МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра систем информатики

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Компьютерные науки и системотехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Бухнера Марка Евгеньевича

Тема работы:

ИССЛЕДОВАНИЕ И ВНЕДРЕНИЕ ФИЛЬТРОВ ДЛЯ УСКОРЕНИЯ ЗАПРОСОВ К РАСПРЕДЕЛЕННОЙ БАЗЕ ТРАНЗАКЦИОННЫХ ДАННЫХ

«К защите допущена»
Заведующий кафедрой,
д.ф.-м.н., профессор
Лаврентьев М. М. / _____
«30» мая 2024 г.

Руководитель ВКР
Dr.rer.nat., доцент
каф. ТК ММФ НГУ
ван Беверн Р. А. / _____
«30» мая 2024 г.

Новосибирск, 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Платформы для анализа больших данных состоят из множества сервисов, среди них ссылки? выделяются системы для непосредственной обработки данных, системы для доступа к данным и сами хранилища данных.

Хранилища данных в таких системах являются внешними удаленными распределенными сервисами, что превращает ввод-вывод такого хранилища в одно из основных узких мест при обработке запросов. Существует множество подходов к организации файлов для их эффективного чтения, что позволяет сократить время оптимизировать? обработки запроса. Тем не менее, ускорение выполнения запросов возможно и другими путями, один из них — отсеивание файлов с данными, которые не удовлетворяют предикату запроса.

Существующие методы отсеивания файлов с данными используют упрощённые сводки (например, минимальные и максимальные значения каждого атрибута) для всех файлов данных, чтобы фильтровать те файлы, в которых не содержится записей, который удовлетворяют предикату запроса. Однако при работе с реальными данными такие подходы не всегда оказываются эффективными ввиду того, что каждый файл с данными может содержать большой диапазон значений, например — отметки времени за январь по декабрь определенного года. Тогда для запросов, нацеленных на извлечение данных за определенные периоды, например, с июня по август того же года, такой подход приведет к необходимости чтения файлов, которые не содержат нужной информации.

Целью данной работы является разработка оптимальных методов индексации данных для ускорения обработки запросов с условиями в контексте платформ больших данных на основе анализа существующих подходов и их применимости на практике.

Задача исследования состоит в разработке структуры данных, которая позволит ускорить обработку интервальных запросов в платформах для обработки больших данных, а также во внедрении этой структуры данных в

одну из платформ с последующим проведением экспериментов на реальных наборах данных.

ГЛАВА 1. ОПРЕДЕЛЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ТЕКУЩИХ РЕШЕНИЙ

1.1 Предметная область

В научной литературе существует множество определений термина **большие данные** [1, 2, 3]. Однако в рамках данной работы целесообразно выделить общие характеристики из всех предложенных определений.

Распространённой ошибкой при определении понятия большие данные является попытка придать данным численную оценку объёма, потому что объём данных, классифицируемых как большие, не только постоянно увеличивается, но и зависит от возрастающих вычислительных мощностей систем, на которых эти данные обрабатываются и хранятся.

Несмотря на необходимость дать численные оценки для данных в некоторых частях данной работы, это не противоречит основному свойству больших данных — их нельзя эффективно обработать или хранить на одном вычислительном узле. Следовательно, обработка таких объёмов данных требует использования множества вычислительных узлов, которые работают над одной задачей и видны пользователям как одна цельная система, такие системы называются **распределенными** [4].

Распределенные системы для обработки больших данных состоят из множества сервисов, однако их можно разделить на три уровня: [5]

1. **Системы для обработки данных** — распределенная вычислительная система, которая отвечает на запросы пользователей.
2. **Система для доступа к данным** — управляет организацией данных для более быстрого доступа к ним.
3. **Хранилище данных** — внешний распределенный сервис, доступ к которому осуществляется по сети.

Системы с такой организацией будем называть **платформами для обработки больших данных**.

Именно системы для доступа к данным организуют структуру файлов в хранилище данных для более быстрого доступа к ним. Например, одной из

систем для хранения и доступа к данным является Apache Hudi — транзакционная система для доступа к данным, которая поддерживает исторические запросы, а также отвечает за консистентность данных при записи и чтении. В первом приближении такую систему можно охарактеризовать как формат хранения данных. В качестве хранилища она использует файловую систему Hadoop Distributed File System (HDFS), являющуюся частью проекта Apache Hadoop. откуда информация?

Для систем существует два способа увеличения общей производительности, а также увеличения предельного объема данных, который возможно сохранить в системе: [6]

1. **Горизонтальное масштабирование** — увеличение количества отдельных вычислительных узлов в системе, выполняющих одну и ту же функцию.
2. **Вертикальное масштабирование** — увеличение производительности каждого вычислительного узла системы, путем использования более производительных компонентов, более объемных устройств хранения данных.

