчислювальна структура запропонованого екстраполятора

Запропонована обчислювальна структура екстраполятора має перевагу над іншими, що полягає в тому, що на виході екстраполіроване значення постійно змінюється, і за умови досить великого показника значення допоміжної функції воно близьке до точного значення. У перехідних процесах, коли інтенсивність у мережі сильно пульсує, екстраполятор дає похибку, зумовлену обмеженістю його часткових характеристик. Але й за несприятливих умов він забезпечує відтворення

тенденції зміни інтенсивності обміну в мережі.

Поліноміальний екстраполятор призначений для прогнозування інтенсивності інформаційних потоків у комп'ютерних мережах. До складу екстраполятора вводяться блоки, які здійснюють безперервні рішення визначення коефіцієнтів апроксимуючого полінома.

Екстраполятор містить блок спостереження інтенсивності інформаційного потоку в точках за годиною із заданою дискретністю. На його вхід інформація надходить безпосередньо з обмінної шини інформацією між елементами мережі. На виході блоку спостереження утворюється послідовність пар значень години та інтенсивності обміну інформацією. Кількість точок спостереження залежить від ступеня поліному, наприклад, для третього ступеня кількість не може бути меншою за чотири. Послідовність m пар (моментів спостереження та інтенсивностей) позначимо:

$$(f_1, f_1), (f_2, f_2), \dots, (f_m, f_m). \quad (3.1)$$

Послідовність (3.1) оновлює із заданою дискретністю у часі. При додаванні до неї чергової пари спостережень вона стає в послідовності останньої, а перша пара втрачається. Для екстраполяції інтенсивність апроксимується у вигляді полінома в ступені n:

$$f(t) = a_{n-1}t^{n-1} + \dots + a_1 t + a_0. \quad (3.2)$$

Для обчислення коефіцієнтів поліному (3.2) утворюється допоміжна функція результатів спостережень:

$$V = \sum_{j=1}^{n}(\sum_{i=0}^{m} a_i\, t_j^i - f_j)^2. \quad (3.3)$$

Мінімум функції (3.3) за коефіцієнтами полінома (3.2) відповідає апроксимації за критерієм мінімуму суми квадратів відхилень результатів спостережень (3.1) від значень за виразом (3.2). Функція (3.3) обчислюється окремим обчислювальним блоком, на вхід якого надходять результати

спостережень, але в виході виходить значення допоміжної функції. Екстраполятор також містить блок обчислення часткових похідних допоміжної функції

$$\frac{\partial V}{\partial a_i} = 2\sum_{j=1}^{m}(\sum_{i=0}^{n} a_i t_j^i - f_i)t_j^i, \quad i = 0, \dots, n. \tag{3.4}$$

На вхід цього блоку надходять результати спостережень, але в виході утворюється вектор часткових похідних – градієнт допоміжної функції. Градієнт (3.4) допоміжної функції використовується для побудови диференційних рівнянь налаштування коефіцієнтів полінома (3.2). Рівняння налаштування мають вигляд:

$$\dot{a}_i = k\frac{\partial V}{\partial a_i}, \quad i = 0, \dots, n. \tag{3.5}$$

Відповідно до (3.5) коефіцієнт змінюється у часі у напрямку антиградієнта допоміжної функції k визначається за умови заданої швидкості загасання допоміжної функції (3.3). Для цього її поведінка підкоряється допоміжному асимптотичному стійкому диференційному рівнянню:

$$\dot{V} + cV = 0. \tag{3.6}$$

У рівнянні (3.6) визначається коефіцієнт с додатковий.

Розглядаючи в рівнянні (3.6) похідну від допоміжної функції по годині через рівняння налаштування, отримаємо вираз для коефіцієнта k:

$$k = \frac{\frac{c}{n+1}V}{\left(\frac{\partial V}{\partial a_i}\right)^2}. \tag{3.7}$$

Підстановка (3.8) (3.6) дає остаточний вираз для рівнянь налаштування:

$$\dot{a}_i = \frac{\frac{c}{n+1}V}{\frac{\partial V}{\partial a_i}}. \tag{3.8}$$

Рівняння (3.8) реалізується окремим блоком, на вхід якого надходять допоміжна функція, її градієнт, показник загасання допоміжної функції η, Що дорівнює апроксимуючого полінома. Відповідно до виразу (3.8) процес

апроксимації здійснюється в реальному часі за допомогою чисельного рівняння налаштування. Таким чином, результат екстраполяції виходить безперервно і тому може використовуватися без затримки в темпі його отримання. В результаті коефіцієнти налаштування змінюються згідно з даними спостережень. Екстраполятор також містить блок обчислення екстраполюваних значень інтенсивностей, на вихід якого подаються поточні значення коефіцієнтів та значення часу, що відповідає заданому моменту екстраполяції. Екстрапольоване значення інтенсивності є вихідною інформацією екстраполятора.

Висновки до розділу 3

1. Виконано дослідження сучасного активного мережевого обладнання, яке може бути використане для побудови РБД, досліджено продуктивність, пропускну здатність портів, сумісність із сучасними серверними системами, вартість. Запропоновано базові структури РБД, що дозволяють досягти найбільшої продуктивності при обробці великих БД при розподілених реєстраціях.

2. Запропоновано комутатор для керування потоками даних відповідно до запропонованих алгоритмів балансування завантаження вузлів мережі РБД, що дозволяють суттєво зменшити дисбаланс на серверах РБД при розподілених реєстраціях.

3. Запропоновано способи розподілу функцій між керуючим комп'ютером, комутатором та серверами РБД при виконанні запитів до даних, які розглядаються як послідовні стадії виконання запиту на СБД. Розроблено систему, що розподіляє навантаження в СБД, залежно від якої обробка невеликих запитів відбувається на комп'ютері, що управляє, а запитів, пов'язаних з великим обсягом обчислень (сортування, перетин та ін.) – на серверах РБД.

4. Розроблено обчислювальну структуру поліноміального екстраполятора, призначеного для прогнозування інтенсивностей інформаційних потоків у комп'ютерних мережах при розподілених реєстраціях.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДУ ОПТИМІЗАЦІЇ РОЗПОДІЛЕНИХ РЕЄСТРАЦІЙ ПІДКЛЮЧЕННЯ РОЗМІЩЕННЯ ТА ОБРОБКИ ДАНИХ У СИСТЕМІ УПРАВЛІННЯ РОЗПОДІЛЕНИХ БАЗ ДАНИХ

4.1 Створення програмних засобів для управління розподілом реєстрацій у локальній мережі з урахуванням безпеки підключення

З метою встановлення інтенсивностей звернень до різних файлів інформаційних баз написано резидентську програму "Аналізатор запитів". Як мову програмування був використаний Асемблер, що надає необхідну компактність та пристосованість при написанні резидентських програм і не вносить помилок у процеси, що вимірюються. Резидентську програму було завантажено на всі мережеві вузли. Час усіх машин було синхронізовано. Протягом 1-го робочого дня накопичувалися дані.

Програма ловить всі звернення до файлів і записує до журналу дату, час, найменування файлу, тривалість запиту, час до відповіді, тривалість відповіді. Протоколирование виконується у внутрішній буфер і скидається на диск лише під час " неодруженого " комп'ютерного ходу, тобто під час здійснення введення з клавіатури чи інших операцій, потребують очікування відповіді.

За допомогою нерезидентської частини програми дані було проаналізовано та знайдено тимчасові коляції звернень до файлів. Мережеве завантаження виявилося переривчастим, спостерігалася і повна відсутність звернень, і одноразове звернення з усіх станцій, що функціонують.

Для перерозподілу запитів пошуку до інформаційних довідкових файлів між мережними вузлами визначено програму «Оптимізатор запитів у локальній мережі». Ця програма блокує утворення черг, які при досягненні інтенсивності запитів конкретного порога дуже "гальмують" мережеву роботу.

Для роботи програмного забезпечення необхідно підготувати каталог розташування файлів inquire.txt (від англ. inquire - справлятися). Для комфортної роботи в мережі необхідно мати можливість за бажанням змінювати список файлів каталогу та їх адреси в локальній мережі. Для забезпечення цієї можливості під час

роботи з програмою «Оптимізатор запитів локальної мережі» планується отримувати дані конфігурації мережі з файлу inquire.txt. Цей файл зберігає дані про всі файли каталогу та їх розташування. Формат створеної позначки у файлі inquire.txt виглядає так:

Номер мережі, номер вузла, пристрій [шлях]\назва файлу

Перетворення у файлі inquire.txt вимагають програмного перезавантаження.

Програма "Оптимізатор запитів" включає три частини: інсталяція, перехоплення запитів, обробка запитів.

Розглянемо алгоритм роботи 1 частини - інсталяції.

