Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.942

Ильюкевич
Андрей Викторович

Адаптация конфигурации хранилищ данных на основе специфики предметной области

АВТОРЕФЕРАТ
на соискание академической степени
магистра технических наук

по специальности 1-40 80 05 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Научный руководитель
Лапицкая Н. В.
к.т.н., доцент

Минск 2019

КРАТКОЕ ВВЕДЕНИЕ

Данные – это неотъемлемая часть каждой информационной системы. Правильное проектирование любой системы, даже самой маленькой начинается с изучения предметной области и данных с которыми система будет работать. При проектировании предметная область представлена множеством объектов, их атрибутов и функций. Но данная абстракция которая хорошо подходит для проектирования классов приложения, зачастую приводит к неправильному использованию хранилищ данных. В связи с этим при разработке и поддержке крупных информационных систем привлекаются специалисты по работе с данными. Администраторы баз данных затрагивают не только объекты и атрибуты в рамках предметной области, но и их свойства. Специалисты часто задают вопросы о ряде свойств, таких как:

– частота, время и объём изменения данных;

– частота, время и объём поступления новых данных.

Все эти вопросы являются основой для построения оптимальных механизмов загрузки и работы с данными, особенно, когда система является концентратором данных из различных систем. Часть ответов получают аналитики при общении с экспертами в предметной области или заказчиками от бизнеса и пользователями, однако не всегда эти ответы удаётся получить на момент проектирования, что приводит к неэффективной работе при эксплуатации приложений.

При работе с внешними источниками данных, случается, что ошибки на стороне источника портят работу целевого приложения. Зачастую в таких случаях разрабатываются метрики и механизмы проверки. Когда формат и объём данных строго детерминирован и был выявлен в момент разработки, это даёт основу для разработки механизмов проверки, но часто бывает, что эти данные нельзя получить, и они поступают постфактум. В таких случаях приходится анализировать причину сбоя, и разрабатывать методику защиты постфактум.

Также при работе с внешними источниками возникает вопрос загрузки данных. Очень удобно, когда источник отслеживает изменения, и может по запросу выдать все новые данные после определённого времени. Но в реальных условиях такая возможность есть не всегда, поэтому возникает вопрос отслеживания изменений и репликации данных. Зачастую для этого разрабатываются специальные сервисы, которые периодически сравнивают данные источника с предыдущим состоянием, это разумно, когда данные поступают постоянно. Но в случаях, когда данные поступают редко, этот механизм ресурсозатратен и почти всегда расходует ресурсы впустую. Поэтому подбор расписания запуска таких механизмов имеет важную роль в эксплуатации систем.

Правильно использование ресурсов актуально, поскольку важный аспект современных систем это своевременное поступление данных и увеличение скорости работы приложения. Поскольку всё чаще вертикальное масштабирование не справляется с нагрузкой, всё актуальнее становится разбиение задач на более простые и атомарные, для использования вертикального масштабирования и оптимизации каждой из них.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель и задачи исследования

Цель магистерской диссертации – разработать инструмент, который позволит анализировать хранилища данных и выявить правила и механизмы для адаптации его под реальное использование в системе.

Объектом исследования являются базы данных, имеющие реляционную модель в уже функционирующих системах.

Предметом исследования являются закономерности поступления данных в хранилище, время выполнения операций, физические затраты ресурсов на выполнение операций.

Основной гипотезой, положенной в основу диссертационной работы, является возможность прогнозировать события на основе исторических данных.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать механизм получения данных о работе реляционной базы данных, по возможности, не влияющий на работу СУБД и работающий независимо.
2. Спроектировать подходящую архитектуру и разработать автоматические инструменты для анализа на основе полученных данных.
3. Провести исследования целевого приложения используя разработанные инструменты и выявить закономерности и аномалии.
4. Разработать автоматические или полуавтоматические механизмы по выявлению неэффективного использования хранилища данных.