Выбор конкретной реализации хранилища данных не важен в контексте данной работы, так как все они обладают общими свойствами, это внешние сервисы, взаимодействие с которыми происходит по сети. Такие системы используются в платформах для обработки больших данных ввиду того, что такие системы хорошо масштабируются горизонтально. Это необходимо для надежного хранения данных и возможности дешевого добавления памяти в файловую систему [7].

Такая архитектура приводит к тому, что самое узкое место всей платформы при обработке данных — доступ к данным, так как скорость чтения данных по сети на порядки ниже скорости чтения данных с локального запоминающего устройства или оперативной памяти [8]. Существует множество способов организации данных для более быстрого их чтения например, колоночный формат хранения данных?, что позволяет сократить время обработки запроса. Тем не менее, ускорение выполнения

запросов возможно и другими путями, один из них — отсеивание файлов с данными, которые не удовлетворяют предикату запроса.

Apache Hudi поддерживает работу с историческими запросами: при обновлении набора данных создаётся новый снимок, в то время как предыдущий остаётся доступным. Каждый снимок соответствует определённому моменту времени на временной шкале, которая представляет собой журнал всех операций, выполненных над таблицей. Эта временная шкала позволяет отслеживать последовательность данных и обеспечивает доступ к информации для исторических запросов. На каждый момент времени приходится не более одного действия, создавая линейный порядок среди всех операций над таблицей. Каждое действие в журнале содержит информацию о времени (с точностью до миллисекунд), типе и состоянии операции. В данной работе рассматриваются только действия, связанные с модификацией данных. Состояние каждого действия может быть «запланировано», «в процессе» или «завершено». Таким образом, Hudi использует временную шкалу для оптимистичного управления параллельным доступом к данным.

В каждой таблице Hudi существует служебная директория, в которой находится вся метаинформация, необходимая для работы. Временная шкала находится в данной служебной директории, каждому состоянию действия соответствует один файл, название которого начинается с момента времени действия.

В рамках служебной директории располагается также таблица метаданных, которая является внутренней таблицей Hudi и остаётся скрытой от пользователя. Эта таблица интегрирована в пользовательскую таблицу, но недоступна для прямого доступа.

В Hudi различают два типа таблиц:

1. Copy-On-Write: данные хранятся в файлах с колоночно-ориентированным форматом. При операции записи генерируется новая версия файла данных, которая создаётся путём слияния существующего файла данных с новыми поступившими данными.

2. Merge-On-Read: данные хранятся в файлах, сочетающих колоночно-ориентированный и строчно-ориентированный форматы. В процессе записи новые данные сначала сохраняются в дельта-файлах строчного формата, а затем сливаются с последней версией файла данных в процессе, известном как компактизация.

Компактизация может происходить как в синхронном, так и в асинхронном режиме.

Размер таблицы увеличивается с добавлением данных. Поскольку для размера файла данных существует определённый разумный предел, данные разделяются на файловые группы. При достижении заданного конфигурируемого предела размера файла данных формируется новая файловая группа. Записи с новыми ключами размещаются в новой файловой группе, в то время как обновления записей с уже известными ключами записываются в ту же файловую группу, где находится оригинальная запись. Независимо от типа таблицы, каждый файл данных ассоциирован с определённой файловой группой. Файловая группа обладает уникальным случайно генерируемым идентификатором, который фигурирует в начале названия каждого файла данных. Кроме того, каждый файл данных ассоциируется с моментом времени действия на временной шкале, который также указан в названии файла.

Файловые группы расположены в директориях файловой системы, причём путь к такой директории называется партицией. Путь для каждой файловой группы генерируется на основе пользовательски заданной схемы партиционирования. Схема партиционирования включает в себя список атрибутов таблицы, значения которых определяют путь расположения файловой группы. Эти значения атрибутов используются для оптимизации числа партиций, которые необходимо просканировать при выполнении запроса. Это означает, что если партиционирование выполнено по определённому атрибуту и запрос включает условие по этому атрибуту, процесс поиска местоположения файловой группы сводится к определению идентификатора этой группы.

Далее рассмотрим основные понятия, связанные с индексами. Индексы представляют собой структуры данных, возникшие из необходимости обеспечить быстрый поиск информации, хранящейся в базах данных. В отсутствие индексов поиск информации по всем записям, удовлетворяющим определённым критериям, требовал бы последовательного доступа к каждой записи для проверки её соответствия условиям. Для базы данных, содержащей N элементов, это потребовало бы времени порядка O(N), что для современных баз данных является неэффективным.