1. Перевірка наявності її у ВП. Якщо в оперативній пам'яті вона не була завантажена раніше, тоді триває наступна установка. В іншому випадку програма завершує роботу.

2. Відкриття сокетів роботи. Якщо сокети вже відкриті іншою програмою, або відкриття не вдалося з технічних причин, то з'являється повідомлення про це і подальша установка програми припиняється.

3. Заповнення внутрішньої таблиці розташування інформаційних файлів з inquire.txt. Якщо довідково-інформаційний файл inquire.txt не знайдено або записи в ньому не відповідають необхідному формату, то надсилається повідомлення та встановлення програмного забезпечення припиняється.

4. Отримання мережевої адреси цього вузла.

5. Отримання векторних адрес застосовуваних переривань.

6. Перевстановлення векторів переривань на вашому власному обробнику.

7. Вихід з програми та збереження її резидентної частини в ЕП.

Під час встановлення програма може надсилати такі повідомлення:

«Неможливо відкрити сокет» – внаслідок технічних неполадок сокети, що використовуються програмою під час роботи, не можуть бути відкриті. У такому разі необхідно перезавантажити комп’ютер і знову встановити програму.

«Сокети вже відкриті» – повідомлення з’являється, коли сокети, які використовує програма для обміну пакетами, відкриті іншою програмою. У цьому випадку потрібно перезавантажити комп'ютер і знову встановити драйвер.

«Файл inquire.txt не знайдено» – це повідомлення означає, що в поточному каталозі відсутній довідково-інформаційний файл inquire.txt. Вам потрібно скопіювати цей файл у поточний каталог або створити його знову.

«Недійсний формат прапора у файлі incuire.txt» – це повідомлення вказує на те, що формат у файлі inquire.txt не дотримується. Потрібно перевірити файловий вміст та оформити його за необхідним форматом.

Відсутність повідомлень означає, що інсталяція програми пройшла успішно.

Ця програма повинна бути встановлена на всіх мережевих вузлах, які будуть використовувати перерозподіл запитів. Тому для повсякденного використання може мати сенс включити програмне завантаження у файл autoexec.bat або у файл мережевого завантаження.

Оскільки потрібно буде обробляти запити на відкриття/закриття та читання/запис файлу, резидентна частина програми повинна обробляти такі функції:

- 3Dh – відкриття файлу;
- 3Eh - закриття файлу;
- 3Fh - читання з файлу;
- 40h – запис у файл.

Крім цього, з метою захисту програми від вторинного завантаження в оперативну пам'ять перехоплюється незастосована функція переривання 21H - FFH.

Розберемо робочий алгоритм 2-ї програмної частини, перехоплення запитів.

1. Перехоплення орієнтиру переривання 21H.
2. Якщо № функції = FFH і № підфункції = 66H, тоді регістр AX вноситься значення коду 6677H і вихід з переривання.
3. Якщо номер функції не є 3DH, або 3FH, або 3EH, або 40H, то переривання передається успадкованому обробнику. В іншому випадку обробник заповнює пакет протоколу IPX із запитом на обробку процесу на деяких вузлах мережі, надсилає повідомлення по мережі та активує режим очікування відповіді на запит. Після отримання відповіді програма заповнює регістри прапорів індикаторами AX,

які надійшли в пакеті, і передає управління програмі, яка викликала переривання.

Щоб обробити мережевий запит на доступ до файлу швидко, безпечно і непомітно для користувача, було вирішено використовувати перепризначення наступних векторів переривань:

- 28h – тимчасовий квант DOS;
- 13h – дискове обслуговування BIOS;
- 1Ch - переривання користувачів за таймером;
- 21h - функції сервісів DOS.

Вибір безпосередньо цих функцій базується на їх дуже частому використанні (1Ch, 13h) і спеціальному призначенні (28h, 21h).

Алгоритми обробників, які перехоплюються програмою переривання, вважаються ідентичними, тому наведено приклад лише обробника переривання 28H. Відмінною рисою цього обробника від інших є те, що на нього покладено функцію захисту програми від повторного завантаження. Блок, який не входить до інших обробників, виділено курсивом.

1. Перехоплення орієнтиру переривання 28H.

2. Якщо функція № =*FFH*, і підфункція № =66H, потім в регістр AX заноситься значення коду 6677H і відбувається вихід переривання.

3. Якщо пакет запиту не надійшов, то переривання передається застарілому обробнику. В іншому випадку обробник заповнює регістри індикаторами, які надійшли в пакеті протоколу IPX, і виконує переривання 21H відповідно до станів регістрів. Потім програма заповнює пакет індикаторами регістрів прапорів зі значеннями AX, заповнює пакет необхідними даними і викликає старий обробник.

4.2 Оптимізація розподілу реєстрацій з прикладу підсистеми «Облік доходів» підприємства

Для визначення інтенсивності доступу до різних файлів інформаційної бази використовувалася резидентна програма «Аналізатор запитів», написана мовою програмування Assembler. Мова гарантує компактність і гнучкість при написанні резидентних програм і не допускає помилок у вимірюваних процесах. Резидентна

програма завантажується на всіх вузлах мережі. Час комп’ютера синхронізовано.

Програма аналізує всі запити до файлів і реєструє дату, час, назву файлу, тривалість запиту, час відповіді, тривалість відповіді.

Логування ведеться у внутрішньому буфері і записується на диск тільки під час простою комп'ютера, коли виконується набір тексту з клавіатури або інші операції, що вимагають очікування відповіді.

За допомогою нерезидентної частини програми проаналізовано оброблені дані та знайдено тимчасові властивості доступу до файлів. Завантаження мережі характеризується нерівномірністю - повна відсутність дзвінків або одночасний дзвінок з усіх робочих станцій.

Для перерозподілу пошукових запитів до довідково-інформаційних файлів між вузлами мережі призначена програма «Оптимізатор запитів локальної мережі», яка значно скорочує утворення черг.

Як приклад розглянемо базу даних авіакомпанії, а саме її систему обліку доходів. Система «Облік доходів» працює з базами даних великого обсягу та встановлена на локальному сегменті мережі, що складається з 12 вузлів. Розбивка бази даних на інформаційні файли наведена в табл. 4.1.

Система автоматично обробляє фінансові документи про доходи та витрати авіакомпанії. До прибуткових документів належать рахунки-фактури, їх оплата, використані авіаквитки та банківські операції. До видаткових документів враховуються рахунки-фактури постачальників, виписки з банківського рахунку, відомості відділів громадського харчування, подорожні листи та відомості про заробітну плату. Під час перегляду таблиць бази даних (квитки, організації, транзакції) існують такі параметри:

– за допомогою універсального фільтра (будь-якого зразка);

– зміна порядку відображення записів (сортування);

– друк поточної відфільтрованої таблиці із зазначенням певного порядку відображення друкованих стовпців, зміною їх ширини та обчисленням сум за числовими стовпцями;

– видалення або заміна відфільтрованих даних;

– перетворення таблиці на картку для більш наочного перегляду одного запису таблиці.

Таблиця 4.1 - Розділ файлу бази даних «Облік доходів»

Ім'я файлу	Дані	Розмір, байт	Середовище. час пошуку, мс.	Інтенсивність запитів на пошук і виправлення
KLIENT	Клієнти	6 764 580	45	4; 0.5
R_SCHET	Облікові записи	1 701 160	42	4; 0.6
IMP_L	Проводки	10544230	62	10; 5
IMP_O	Залишки їжі	16 146	3	0,1; 0.1
CONFIG	Каталог груп	99 737	3	1,5; 0.01
IMP_N	Податки	15 144	2	10; 0.01
BANK	Банки	12 730	2	0,1; 0.01
REZERV	Журнал бронювання	9,046	2	0,1; 0.01
AKT	Акти	3,083	2	0,1; 0.01
KLIENT1	Співробітники та відділи	2 386	1	0,2; 0,02
KLIENT2	Клієнти авіакомпанії	5 231	1	0,2; 0,02
KLIENT3	Відстрочки для клієнтів по платежах	9,552	1	0,2; 0,02
KLIENT4	Нестандартний розподіл	940	1	0,2; 0,02
AKT_POL	Погашення штрафів	1 486	1	0,1; 0.01
BILETY	Зображення квитків і інформація про рейси	37028361	114	0,1; 0.01

Система «Облік доходів» має такі режими роботи:

– введення платіжних документів;

– уточнення квитанцій;

– залік переплат;

– взаєморозрахунки між клієнтами;

– формування та друк документів;

– автоматичне формування оперативних документів з дебіторської та кредиторської заборгованостей з контролем строків виплат.