Связь работы с приоритетными направлениями научных исследований и запросами реального сектора экономики

Работа выполнялась в соответствии с научно-техническим заданием и планом работ кафедры «Программное обеспечение информационных технологий» по теме «Разработка моделей, методов, алгоритмов, повышающих показатели проектирования, внедрения и эксплуатации программных средств для перспективных платформ обработки информации, решения интеллектуальных задач, работы с большими массивами данных и внедрение в современные обучающие комплексы» (ГБ № 16-2004, № ГР 20163588, научный руководитель НИР – Н. В. Лапицкая).

Личный вклад соискателя

Результаты, приведенные в диссертации: сбор данных, архитектура, разработка инструментов для анализа и результаты анализа, – получены соискателем лично.

Публикация результатов диссертации

В рамках выполнения исследования было опубликовано 3 печатные работы. Среди них 2 публикации были сделаны на международной конференции Webconf 2018 и Информационные технологии и системы 2018; цель публикации – продемонстрировать результаты применения разрабатываемой системы.

Структура и объем диссертации

Диссертация состоит из введения, общей характеристики работы, трёх глав, заключения, списка использованных источников, списка публикаций автора и приложений. Первая глава посвящена выбору метода сбора данных о работе хранилища данных и выбору инструментов для разработки приложения. Во второй главе описываются алгоритмы и технологии реализующие различные компоненты анализа. В третьей главе приведён практический опыт использования разработанной системы на примере двух реально функционирующих систем. В главе приведены примеры того, как система помогла решить ряд задач и выявить несколько неполадок. Также приведены примеры того, как система, помогает анализировать дефекты и устранять их.

Общий объем работы составляет 60 страниц, из которых основного текста 47 страниц, 15 рисунков, 2 таблицы, список использованных источников из 36 наименований и 1 приложения на 13 страницах.

ОСНОВНОЕ СОДЕРЖАНИЕ

Во **введении** определена область и указаны основные направления исследования, показана актуальность темы диссертационной работы, дана краткая характеристика исследуемых вопросов, обозначена практическая ценность работы.

В **первой главе** проведены сбор и подготовка данных, а также выбор инструмента их анализа. Анализ существующих решений сбора данных показал, что минимизировать влияние инструмента для сбора данных на целевую БД можно с помощью отслеживания лога транзакций. На основе циклического метода хранения логов транзакций реализован на shell алгоритм, позволяющий отслеживать появление новых файлов с логами, не затрагивая при этом менеджер БД и минимально нагружая менеджер файловой системы.

Проанализированы существующие архитектурные решения, произведен выбор внешнего хранилища данных. Сделан выбор в пользу микросервисной архитектуры в сочетании с событийным шаблоном. Данный выбор обусловлен в первую очередь отказоустойчивостью: приложение будет работать несмотря на отказ одного из его модулей. Такая архитектура обладает независимыми развертыванием и масштабированием, которые обеспечивают отказоустойчивость системы. Событийный шаблон реализует модель публикации и подписки: клиенты подписываются на поток и каждое произошедшее событие немедленно отправляется им как сообщение. Такой шаблон полностью реализует Kafka. Он будет работать как связующее звено, обеспечивающее взаимодействие всех приложений. Это позволит приложениям быть независимыми друг от друга.

Spark выбран в качестве инструмента для обработки данных. Он позиционируется как инструмент для «молниеносных кластерных вычислений». Инструмент позволяет производить вычисления в памяти в 100 раз, а на диске в 10 раз быстрее, чем аналогичный ему Hadoop. Подходящие технические параметры, дополнительные библиотеки и многочисленное сообщество позволяют использовать Spark для широкого круга задач.

Во **второй главе** проведено моделирование системы и разработаны инструменты анализа данных. На данном этапе были разработаны ряд инструментов, которые позволяют в дальнейшем просматривать данные в различных разрезах, прогнозировать изменения, кластеризовать таблицы, отличать физические изменения данных от изменений с точки зрения бизнес сущностей. Набор инструментов был получен на основе эмпирического опыта, полученного за несколько лет работы с различными базами и приложениями. Благодаря микросервисной архитектуре, выбор инструмента для каждой задачи был независимым и подбирался индивидуально.