Индекс — это структура данных, которая позволяет ускорить процесс поиска за счёт использования дополнительного пространства и выполнения дополнительных операций записи для поддержания своей структуры. Существует множество типов индексов, предназначенных для работы с различными типами данных, такими как пространственные, временные, текстовые, многомерные и другие. Выбор подходящего индекса для конкретной задачи является ключевым аспектом процесса оптимизации, поскольку это может значительно влиять на временную сложность поиска, которая варьируется от O(logN) до O(1).

Учитывая изложенное, можно заключить, что задача определения расположения файла данных по заданному атрибуту с использованием индекса сводится к нахождению файловой группы, в которой расположена данная запись. В дальнейшем, для целей данной работы, термин «файловая группа» будет интерпретирован как «местоположение файла данных». При этом выбор конкретной версии файла данных в файловой группе зависит от типа запроса, включая исторические запросы. В рамках данной работы индекс будет отображать некоторые значения атрибутов на идентификатор файловой группы (расположение файла данных), в которой расположены соответствующие записи.

ГЛАВА 2. ДРУГАЯ ЧАСТЬ

2.1 Первый подраздел

Таблица 1 – Пример таблицы для ВКР

Модель	**Точность**	**Время обучения (ч)**
Модель 1	92	12
Модель 2	94	18
Модель 3	95	20

Таблица 2 – Пример большой таблицы

№	Фамилия	Имя	Отчество	Оценки		Статистик
				Математика	Физика	Средний балл
1	Иванов	Иван	Иванович	5	4	4.5
2	Петров	Петр	Петрович	4	5	4.5
3	Сидоров	Сидор	Сидорович	5	5	5
4	Кузнецов	Дмитрий	Иванович	4	4	4
5	Морозов	Андрей	Андреевич	3	5	4
6	Новиков	Иван	Дмитриевич	5	3	4
7	Васильев	Андрей	Петрович	3	4	3.5
8	Андреев	Андрей	Андреевич	4	5	4.5
9	Григорьев	Иван	Петрович	3	3	3
10	Дмитриев	Дмитрий	Андреевич	4	4	4

ЗАКЛЮЧЕНИЕ

По итогу выполненной работы разработан и реализован на языке C++ алгоритм решения задачи, который прошел тестирование на ряде тестовых данных и показал хорошие результаты по скорости и точности.

Для дальнейшего улучшения алгоритма возможно использование различных оптимизаций и улучшений, таких как:

1. Использование параллельных вычислений для ускорения работы алгоритма на многопроцессорных системах.
2. Оптимизация алгоритма с использованием более эффективных структур данных и алгоритмов, например, деревьев или хеш-таблиц.
3. Расширение функциональности алгоритма для обработки более широкого спектра данных, например, текстовых данных или изображений.
4. Интеграция алгоритма с другими системами и инструментами для улучшения процесса работы и повышения эффективности.

Также возможно проведение дополнительных исследований для определения оптимальных параметров алгоритма и проведения более детального анализа результатов работы. В целом, разработанный алгоритм представляет собой хорошую основу для дальнейшей работы и развития в этой области.

Результаты работы представлены на 60-й Международной научной студенческой конференции (НГУ, 10–20 апреля 2022 г.) в секциях «Наукоёмкое программное обеспечение» и «Теоретическая кибернетика». В последней работа была отмечена дипломом первой степени.

Выпускная квалификационная работа выполнена мной самостоятельно с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Бухнер Марк Евгеньевич _____

(подпись)

«___»_____20___г.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Sagiroglu S. Sinanc D. Big data: A review // 2013 international conference on collaboration technologies and systems (CTS). — 2013. — С. 42.
2. H. Mohanty. Big data: An introduction. — Springer India, 2015. — 1–2 с.
3. Fan J. Han F. Liu H. Challenges of big data analysis // National science review. — 2014. — Т. 1, № 2. — С. 293.
4. L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System // Communications of the ACM. — 1978. — Т. 21, № 7. — С. 558.
5. др. Errami S. A. и. Spatial big data architecture: from data warehouses and data lakes to the Lakehouse // Journal of Parallel and Distributed Computing. — 2023. — Т. 176. — С. 70–73.
6. El-Rewini H. Abd-El-Barr M. Advanced computer architecture and parallel processing. — John Wiley & Sons, 2005. — 63–67 с.
7. Pan X. Luo Z. Zhou L. Navigating the Landscape of Distributed File Systems: Architectures, Implementations, and Considerations // Innovations in Applied Engineering and Technology. — 2023. — С. 1–12.
8. И. Соловьев А. Хранение и обработка больших данных // Тенденции развития науки и образования. — 2018. — № 37-6. — С. 47–51.