Крім того, система надає користувачеві додаткові можливості:

– робота з бланками квитків, не виходячи з програми, перегляд і друк зображень квитків, що надійшли та зберігаються у спеціальній базі, видача звіту щодо завантаження різних рейсів авіакомпанії;

– контекстно-залежна допомога;

– контроль за роботою бортпровідників та пілотів за допомогою, що передається з відділу відряджень інформації;

– заключні обороти;

– друк на бланках IATA, формування описів, реєстрів тощо;

– перевірка цілісності інформації та автоматичне усунення порушень;

– реконфігурація системи за бажанням користувача або при виникненні змін у законодавстві;

– гнучка система паролів для розмежування доступу, криптографічне закриття інформації.

Робота доходного відділу авіакомпанії полягає у досить складній схемі. Четверо співробітників на початку робочого дня вносять платежі протягом трьох-чотирьох годин, далі протягом одного-двох годин суми, що надійшли розподіляються між бюджетами, та створюється головна звітність. Отже, час проведення операцій дуже критично. У цьому інші співробітники зайняті некритичним у часі створенням звітів, запровадженням квитків тощо. Оброблювані дані передаються до банку, і створюються звіти керівників компанії.

Великі обсяги даних можуть істотно знизити робочу швидкість програмного пакета при недоцільному розподілі файлів інформації. Для розподілу файлів інформації РБД "Прибутковий облік" застосовувалась програма opt1.

Як аспект оптимальності прийнято єдиний час, необхідний обслуговування всіх запитів, які надійшли до системи за 1-цу часу.

Мережеві користувачі працюють з інформацією, що зберігається у п'ятнадцяти файлах даних баз даних. При створенні масивів вихідної інформації враховують оцінку інтенсивностей виправляючих та пошукових запитів λ_i, λ_i' та середнього пошукового часу у файлах t_{ij}, що отримані за допомогою резидентської

програми "Аналізатор запитів" [7, 9].

Оперативність запитів на виправлення та пошук у різних файлах даних РБД "Облік доходів" та середнього пошукового часу представлені в таблиці.

Складено файл вихідної інформації opt1.dat для програми оптимізування opt1 у наступній послідовності: оперативність пошуків та виправлень у будь-якому файлі РБД; тривалість записів у файлах та середній період обробки запитів пошуку до файлів даних.

Як аспект оптимальності взятий середній обсяг інформації, що пересилається, по лініях зв'язку при обробці запитів. Отримана матриця доцільного розподілу файлів по мережевим вузлам в результаті розрахунку за програмою оптимізування opt1 має такий вигляд:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Так, файли, в які були внесені зміни, досить часто зберігаються по 1-й копії. Копії інших файлів дублюються у локальних базах мережевих вузлів.

Якщо як аспект оптимальності прийняти єдиний час, необхідне обслуговування всіх запитів, які надходять до системи протягом 1-ци часу, матриця доцільного розподілу файлів іншого виду:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Так, перші 4 файли інформаційних баз, інтенсивності виправлень в яких досить великі, зберігаються на 1-му з найбільш продуктивних вузлів (по 1-й копії). Копії інших файлів, коригування та додавання в яких ведуться рідше, розташовані в локальних базах 1-х чотирьох комп'ютерних вузлів.

Насправді зручніше у разі використовувати другий спосіб розподілу файлів, т.к. для дуже обмеженої у часі роботи з запровадження платежів, розподілу між бюджетами сум, що надійшли та створення головної звітності достатньо застосувати чотири перші мережеві вузли з найвищою швидкістю. Аналогічне розміщення копій файлів даних дозволить уникнути зайвої інформаційної надмірності та труднощів щодо її узгодження.

Запропоновано практичну реалізацію методу балансування завантаження вузлів мережі РБД, призначеного для обробки БД великих і надвеликих обсягів. Результати роботи системи обліку доходів, що ґрунтується на запропонованому методі, свідчать про можливість значного збільшення швидкості обробки даних у РБД великого обсягу за рахунок використання механізмів оптимізації завантаження вузлів мережі, що використовуються для обробки РБД. Застосування методу дозволяє підвищити продуктивність та знизити час реакції інформаційної системи, що працює з РБД.

4.3 Метод оптимізації роботи SQL сервера під час роботи з великими базами даних із розподілу доступу

Розпаралелювання введення-виведення у поєднанні з внутрішнім розпаралелюванням СБД дозволяє здійснити одночасний доступ до фрагментованих таблиць та індексів, розташованих на кількох фізичних дисках [28-31, 47, 48]. При цьому продуктивність прикладної системи залежить від конструкції БД та написаних для неї запитів.

За зразок розглядалася база даних авіакомпаній [10]. У цій системі застосовується стандартне об'єднання двох таблиць: Archiv, що має сімсот п'ятдесят тисяч рядків і Employees, що містить близько двадцяти тисяч записів. Зв'язок між таблицями здійснюється за допомогою стовпця empId. Якщо ж таблиця Archiv не має індексу даного стовпця, тоді сервер відсканує таблицю Archiv двадцять тисяч разів. Виконання цього запиту займе багато часу, оскільки... він змусить сервер зробити понад сімдесят п'ять мільйонів логічних операцій. Формування індексу за стовпцем зменшить час обробки до секунд, а кількість логічних операцій введення-виведення приблизно до 750 тисяч [28]. У SQL сервері утиліти SHOWPLAN дає можливість переглянути запитний план і оптимізувати його [46].

Рядок SQL запиту має бути найбільш компактним. Великі показники довжини рядка можуть значно знизити операційний відсоток введення-виведення. Переповнення рядків змінної довжини більш ніж переповнення рядків фіксованої довжини. За допомогою утиліти DBCC SHOWCONTIG встановлюється середня густина сторінки, а також середня кількість вільних байтів. Таблиці, що мають середню кількість вільних байтів наближено до розміру рядка, описують ситуацію, коли негайно виконуватимуться операції введення-виведення.

Не дуже економне застосування пам'яті описує таблиці, в яких було виконано видалення максимальної кількості рядків, і немає кластеризованих індексів. У результаті видалень на сторінках утворюються порожні місця, оскільки SQL сервер неспроможна повторно застосувати простір сторінки, за відсутності

кластеризованого індексу, нові рядки цієї таблиці розташовані на останній сторінці. Табличні сторінки будуть заповнені менше, ніж на сто відсотків, що збільшить кількість операцій вводу-виводу.

Незаповнені місця в таблиці можуть утворитися не тільки при видаленні, а й при застосуванні функції оновлення UPDATE. Визначальним фактором вважається метод, яким сервер структурує операційні серії для запобігання порушенням обмежень цілісності. Наприклад, під час проведення операції UPDATE в таблиці, Archiv, де au_id вважається первинним (чи унікальним) ключем, оновлення 1-го рядка може порушити обмеження унікальності, якщо au_id – поступово зростаюча величина. У такому випадку сервер застосовує режим відкладеного оновлення, в якому операція оновлення проводиться в дві стадії: відразу видаляється старий рядок, а потім вводиться новий з необхідним показником. Сервер обробляє такий випадок, вводячи журнал неопераційні записи, де вказує, яку операцію потрібно виконати. Після виявлення всіх рядків та запису в операційний журнал видалення та введення сервер повертається до початку операції і приступає до проведення операцій з видалення. Після видалення сервер вводить рядки. Всі видалення та введення тепер являють собою закінчені операції, а тому супроводжуються модифікуванням всіх індексів, які торкаються.

Відкладені оновлення можуть істотно знизити продуктивність системи БД і додатки, т.к. вони призводять не тільки до великих витрат серверної пам'яті під записи операційного журналу, але й повільніше, ніж оновлення прямого режиму. Чим повільніше виконується оновлення, тим довше тривають виключні блокування, отже, іншим користувачам доводиться довго чекати на звільнення сторінки [41-44]. Імовірність появи тупикової ситуації зростає.

Операції поновлення проводяться різними способами [42, 43]. Найкращим за швидкодією є спосіб прямого оновлення заміщенням. При його застосуванні не виникає жодних переміщень, а в операційний журнал міститься один запис, який містить дані про байти, що отримали нові значення. Найповільнішим вважається метод відкладеного оновлення, який розглядався вище. Інші методи базуються на прямому оновленні, коли в операційний журнал не вноситься жодних зайвих

записів, а запис нових показників не проводиться на те саме місце, на якому була інформація, що оновлюється.

Пряме оновлення за підходом заміщення передбачає задоволення таких вимог: не можна оновлювати головні стовпці в кластеризованому індексі; таблицю не можна позначити реплікації. Зміни кластеризованого індексу змушують сервер переміщати рядок на нове місце, яке відповідає її змісту, що зазвичай супроводжується спочатку видаленням, а потім вставкою рядка. Під час реплікацій відбувається читання журналу та створення команд ODBC для користувачів. Тому поєднання видалення-вставка є більш простим відображенням операції оновлення. Обидві ситуації виключають оновлення безпосереднім заміщенням.