Задача преобразования логов во внутренний формат системы была решена с помощью библиотеки Patterns, ставшей основой для парсера pars_log, написанного на ЯП Python. Данная библиотека позволяет создать маску для чтения файла, схожую с регулярным выражением. Маска применяется к объекту типа поток, который совместим с основным интерфейсом чтения данных kafka connect. На выходе получается список объектов, которые в свою очередь

передаются потоку на запись, который записывает в новую тему в Kafka (parsed_database_log). Таким образом из целевой БД данные поступают в темы Kafka и хранятся в них.

Эти данные необходимо фильтровать на типы. Типы логов могут быть следующими: модификация данных, изменение структуры таблицы, создание внутренних объектов базы или системные команды. Эта задача была решена с использованием ЯП Scala, на котором написан Spark. Сервис db_log_clasification считывает данные из темы parsed_database_log и записывает их в 3 других темы: db_change_data, db_change_structure, db_change_tmp_object. При этом запись в эти темы происходит параллельно и одновременно, благодаря механизму мультивывода у потоков Scala. Данные разбиваются на отдельные темы, поскольку дальнейшая обработка зависит от типа, и может выполняться независимо.

Далее был реализован инструмент для оценки эффективности кода. Этот инструмент позволяет дифференцировать физические и логические изменения данных в таблице. При изменении строки в таблице логически происходит одно действие, однако физически удаляется старая строка и создается новая. Многочисленные исследования показывают, что наличие менее 30% изменения и более 100 тысяч записей в таблице, в независимости от их размера, делают неэффективным полный пересчёт таблицы по причине очень большой нагрузки на диск. В связи с этим, инструмент, который сможет постфактум оценить эффективность выполняемых запросов, выглядит весьма полезным. В связи с этим для реализации была выбрана библиотека Apache Spark Streaming. Данная библиотека подходит для фильтрации и операций слияния потока самим с собой.

Но данная библиотека накладывает достаточно весомое ограничение – необходимо определить временное окно, в рамках которого осуществляется слияние. Данное окно определятся на основе данных из темы db_change_tmp_object, в который попадают данные о коммитах в каждом из процессов. Сервис подписан на тему о коммитах. Когда он получает информацию о открытой и закрытой транзакции, он формирует временное окно и считывает информацию из темы db_change_data, который отражает физические изменения, осуществляет операцию слияние и пишет результат в новую темы db_real_change_data, отражающий логические изменения данных.

В процессе эксплуатации, данная служба показывала показатели выше, чем ожидалось, однако после анализа была выявлена проблема – сравнивается весь фрейм данных. Получается если таблица содержит системные автоматически сгенерированные колонки, например, со временем, сервис показывает 100% изменение. Поэтому к существующему фильтру по типу операции над таблицей, были добавлены дополнительные, на основе типа данных, чтобы улучшить точность измерения.

Задача обработки отката транзакции в БД была решена с помощью сервиса, срабатывающего по событию. Сервис подписан на тему db_change_tmp_object, и, если он обнаруживает в данных, лежащих в теме, команду commit, то запускает классификатор с параметром «реальное изменение». Если же в данных из темы сервис обнаруживает команду rollback, то параметр будет «ошибочные данные». Такие данные будут накапливаться в теме db_rollback_data.

Кластеризация таблиц – это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны. Главное отличие кластеризации от классификации состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма.

Чтобы использовать алгоритмы кластеризации для начала нужно составить вектор характеристик для каждого объекта - как правило, это набор числовых значений, например, рост-вес человека. Однако существуют также алгоритмы, работающие с качественными (категорийными) характеристиками.

После того, как определён вектор характеристик, можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния». В процессе нормализации все значения могут приводиться к некоторому диапазону. Наконец, для каждой пары объектов измеряется «расстояние» между ними – степень похожести.