Подібні правила застосовуються до стовпців непостійної довжини та стовпців, які містять невизначені показники. При оновленнях, які стосуються багато рядків, стовпець повинен мати фіксовану протяжність, для допуску заміщення старого показника на новий. Сервер зберігає стовпець з невстановленими показниками, як стовпець непостійної протяжності, навіть коли програміст визначив його як стовпець з фіксованою довжиною. Для оновлення багатьох рядків стовпця з невстановленими показниками сервер використовує відкладене оновлення.

Значних результатів досягається за допомогою обліку обмежень при конфігурації БД, особливо коли потрібно використовувати всі можливості збільшення пропускної спроможності при оновленнях. Використання методів, які забезпечують пряме заміщення, дає змогу заощадити на введенні-виводі при записуванні в журнал, при читанні логічних журнальних сторінок, при резервуванні та поновленні журналу, а також при відновленні баз даних. При проектуванні інформаційних баз необхідно дотримуватися стандарту, відповідно до якого варто використовувати лише стовпці фіксованої протяжності, які не містять невстановлених значень.

Легкий спосіб знизити обсяг введення-виведення, необхідного для обробки запиту – зменшити кількість рядків, які повинен досліджувати SQL сервер, що забезпечується за допомогою вибіркових аспектів пошуку у виразі WHERE, що

входить до структури запиту [42]. Ці аспекти (аргументи пошуку) надають допомогу оптимізатору запитів порадами щодо того, який підхід доступу до інформації виявиться найшвидшим. Аргументи пошуку є таким записом:

Назва Стовпця оператор [Найменування Стовпця чи константа]

де оператором може бути 1 з таких порівняльних знаків =, <, >, <=, <=. Аргументи пошуку можуть бути об'єднані булевським оператором AND. Вираз BETWEEN ..... AND теж допустимий, так як умова, що їм задається, можна сформувати по-іншому за допомогою операторів >= і <=. Нижче наведено ряд прикладів пошукових аргументів:

Client= ....

Client>= .....

OrderDate.....

Тут згадано жодного негативного оператора, т.к. Для обробки запиту, який містить вираз WHERE (ProductId <>2), сервер читає кожен рядок, перевіряючи, чи його показник не дорівнює 2-м. Навіть індекс по ProductId не значно спрощує ситуацію, якщо рядки, що містять показник 2, не становлять дуже незначну частку таблиці. Практично абсолютно завжди сервер здійснює даний запит, переглядаючи таблиці, а не індекс.

З погляду оптимізування запитів оператор LIKE малоефективний, як і оператор NOT. Якщо у запиті є, наприклад, вираз:

WHERE client LIKE'%Mc%',

тоді сервер шукатиме заданий шаблон у всьому стовпці. Індекс не допоможе, оскільки оптимізатор запитів вибере сканування. Існує ще один тип винятків, коли пошуковий аргумент виглядає так:

WHERE client LIKE 'Le%'.

Різниця полягає в тому, що цей аспект пошуку логічно еквівалентний виразу

WHERE client>= 'Le' AND LastName < 'LF',

який, згідно з визначенням, є пошуковим аргументом. Аргументи пошуку надають запитам допомогу в тому, що полегшують оптимізатору запиту встановити ступінь селективності індексу під час обробки. Вирази, які

використовують оператори =, <, >, вважаються безпосередньо аргументами пошуку, оскільки вони обмежують пошукову область лише рядками, які у підсумковий набір. Оператор = обмежує пошукову область до єдиної строчки, а оператори зменшують її до певного спектра.

Селективність виразу відображає, наскільки ефективно аргумент пошуку обмежує область перегляду. Це значення вимірюється відношенням кількості рядків, що повертаються до загального числа рядків у таблиці. Невисокий відсоток означає, що вираз має високу селективність. Інакше високий відсоток означає слабку селективність. Так як оператор AND комутативний (a AND b позначає те саме, що і b AND a), то оптимізатор запиту може вибирати для оброблення запиту більш селективне вираз з усієї кількості виразів, які об'єднані оператором AND. Це доцільно, т.к. вибір більш селективного виразу може помітно знизити обсяг вироблених операцій введення-виведення.

Прикладом може бути запит

SELECT .... FROM Archiv where docid= 21345 AND Val='UAH'....

У цьому запиті вираз docid=21345 має дуже високу вибірковість, а вираз Val='UAH', навпаки, має середню або навіть низьку вибірковість. У разі, якщо в таблиці був би лише один рядок, у стовпці Val був би показник 'UAH', то ці два вирази були б селективні однаковою мірою.

Оптимізатор запитів встановлює, наскільки селективним є пошуковий аргумент, покладаючись на статистику відповідного індексу. Статистика свідчить, скільки записів буде відповідати заданому аспекту. Якщо ж оптимізатор запитів знає, скільки рядків міститься у таблиці, і скільки рядків буде повернено при застосуванні умов 2-х частин висловлювання WHERE, можна визначити, який індекс раціонально використовувати. У запиті, який розглядається, якщо є індекси і по стовпчику val і по стовпчику docid, запитний оптимізатор віддасть перевагу індексу по docid. Якщо ж індексу в таблиці Archiv по docid немає, а по val створено, то запитовий оптимізатор вибере його, тому що за будь-яких обставин використання індексу є найбільш селективним, ніж повне сканування таблиці.

Якщо ж відсутні обидва індекси, тоді єдиним рішенням вважається сканування таблиці з метою знаходження всіх рядків, які відповідають умовам.

У сервері кожної таблиці можна сформувати лише 1-н кластеризований індекс, тому його необхідно побудувати, задоволення максимальної кількості запитів. Кластеризовані індекси ефективні для запитів, які застосовують умови до спектру значень. Максимальна вигода від використання кластерного індексу можлива, коли речення WHERE запиту містить оператори >, < або BETWEEN .... AND, а також речення GROUP BY, де стовпці вказуються в тому самому порядку, що й у індекс. Хоча це може не допомогти ідентифікувати рядки, кластеризований індекс може покращити продуктивність системи під час обробки виразів ORDER BY, лише якщо однакові стовпці застосовуються як в індексі, так і у виразі ORDER BY і у відповідному порядку.

Оскільки інтервальний рівень кластеризованого індексу дуже невеликий, він раціональний при виявленні унікальних показників. Некластеризовані індекси краще застосовувати для точкових запитів, які можуть виявити невелику кількість рядків. Речення WHERE з оператором = вважаються основними кандидатами для створення некластеризованих індексів у відповідних стовпцях. Цей тип індексу був би більш придатним для функцій агрегації MIN і MAX, оскільки легко ідентифікувати перший і останній записи для спектру значень, якщо використовувати рівень листка індексу. Некластеризовані індекси прискорюють виконання функції COUNT, т.к. сканування рівня листка індексу набагато швидше, ніж сканування таблиці.

Результати тестування програми, побудованої за описаними вище методами, свідчать про можливість значного підвищення безпеки з’єднання та швидкості виконання запитів до великих баз даних за рахунок використання механізмів паралельної обробки.

4.4 Метод організації інтерфейсу між SQL-сервером та системою баз даних для розподілу реєстрацій

Організація інтерфейсу між SQL-сервером і ASDB може бути здійснена

двома способами: за допомогою драйверів Windows, що постачаються разом з процесорами, або за допомогою спеціальних функцій, що забезпечують доступ до системи DSP на SQL-сервері. Основною ланкою такого інтерфейсу є збережена процедура xp_cmdshell [42 – 45], яка дозволяє програмісту писати власні підпрограми будь-якою мовою програмування та викликати їх безпосередньо з SQL-запиту. Синтаксис оператора такий:

xp_cmdshell{'command_string'} [no_output]

де 'command_string' — це командний рядок для виклику будь-якої програми, яка має довжину varchar(255) або nvarchar(4000). Командний рядок містить шлях до виконуваної програми та всі перемикачі для виклику цієї програми.

no_output– означає, що повідомлення, які показує програма, не будуть видимі для клієнта, який запустив запит.

Існує три вимоги до викликаної підпрограми:

1) модуль повинен бути виконаний у стандартному об'єктному форматі;

2) модуль може бути консольним додатком, тобто. не викликати будь-які форми;

3) повинні дотримуватися угоди мови Windows про виклики та надсилання параметрів.

Функція виклику повинна формувати виклик системи DSP, який форматується як виклик програми з параметрами. Параметри – блок операторів для внутрішнього представлення мови БД. Після виклику цього виклику сервер передає блок параметрів багатопроцесорній системі та повертає відповідь.

При передачі великого обсягу даних необхідно використовувати проміжний файл, в який SQL сервер запише дані, і звідки програма їх братиме. Для передачі даних можна використовувати програму "масивного завантаження" bcp.exe (Bulk Copy Program). Програма використовується для обміну даними між текстовими файлами та таблицями бази даних. Текстові файли, що генеруються цією утилітою, можуть мати різні формати. Можливий вибір внутрішнього формату SQL сервера, що забезпечує максимальну продуктивність або символьний, що представляє дані

у зрозумілій для візуального перегляду формі. Розглянемо синтаксис цієї команди у спрощеному вигляді для нашого випадку:

$$\text{bcp Arch..archiv out archiv.txt -c } -U_{sa} \ -P_{\text{пароль}} \ -S_{\text{ім'я сервера}},$$

де Arch..archiv – ім'я таблиці, що копіюється,

archiv.txt – ім'я файлу, до якого копіюються дані,

-c – ключ для отримання текстового файлу у символьному вигляді,

$-U_{sa}$ – ім'я реєстрації для доступу до інформації,

$-P_{\text{пароль}}$ – пароль для доступу до інформації,

$-S_{\text{ім'я сервера}}$ – ім'я сервера, де зберігається інформація.

За допомогою цих команд можна створити функції для кожної з ресурсомістких операцій з обробки великих масивів інформації, що вимагають розпаралелювання, за допомогою яких можна створити інтерфейс між SQL сервером і програмами управління DSP системою.

4.5 Застосування розроблених засобів забезпечення доступу розподілених реєстрацій для оптимальної організації розподілених інформаційних ресурсів в аеропортах

В даний час більшість аеропортів світу використовують велику кількість складних інформаційних систем. У цих системах зберігаються та обробляються великі обсяги даних, які з кожним днем продовжують зростати. Дані у таких системах мають реальну цінність лише тому випадку, якщо ці дані отримані у прийнятні терміни. Це означає, що для таких систем необхідно забезпечити якнайменший проміжок між запитом користувача та отриманням ним даних.

Тому в аеропорту не тільки нарощують ємність носіїв, які зберігають БД систем, а й збільшують кількість центрів зберігання та обробки даних, а також підвищують швидкість доступу до них через мережу.

Для прискорення обробки великих БД використовують розподіл файлів БД кількома серверами, з'єднаних між собою локальної мережею [1, 5, 6, 32]. Системи управління БД, що використовують розподілену обробку таблиць БД, дозволяють

оптимізувати роботу з БД за рахунок застосування індексних таблиць, кешування, запобіжного читання даних і т.д. [31, 32]. Проте така оптимізація ефективна під час роботи з БД невеликих обсягів, та її ефект зменшується зі збільшенням обсягу даних.

Одним із способів додаткового підвищення продуктивності існуючих ІОС на базі СКБД є розпаралелювання алгоритмів [7, 38]. Обробка запиту в СКБД вимагає звернення до кількох неподільних операцій, розкладання та послідовність яких залежить від конкретного запиту, а також план виконання, обраний оптимізатором запиту. Зазвичай операції БД виконуються у суворій послідовності, причому вихід однієї операції є входом наступної. Для підтримки паралельної обробки запитів стосовно кількох серверів РБД вибирають неподільні операції, які можуть бути скопійовані і далі одночасно проводять обробку фрагментів даних і операції, результати яких можуть бути представлені так, ніби єдиний ланцюжок обчислень виконав операцію. З цією метою запити розбивають та обробляють незалежно від способів комунікації. Паралельна обробка різних ділянок БД на серверах дозволяють СКБД ефективно використовувати багатовимірні таблиці та розбиття таблиць. При правильному розподілі паралельної обробки між серверами у мережі суттєво зростає ефективність роботи з БД та управління ресурсами [1, 5, 6].

Аеропорт є складним об'єктом, [46] який включає комплекси, що оперують оперативною, статистичною і фінансовою інформацією. В даний час розвиток ІОС аеропорту зводиться до створення єдиної відкритої системи, яка заснована на використанні єдиного сховища даних різними інформаційними комплексами, заснованими на різних платформах та операційних системах. Єдине сховище (рис. 4.1) побудовано основі системи серверів, з'єднаних між собою високошвидкісної мережею.

Для доступу до єдиного сховища всі комплекси повинні бути з’єднані між собою локальною мережею, яка має доступ до глобальних мереж SITA, AFTN, Internet тощо. Для забезпечення безпеки системи, що працюють з глобальними мережами, не мають прямого доступу до одного сховища [47].

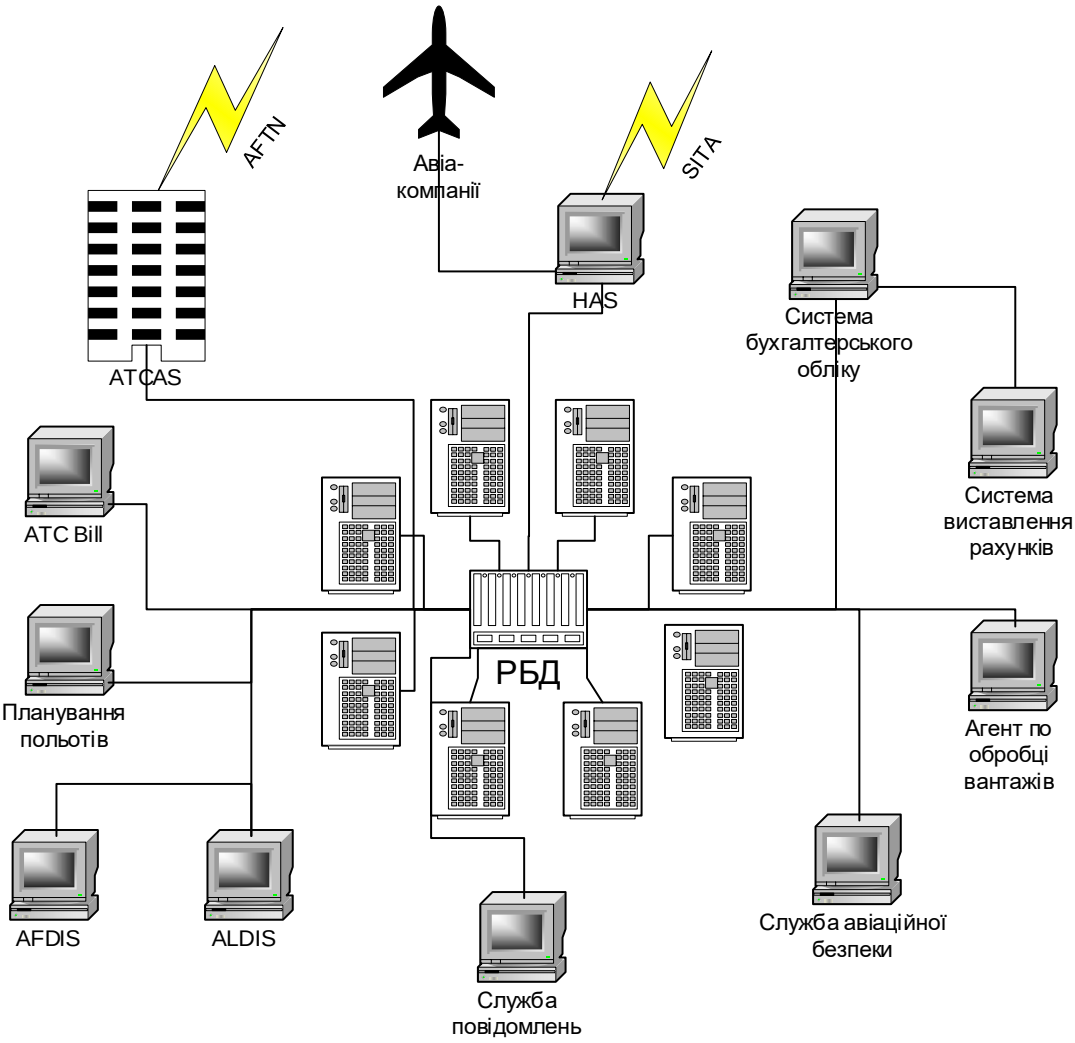


Рисунок 4.1 - Схема інформаційної системи в аеропорту

Для таких систем створено додаткові сховища, які забезпечують повну незалежність єдиної системи від зовнішніх чинників, і дозволяє вносити модифікації даних, отриманих із зовнішніх джерел. Наприклад, управлінням повітряним рухом, БД системи управління зльотом і посадкою може бути внесена коригування, яка через певний час автоматично вноситься до всіх інших систем аеропорту, наприклад, в систему інформування пасажирів. До подібних систем також можна віднести систему інформування про погодні умови, системи SITA та AFTN.