Алгоритм кластеризации таблиц был реализован с помощью библиотеки Apache Spark Streaming. Он основан на искусственной нейронной сети и работает по принципу соревнования.

Задача определения оптимальной структуры нейронной сети решалась с помощью генетического алгоритма. Генетические алгоритмы применимы в поиске параметров нейронной сети, определения ее топологии. Задачу можно модифицировать путем добавления поиска конкретного метода обучения, либо наличие на конечном этапе принципа конгломерата нейронных сетей.

Для реализации алгоритма обучения и прогнозирования была выбрана библиотека TensorFlow от компании Google.

В **третьей главе** было проведено тестирование разработанной системы и анализ результатов ее работы. Для этого выбраны 2 проекта на базе IBM DB2. Первый проект является концентратором данных из различных источников, а также является операционной базой для приложения. Второй проект характеризуется более активным пользовательским вводом и процессом загрузки и пересчета данных онлайн.

Два проекта стали источником данных для аналитической системы. Результаты полученные в процессе тестирования, позволили решить ряд задач в этих проектах, а также найти ряд ошибок. После нахождения различных аномалий они были проанализированы и превращены в правила, которые направлены на диагностику найденных аномалий и их исправление.

В этой главе рассматривается применение инструментов, разработанных на предыдущих этапах для анализа функционирования реальных хранилищ данных, разбор полученных результатов, формирование, и разработка автоматических правил по поиску схожих проблем и предсказаний их появления.

Задача достижения максимальной производительности при пересчете статистики БД была решена с помощью использования инструмента, который позволяет определить наилучшее время для системного процесса в СУБД, срабатывающего при условии систематического несовпадения предполагаемого времени выполнения запросов с прогнозируемым. Этот инструмент позволил уменьшить количество вызовов этого процесса при высокой нагрузке базы, а также увеличил производительность запросов.

Возможность предоставлять актуальную информацию к установленному времени является крайне важной, поэтому была поставлена задача обеспечить загрузку данных к началу рабочего дня. На основе статистики ежедневного поступления данных и времени выполнения запроса на источник, система прогнозирует количество данных в следующей загрузке и на основе этого результата изменяет расписание загрузки. Таким образом пользователи получают максимально актуальные данные из внешних систем к установленному времени, что составляет конкурентное преимущество приложения и положительно сказывается на опыте пользователей.

Для решения задачи ускорения работы хранимых процедур был разработан модуль, который прогнозирует изменения в исходных таблицах и автоматически переписывает запросы, меняя конфигурацию чтобы менеджер базы выполнял запросы эффективно. В модуле регистрируются процедуры, которые отмечают запросы, которые необходимо отслеживать. Парсер определяет все запросы и строит список зависимых таблиц. После чего данный список проверяется в модуле прогнозирования. И если прогноз на ближайшие дни даёт сильный скачок данных, то модуль инициирует обновления кода и использует его до тех пор, пока не будет обнаружен следующий скачок. Так, без использования дополнительных физических комплектующих, код обеспечивает оптимальную производительность, за счёт адаптации под текущий поток данных.

В процессе тестирования системы были выявлены аномалии: изменение объема данных из таблицы провоцирует изменение класса таблицы, чтение данных из таблицы работает медленней, чем в таблицах такого же класса и то, что временные объекты внутри базы регулярно пересоздаются.

По результатам анализа аномалий были сформировано правила: если процент реального изменения данных сильно отличается от физического, то таблица используется неэффективно, и система создаёт отчёт и высылает его команде поддержке для устранения. Второе правило анализирует логи изменения таблицы и выявляет многократную запись данных, что свидетельствует о том, что менеджер базы не понимает как расположены данные. В таком случае необходим или запуск механизма реорганизации данных либо анализ кода.

Третье правило проверяет показатели чтения/записи и скорости в рамках одного класса, чтобы выявит “отстающие” таблицы. Также был разработан дополнительный модуль, который рассчитывает теоретическую размерность записи, на основе средней длинны и сравнивает данный показатель с фактическим в базе данных.