До систем, що працюють в аеропорту та використовують єдине сховище даних [47], можна віднести:

– інформаційну систему індикації розкладу польотів в аеропорту (AFIDS);

– система інформування пасажирів про зліт і посадку (ALDIS);

– Адміністрація системи управління повітряним рухом (ATCAS);

– система планування польоту, що працює як підсистема управління повітряним рухом (ATCBill);

– систему виставлення рахунків;

– план польотів;

– систему агента з обробки вантажів;

– систему бухгалтерського обліку;

– службу повідомлень;

– інформаційну систему служби авіаційної безпеки

– систему інформування пасажирів через внутрішню мережу аеропорту (RAMP);

– систему роботи з агентами авіакомпаній (HAS)

З метою виконання алгоритмів забезпечення безпеки підключення та оптимізації розміщення файлів даних по реальних комп'ютерних мережевих вузлах аеропорту, насамперед, потрібна точна вихідна інформація [7, 14]. Для встановлення інтенсивності звернень до множинних файлів інформаційних баз застосовується резидентська програма "Аналізатор запитів". Резидентська програма завантажується усім мережевих вузлах.

Програма розглядає всі звернення до файлів та записує до журналу звернень дату, час, найменування таблиці, довжину запиту, час до відповіді, довжину відповіді. Запис виконується у внутрішній буфер і пишеться диск лише під час комп'ютерного простою, коли йде введення з клавіатури чи інші операції, які вимагають очікування відповіді.

За допомогою нерезидентської частини програми виконано аналіз даних і знайдено характеристики часу звернень до файлів. Мережева завантаження характеризується нерівномірністю – абсолютною відсутністю звернень чи

одноразовим зверненням з усіх функціонуючих станцій.

Для перерозподілу запитів пошуку до довідково-інформаційних файлів між комп'ютерними мережевими вузлами призначається програма "Запитний оптимізатор у локальній мережі", що значно скорочує утворення черг.

Функціонування програми передбачає підготовку довідника розташування файлів inquire.txt. З метою забезпечення можливості змінювати за бажанням список файлів-довідників та їх мережеві адреси під час роботи з програмою "Запитовий оптимізатор у локальній мережі" передбачається отримання даних про конфігурацію мережі із файлу inquire.txt. У файлі зберігаються дані про всі файли-довідники та їх місцезнаходження. Сформований записуючий формат у файлі inquire.txt виглядає так: номер мережі, номер вузла, пристрій \[шлях]\найменування файлу. Зміни файлу inquire.txt потребують програмного перезавантаження.

Програма "Запитний оптимізатор у локальній мережі" включає три частини: інсталяція, перехоплення запитів, обробка запитів. У ході інсталяції програма здатна надсилати певні повідомлення.

Так як має бути обробка запитів на відкриття-закриття та читання-запису файлу, резидентська програмна частина вимагає обробки функцій файлового відкриття та закриття; читання із файлу; записування у файл.

Розберемо алгоритм програмної роботи з перехоплення запитів.

1. Перехоплення вектора переривання 21H.

2. Якщо № функції = FFH і № підфункції = 66H, тоді регістр AX вноситься значення коду 6677H і вихід з переривання.

3. Якщо № функції не = 3DH чи 3FH, чи 3EH, чи 40H, тоді переривання віддається застарілому оброблювачу. В іншому випадку обробник заповнює протокольний пакет IPX із запитом обробки процедури на якомусь із мережевих вузлів, відсилає повідомлення по мережі та підключає режим очікування відповіді на запит. Після того, як отримана відповідь, програма заповнює регістри прапорів та AX показниками, які прийшли в пакеті, і передає управлінські функції програмі, що викликала переривання.

В аеропорту використовується система управління даними агентів

авіакомпаній. Система працює з БД великого обсягу та встановлена на сегменті локальної мережі, що складається з 8 вузлів.

Система здійснює автоматичну обробку інформації, що вводиться авіакомпаніями та агентами з обробки вантажів. Інформація вводиться до складу системи та в подальшому обробляється відповідно до стандартів IATA. Під час перегляду та редагування даних агенту авіакомпанії надаються такі можливості:

– можливість додавання, видалення та припинення виконання рейсу, введення часу вильоту та прибуття для пасажирів;

– введення та редагування інформації щодо пасажирів та вантажів (що включає можливість замовлення, певного бортхарчування);

– введення запитів на виділення місць реєстрації пасажирів, час відкриття та закриття реєстрації на рейс;

– введення запитів на виділення гейтів для проходу пасажирів на повітряне судно, час відкриття та закриття гейтів;

– введення деталей польоту (які регламентуються Управлінням повітряного руху);

– введення оцінного часу зльоту та посадки ЗС під час прибуття та вильоту (яке не регламентуються Управлінням повітряним рухом);

– введення сезонних змін у розкладі.

Введення оцінного часу зльоту та посадки використовується системою виставлення рахунків авіакомпаніям, щоб обчислити стоянку часу. Оціночний час також використовується для збирання статистики за часом керування повітряним судном.

Інформація щодо пасажирів та вантажів інформація використовується для обчислення навантажень на термінали аеропорту, для збору статистичної інформації та обчислення коефіцієнта завантаження для терміналів у кожний момент часу, а також для аналізу прибутку аеропорту.

Інформація про час відкриття та закриття реєстрації пасажирів, а також інформація про час відкриття та закриття гейтів для проходу пасажирів,

використовується в базі даних системи індикації інформації про польоти та систему виставлення рахунків за послуги аеропорту авіакомпаніям.

При функціонуванні системи можливі такі дії користувача:

– використання універсального фільтра, тобто. будь-яка вибірка;

– зміна порядку відображення записів (сортування);

– друк поточної відфільтрованої таблиці із завданням певного порядку показу колонок, що друкуються, зі зміною їх ширини, з підрахунком сум по числовим стовпцям;

– видалення чи заміна відфільтрованих даних;

– перетворення таблиці на картку для більш наочного перегляду одного запису таблиці.

Крім того, система надає користувачеві додаткові можливості:

1) робота з формами замовлень без виходу з програми, перегляд і друк зображень замовлень, що надійшли і зберігаються в спеціальній базі даних, видача звіту про завантаженість різних рейсів авіакомпаній;

2) контекстно-залежну допомогу;

3) контроль за завантаженістю місць реєстрації;

4) остаточний розклад;

5) друк на бланках IATA, формування описів, реєстрів тощо;

6) перевірку цілісності інформації та автоматичне усунення порушень;

7) реконфігурація системи за бажанням користувача або у разі зміни законодавства;

8) гнучка система паролів для контролю доступу, криптографічного закриття інформації.

Великі обсяги оброблюваних даних можуть істотно знизити швидкість роботи програмного комплексу, якщо розподіл інформаційних файлів буде невідповідним.

Програма opt1 використовувалася для розповсюдження інформаційних файлів у системі управління інформацією агента авіакомпанії. В якості аспекту

оптимальності приймається єдиний час, необхідний для обслуговування всіх запитів, що надійшли в систему за 1 раз.

Користувачі мережі працюють з інформацією, що зберігається у восьми інформаційних файлах бази даних. При формуванні масивів вихідної інформації враховується оцінка інтенсивності корекційних і пошукових запитів λ_i,$\lambda_i^{'}$ та середнього пошукового часу у файлах t_{ij}, що отримані за допомогою резидентської програми "Аналізатор запитів" [7].

Інтенсивність запитів на виправлення та пошук у кількох файлах даних системи керування інформацією авіакомпанії RDB та середній час пошуку були записані, і з цієї інформації був створений початковий файл даних opt1.dat для програми оптимізації opt1 у такій послідовності:

– інтенсивність пошуків та виправлень у будь-якому файлі РБД;

– довжина записів у файлах і середній час обробки пошукових запитів до файлів даних.

Якщо в якості аспекту оптимальності взяти єдиний час, необхідний для обслуговування всіх запитів, що надійшли в систему протягом 1 разу, то буде створена матриця відповідного розподілу файлів, в якій перші 4 файли бази даних, інтенсивність виправлень в які є досить високими, зберігаються на 1-му з найбільш ефективних вузлів. Таке розміщення копій інформаційних файлів дозволить уникнути непотрібної надмірності даних і проблем з їх узгодженням.

У роботі запропоновано практичну реалізацію методу балансування навантаження вузлів мережі при обробці великих і надвеликих обсягів даних. Результати роботи системи управління даними авіаагента, яка працює за запропонованим методом, свідчать про можливість значного підвищення швидкості обробки даних у великооб’ємній РБД за рахунок використання механізмів оптимізації завантаження вузлів мережі, що використовуються для обробки даних. RDB. Застосування запропонованого методу дозволяє підвищити продуктивність і скоротити час відгуку інформаційної системи, що працює з РБД.

Висновки до розділу 4

1. Досліджено методи та засоби організації інтерфейсів СБД. Для організації інтерфейсів між СБД і програмами запропоновано використовувати спеціальні функції користувача MS SQL Server, що дозволяють розширити функції сервера і викликають програми керування РБД і дані, що передають в неї.

2. Запропоновано методи оптимізації мови запитів, що використовується при роботі з SQL сервером, засновані на використанні особливостей зберігання даних та обробки запитів у СКБД, а також на додаткових функціях користувача. Результати тестування програми, побудованої відповідно до цих методів, свідчать про можливість значного збільшення швидкості виконання запитів до великих БД за рахунок використання механізмів паралельної обробки SQL сервера.

3. Розглянуто організацію обробки запитів усередині СБД, дію SQL оптимізатора запитів. Запропоновано методи оптимізації управління доступом та формування запитів до сервера, методи забезпечення безпеки підключення, методи покращення обробки даних, що базуються на використанні кластеризованих та некластеризованих індексів, завдання вибіркових критеріїв пошуку тощо, що дозволяють значно прискорити обробку великих масивів.

4. Запропоновано метод оптимізації завантаження серверів РБД на основі перерозподілу завантаження залежно від результатів обробки масивів інформації, щоб не впливати на швидкість забезпечення безпеки підключення.

5. На основі запропонованого методу розроблено програму – аналізатор інтенсивності звернень до інформаційних ресурсів серверів та програму для складання оптимізованих таблиць розподілу запитів по серверах РБД, що дозволяє максимально ефективно розміщувати запити на обробку великих обсягів даних.

6. Створено програмні модулі, призначені задля забезпечення раціонального розподілу ресурсів РБД.

7. На базі запропонованих алгоритмів розроблено програми для авіакомпаній, які використовують РБД та дозволяють у стислий термін отримувати результати аналізу статистичної та фінансової інформації.

ВИСНОВКИ

У рамках дослідження та аналізу впливу розподілених систем на ефективність програмного забезпечення було досягнуто ряд теоретичних та практичних результатів, що можна ретельніше описати наступним чином:

1. Досліджено метод оптимального розміщення файлів даних у локальній мережі, враховуючи критерій загального часу обслуговування запитів під час розподілу реєстрацій. Цей метод дозволяє ефективно регулювати завантаження серверів та використовувати ресурси з максимальною продуктивністю, що призводить до зменшення часу виконання запитів.

2. Досліджено інтелектуальний алгоритм балансування завантаження вузлів мережі розподіленої бази даних, здатний оптимізувати обробку великих обсягів даних. Результати досліджень підтверджують можливість значного підвищення швидкості обробки даних за рахунок використання механізмів оптимізації завантаження вузлів мережі.

3. Досліджено адаптивний алгоритм маршрутизації, спрямований на забезпечення мінімального часу доставки пакетів даних з урахуванням властивостей інформації у пакетах. Це дозволяє ефективно знижувати час розподілу реєстрацій у розподілених системах.

4. Розроблений оптимізований алгоритм розміщення інформаційних ресурсів у РБД з метою досягнення мінімального часу відгуку інформаційно-обчислювальної системи. Експериментальні дослідження підтверджують, що розроблені алгоритми виявляються найбільш продуктивними серед існуючих у контексті розподілених обчислень.

5. Запропоновано програмований комутатор на базі програмованої логічної схеми для реалізації розроблених алгоритмів та поліпшення ефективності обчислювальної структури.

6. Представлена обчислювальна структура системи РБД, спеціально адаптована для паралельної обробки великих та надвеликих баз даних у розподілених системах.

7. Покращено структуру програмованого комутатора за рахунок впровадження поліноміального екстраполятора для прогнозування інтенсивностей інформаційних потоків у комп'ютерних мережах.

8. На основі розробленого методу створено програму-аналізатор інтенсивності звернень до інформаційних ресурсів серверів та програму для оптимізації таблиць розподілу реєстрацій у серверах РБД. Ці програми максимально ефективно впорядковують розподіл реєстрацій, сприяючи обробці великих обсягів даних.

9. Розроблено програмні модулі для раціонального розподілу ресурсів у РБД, забезпечуючи оптимальне використання обчислювальних та мережевих ресурсів.

10. На основі досліджених алгоритмів створено програмне забезпечення для авіакомпаній, яке використовує РБД, забезпечуючи швидкий аналіз статистичних та фінансових даних при розподілі реєстрацій даних у розподілених системах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт системи наближення землі EOS/DIS. https://worldview.earthdata.nasa.gov/ (Дата звернення: 15.01.2024)

2. Офіційний сайт центру Стенфордського лінійного прискорювача. https://www6.slac.stanford.edu/ (Дата звернення: 15.01.2024)

3. Яковлєв Д. А. Дослідження інтелектуальних методів синтезу розподілених баз даних : атестаційна робота здобувача вищої освіти на другому (магістерському) рівні, спеціальність 122 Комп'ютерні науки / Д. А. Яковлєв ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2022. 81 с.

4. Atul Adya, Daniel Myers, Jon Howell, Jeremy Elson, Colin Meek, Vishesh Khemani, Stefan Fulger, Pan Gu, Lakshminath Bhuvanagiri, Jason Hunter, et al. 2016. Slicer: Auto-sharding for datacenter applications. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 739-753.

5. Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, Alec Wolman, and Habinder Bhogan. 2010. Volley: Automated data placement for geo-distributed cloud services. (2010).

6. David F Bacon, Nathan Bales, Nico Bruno, Brian F Cooper, Adam Dickinson, Andrew Fikes, Campbell Fraser, Andrey Gubarev, Milind Joshi, Eugene Kogan, et al. 2017. Spanner: Becoming a SQL system. In Proceedings of the 2017 ACM International Conference on Management of Data. 331-343.

7. Peter Bailis, Ali Ghodsi, Joseph M Hellerstein, and Ion Stoica. 2013. Bolt-on causal consistency. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 761-772.

8. Itzik Ben-Gan. 2012. Microsoft SQL Server 2012 T-SQL Fundamentals .Pearson Education.

9. Peter Boncz, Marcin Zukowski, and Niels Nes. 2005. MonetDB/X100: Hyper-Pipelining Query Execution. Cidr, Vol. 5 (2005), 225-237.

10. Michael J. Cahill, Uwe Röhm, and Alan D. Fekete. 2008. Serializable Isolation for Snapshot Databases. In Proceedings of the 2008 ACM SIGMOD International

Conference on Management of Data (SIGMOD '08). ACM, New York, NY, USA, 729--738. https://doi.org/10.1145/1376616.1376690

11. Aleksey Charapko, Ailidani Ailijiang, and Murat Demirbas. 2018. Adapting to Access Locality via Live Data Migration in Globally Distributed Datastores. In 2018 IEEE International Conference on Big Data (Big Data). IEEE, 3321-3330.

12. Christos Chrysafis, Ben Collins, Scott Dugas, Jay Dunkelberger, Moussa Ehsan, Scott Gray, Alec Grieser, Ori Herrnstadt, Kfir Lev-Ari, Tao Lin, et al. 2019. FoundationDB Record Layer: A Multi-Tenant Structured Datastore. In Proceedings of the 2019 International Conference on Management of Data. 1787-1802.

13. Brian F Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. 2008. PNUTS: Yahoo!'s hosted data serving platform. Proceedings of the VLDB Endowment, Vol. 1, 2 (2008), 1277-1288.

14. Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing. 143-154.

15. Sudipto Das, Divyakant Agrawal, and Amr El Abbadi. 2010. G-store: a scalable data store for transactional multi key access in the cloud. In Proceedings of the 1st ACM Symposium on Cloud Computing. ACM, 163-174.

16. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: amazon's highly available key-value store. In ACM SIGOPS operating systems review, Vol. 41. ACM, 205-220.

17. Murat Demirbas, Marcelo Leone, Bharadwaj Avva, Deepak Madeppa, and Sandeep Kulkarni. 2014. Logical physical clocks and consistent snapshots in globally distributed databases. (2014).

18. Hua Fan and Wojciech Golab. 2019. Ocean vista: gossip-based visibility control for speedy geo-distributed transactions. Proceedings of the VLDB Endowment, Vol. 12, 11 (2019), 1471-1484.

19. Rachid Guerraoui and Jingjing Wang. 2017. How fast can a distributed transaction commit?. In Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. ACM, 107-122.

20. Robert Kallman, Hideaki Kimura, Jonathan Natkins, Andrew Pavlo, Alexander Rasin, Stanley Zdonik, Evan PC Jones, Samuel Madden, Michael Stonebraker, Yang Zhang, et al. 2008. H-store: a high-performance, distributed main memory transaction processing system. Proceedings of the VLDB Endowment, Vol. 1, 2 (2008), 1496-1499.

21. Rusty Klophaus. 2010. Riak core: Building distributed applications without shared state. In ACM SIGPLAN Commercial Users of Functional Programming. ACM, 14.

22. Alexey Kopytov. 2012. SysBench manual. http://imysql.com/wp-content/uploads/2014/10/sysbench-manual.pdf. MySQL AB (2012).