Использование данных правил позволило выявить большое количество устаревших структур, удаление которых позволило повысить производительность отдельных модулей в среднем на 30-40%.

Четвертое правило сопоставляет выполняемые функции в приложении и создание объектов. Данное правило позволяет выявлять неэффективные запросы от приложения, а также даёт рекомендации каких индексов не хватает в приложении, на основании частоты использования.

Использование данного правила позволило повысить общую производительности всей системы на 8%, только за счёт автоматического

выявления недостающих индексов. После исправления неудачных запросов от приложения, показатель производительности повысился до 15%.

ЗАКЛЮЧЕНИЕ

Основные научные результаты диссертации

Основные научные результаты диссертации

1. Предложена архитектура программного средства для сбора и анализа данных о работе реляционной базы данных. Для реализации предложена микросервисная архитектура с событийным шаблоном. Каждый компонент разработан с поддержкой кластерных вычислений при помощи платформы Apache Spark для обеспечения максимальной скорости обработки за счёт вертикального масштабирования. Для центрального хранилища данных используется Kafka для реализации эффективного обмена данными между сервисами. Предложенная архитектура накладывает минимальные издержки на целевую систему. Результаты опубликованы на конференции [2]

2. Рассмотрен ряд алгоритмов кластеризации и прогнозирования, предложен вариант их реализации и комбинирования для решения ряда задач, необходимых для оптимизации и адаптации хранилища данных в условиях реального использования. Для повышения эффективности обучения ИНС был применён генетический алгоритм для определения эффективной модели и статистический подход по подготовке данных, что привело к низкому отклонению при прогнозировании. Использование алгоритмов кластеризации позволило выявить ошибки обучения ИНС и повысить точность обучения [1].

3. Разработан ряд вспомогательных сервисов по сбору системных данных о работе с базы данных и их накоплению, что позволило выявить дополнительные аномалии и расширить функциональность системы.

4. Проведено исследование ряда выявленных аномалий, на основе которых были сформированы правила, способные в автоматическом режиме обнаруживать неисправности такого рода уже после 2 недель эксплуатации системы.

5. Разработанная система прошла внутреннею аттестацию IBA Group, где по результатам комиссии было принято решение внедрить систему в проекты бизнес партнёров IBM и система успешно эксплуатируется в 2 крупных проектах, для повышения качества предоставляемых услуг уже более года.

Рекомендации по практическому использованию результатов

1. Полученные результаты формируют теоретическую и практическую базу для разработки ПО компьютерных систем для решения задач сбора и анализа данных о работе хранилищ данных. Они могут быть использованы для модернизации и дальнейшего развития существующих систем.

2. Разработанные инструменты на основе рассмотренных алгоритмов можно использовать для выявления новых аномалий, их изучения и формализации правил по их обнаружению и ликвидации.

3. Результаты работы могут использовать при подготовке персонала для разработки и обслуживания компьютерных систем, решающих задачи по сбору исторических данных, дальнейшему анализу и построению моделей прогнозирования.

СПИСОК ПУБЛИКАЙИ СОИСКАТЕЛЯ

1. Ильюкевич А. В., Смартфон как инструмент для предоставления персональных рекомендаций стратегии сбалансированного питания / Н. В. Лапицкая, В. В. Трус, А. В. Ильюкевич., Д. Антоненко, А. В. Варфоломеев // Веб-программирование и интернет-технологии WebConf2018: Тезисы докладов 4-й Международной научно-практической конференции (Минск, 14–18 мая 2018 г.). – Минск: БГУ, 2018. –С. 23-24.

2. Ильюкевич А. В., Смартфон как инструмент для предоставления персональных рекомендаций стратегии сбалансированного питания / Н. В. Лапицкая, В. В. Трус, А. В. Ильюкевич., Д. Антоненко, А. В. Варфоломеев // Информационные технологии и системы 2018: Материалы международной научной конференции (Минск, 25 октября 2018 г.). – Минск: БГУИР, 2018. –С. 17-20.