23. Tim Kraska, Gene Pang, Michael J Franklin, Samuel Madden, and Alan Fekete. 2013. MDCC: Multi-data center consistency. In Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 113-126.

24. Avinash Lakshman and Prashant Malik. 2010. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, Vol. 44, 2 (2010), 35-40.

25. Qian Lin, Pengfei Chang, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, and Zhengkui Wang. 2016. Towards a non-2pc transaction management in distributed database systems. In Proceedings of the 2016 International Conference on Management of Data. ACM, 1659-1674.

26. Guoxin Liu and Haiying Shen. 2017. Minimum-cost cloud storage service across multiple cloud providers. IEEE/ACM Transactions on Networking (TON), Vol. 25, 4 (2017), 2498-2513.

27. Hatem Mahmoud, Faisal Nawab, Alexander Pucher, Divyakant Agrawal, and Amr El Abbadi. 2013. Low-latency multi-datacenter databases using replicated commit. Proceedings of the VLDB Endowment, Vol. 6, 9 (2013), 661-672.

28. Faisal Nawab, Divyakant Agrawal, and Amr El Abbadi. 2018. Dpaxos: Managing data closer to users for low-latency and mobile applications. In Proceedings of the 2018 International Conference on Management of Data. ACM, 1221-1236.

29. Faisal Nawab, Vaibhav Arora, Divyakant Agrawal, and Amr El Abbadi. 2015. Minimizing commit latency of transactions in geo-replicated data stores. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 1279-1294.

30. Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, et al. 2013. Scaling memcache at facebook. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 385-398.

31. Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference (USENIX ATC 14). 305-319.

32. Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C Mowry, Matthew Perron, Ian Quah, et al. 2017. Self-Driving Database Management Systems.. In CIDR, Vol. 4. 1.

33. Fan Ping, Jeong-Hyon Hwang, XiaoHu Li, Chris McConnell, and Rohini Vabbalareddy. 2011. Wide area placement of data replicas for fast and highly available data access. In Proceedings of the fourth international workshop on Data-intensive distributed computing. ACM, 1-8.

34. Dan RK Ports and Kevin Grittner. 2012. Serializable snapshot isolation in PostgreSQL. Proceedings of the VLDB Endowment, Vol. 5, 12 (2012), 1850-1861.

35. Raphael 'kena' Poss. 2018. The “PostgreSQL” in CockroachDB - Why? (May 2018). URL: https://dr-knz.net/postgresql-cockroachdb-why.html (дата звернення: 04.01.24).

36. Ian Rae, Eric Rollins, Jeff Shute, Sukhdeep Sodhi, and Radek Vingralek. 2013. Online, Asynchronous Schema Change in F1. VLDB, Vol. 6, 11 (2013), 1045-1056.

37. Kun Ren, Dennis Li, and Daniel J Abadi. 2019. SLOG: serializable, low-latency, geo-replicated transactions. Proceedings of the VLDB Endowment, Vol. 12, 11 (2019), 1747-1761.

38. RocksDB. [n.d.]. URL: https://rocksdb.org/ (дата звернення: 11.01.24).

39. Debanjan Saha, Gurmit Singh Ghatore, and Brandon O'Brien. 2017. DAT202: Getting started with Amazon Aurora. URL: https://www.slideshare.net/AmazonWebServices/dat202getting-started-with-amazon-aurora/14 (дата звернення: 11.01.24).

40. Marco Serafini, Essam Mansour, Ashraf Aboulnaga, Kenneth Salem, Taha Rafiq, and Umar Farooq Minhas. 2014. Accordion: Elastic scalability for database systems supporting distributed transactions. Proceedings of the VLDB Endowment, Vol. 7, 12 (2014), 1035-1046.

41. Marco Serafini, Rebecca Taft, Aaron J Elmore, Andrew Pavlo, Ashraf Aboulnaga, and Michael Stonebraker. 2016. Clay: Fine-grained adaptive partitioning for general database schemas. Proceedings of the VLDB Endowment, Vol. 10, 4 (2016), 445-456.

42. Artyom Sharov, Alexander Shraer, Arif Merchant, and Murray Stokely. 2015. Take me to your leader!: online optimization of distributed storage configurations. Proceedings of the VLDB Endowment, Vol. 8, 12 (2015), 1490-1501.

43. Alexander Shraer, Benjamin Reed, Dahlia Malkhi, and Flavio P Junqueira. 2012. Dynamic Reconfiguration of Primary/Backup Clusters. In Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12). 425-437.

44. Jeff Shute, Radek Vingralek, Bart Samwel, Ben Handy, Chad Whipkey, Eric Rollins, Mircea Oancea, Kyle Littlefield, David Menestrina, Stephan Ellner, et al. 2013. F1: A distributed SQL database that scales. Proceedings of the VLDB Endowment, Vol. 6, 11 (2013), 1068-1079.

45. Kristina Spirovska, Diego Didona, and Willy Zwaenepoel. 2018. Wren: Nonblocking reads in a partitioned transactional causally consistent data store. In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 1-12.

46. Michael Stonebraker and Ariel Weisberg. 2013. The VoltDB Main Memory DBMS. IEEE Data Eng. Bull., Vol. 36, 2 (2013), 21-27.

47. Roshan Sumbaly, Jay Kreps, Lei Gao, Alex Feinberg, Chinmay Soman, and Sam Shah. 2012. Serving large-scale batch computed data with project voldemort. In Proceedings of the 10th USENIX conference on File and Storage Technologies. USENIX Association, 18-18.

48. Rebecca Taft, Essam Mansour, Marco Serafini, Jennie Duggan, Aaron J Elmore, Ashraf Aboulnaga, Andrew Pavlo, and Michael Stonebraker. 2014. E-store: Fine-grained elastic partitioning for distributed transaction processing systems. Proceedings of the VLDB Endowment, Vol. 8, 3 (2014), 245-256.

49. Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J Abadi. 2012. Calvin: fast distributed transactions for partitioned database systems. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 1-12.

50. Benjamin Treynor Sloss. 2019. An update on Sunday's service disruption. URL: https://cloud.google.com/blog/topics/inside-google-cloud/an-update-on-sundays-service-disruption (дата звернення: 15.01.24).

51. Misha Tyulenev, Andy Schwerin, Asya Kamsky, Randolph Tan, Alyson Cabral, and Jack Mulrow. 2019. Implementation of Cluster-wide Logical Clock and Causal Consistency in MongoDB. In Proceedings of the 2019 International Conference on Management of Data. ACM, 636-650.

52. Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao. 2017. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 1041-1052.

53. Todd Warszawski and Peter Bailis. 2017. ACIDRain: Concurrency-related attacks on database-backed web applications. In Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 5-20.

54. Zhe Wu, Michael Butkiewicz, Dorian Perkins, Ethan Katz-Bassett, and Harsha V Madhyastha. 2013. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 292-308.

55. Xinan Yan, Linguan Yang, Hongbo Zhang, Xiayue Charles Lin, Bernard Wong, Kenneth Salem, and Tim Brecht. 2018. Carousel: low-latency transaction processing for globally-distributed data. In Proceedings of the 2018 International Conference on Management of Data. ACM, 231-243.

56. Victor Zakhary, Faisal Nawab, Divy Agrawal, and Amr El Abbadi. 2018. Global-Scale Placement of Transactional Data Stores.. In EDBT. 385-396.

57. Irene Zhang, Naveen Kr Sharma, Adriana Szekeres, Arvind Krishnamurthy, and Dan RK Ports. 2018. Building consistent transactions with inconsistent replication. ACM Transactions on Computer Systems (TOCS), Vol. 35, 4 (2018), 12.

58. Tao Zhu, Zhuoyue Zhao, Feifei Li, Weining Qian, Aoying Zhou, Dong Xie, Ryan Stutsman, Haining Li, and Huiqi Hu. 2018. Solar: towards a shared-everything database on distributed log-structured storage. In 2018 USENIX Annual Technical Conference (USENIX ATC 18). 795-807.

59. Офіційна сторінка специфікації методики EJB. https://www.oracle.com/java/technologies/#specs (Дата звернення: 15.01.2024).

60. Розділ DCOM у бібліотеці MSDN. https://msdn.microsoft.com/en-us/library/ms878122.aspx (Дата звернення: 15.01.2024).

61. Офіційний веб-сайт системи InterBase. https://www.embarcadero.com/products/interbase (Дата звернення: 15.01.2024).

62. Офіційний сайт Mathworks Matlab. https://www.embarcadero.com/products/interbase (Дата звернення: 15.01.2024).

63. Офіційна сторінка специфікацій Xilinx. https://www.xilinx.com/products/silicon-devices/fpga.html (Дата звернення: 15.01.2024).

64. Офіційний сайт операційної системи NetBSD. https://www.netbsd.org/ (Дата звернення: 15.01.2024).