Д.С. Синюков, А.Д. Данилов, О.Я. Кравец

УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ С ОПЕРАТИВНЫМ КОНТЕНТОМ НА ОСНОВЕ РАСПРЕДЕЛЕННОГО КЭШИРОВАНИЯ

Монография

Воронеж
Издательство «Научная книга»
2023

УДК 004.5
ББК 32.973
С 38

Рецензенты:

Мельник Э.В., доктор технических наук, ФГБУН «Федеральный исследовательский центр Южный научный центр Российской академии наук», главный научный сотрудник, заведующий лабораторией информационных технологий и процессов управления (г. Ростов-на-Дону);

Лавлинский В.В., доктор технических наук, доцент, ФГКВОУ ВО «Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина»», старший научный сотрудник научно-исследовательского центра образовательных и информационных технологий (г. Воронеж)

Л 74 **Сюнюков, Д.С.** Управление транзакциями с оперативным контентом на основе распределенного кэширования: Монография/ Д.С. Синюков, А.Д. Данилов, О.Я. Кравец. – Воронеж: Издательство «Научная книга», 2023. – 152 с.

ISBN 978-5-907328-31-0

Монография посвящена проблеме в облачных средах, связанной с алгоритмами разделения и перенаправления запросов между клиентами, межобъектными интерфейсами или облаком. Особенностью механизма распределения данных о специальных транзакциях в реальном времени является использование распределенного кэша.

Монография предназначена для научных и инженерных работников, интересующихся вопросами технической диагностики, а также может быть полезна студентам, магистрантам и аспирантам направления «Информатика и вычислительная техника».

Рис. 31. Табл. 8. Библиогр.: 175 назв.

УДК 004.5
ББК 32.973
С 38

ISBN 978-5-907328-31-0

Оглавление

Введение

Актуальность темы. Взрывной рост потребностей общества в программном обеспечении, взаимодействующем с облачными хранилищами (СУБД), породил множество методов его проектирования. Большой вклад в развитие методов проектирования внесли Мурзин Ф.А., Сахаров Д.В., Царегородцев А.В., Dar S., Jaziri W., Stankovic J.A., Zhang Q., Zhao Z. В теоретическом плане ряд методов управления транзакциями в облачных средах сводится к оптимизации выбора направлений и локальных баз в распределенной системе внутри облака. Разумной идеей кажется создание алгоритма локального кэширования хронологически запрошенных данных транзакций, обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях. Особенностью механизма распределения данных о специальных транзакциях в реальном времени должно являться использование распределенного кэша.

Несмотря на глубоко развитую теорию фиксации транзакций, существуют проблемы в облачных средах, связанные с алгоритмами разделения и перенаправления запросов между клиентами, межобъектными интерфейсами или облаком. В данной области ведущие исследователи Болодурина И.П., Борисенко О.Д., Никульчев Е.В., Zatwarnicki K. развили множество походов. Ключевые показатели для этих подходов, к сожалению, не учитывают интегрированную информацию о состоянии кэширования, предполагаемом размере данных и пропускной способности. Необходима метрика, учитывающая эту информацию, для минимизации задержки передачи данных транзакции.

СУБД реального времени часто проходит через периоды перегрузки после неожиданного поступления пользовательских транзакций. В такие периоды транзакции с большей вероятностью пропускают свои предельные сроки, и это напрямую влияет на QoS (качество

обслуживания), предоставляемое пользователям. Таким образом, необходимы дополнительные исследования в области развития протоколов планирования транзакций в СУБД реального времени, учитывающих не только временные ограничения транзакций, но и критерии, установленные пользователями базы данных, обеспечивающие корректное определение приоритетов выполняемых транзакций. Вытекающая отсюда проблема – необходимость создания архитектуры СУБД реального времени, отличающейся применением модифицированного протокола планирования транзакций и улучшающей качество предоставления услуг пользователям при реализации управления с обратной связью.

Таким образом, актуальной является задача разработки математического и программного обеспечения управления транзакциями с оперативным контентом на основе распределенного кэширования.

Целью работы является разработка математического и программного обеспечения управления транзакциями с оперативным контентом на основе распределенн ширования с использованием модифицированного протокола планирования транзакций.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать механизм распределения данных о специальных транзакциях с оперативным контентом, обеспечивающий минимизацию времени передачи данных.

2. Разработать алгоритм локального кэширования хронологически запрошенных данных транзакций, обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях

3. Создать алгоритм разделения и перенаправления запросов между клиентами, межобъектными интерфейсами или облаком, дополнительно учитывающий интегрированную информацию о состоянии кэширования, предполагаемом размере данных и пропускной способности.

4. Модифицировать протокол планирования транзакций в СУБД реального времени для учета не только временных ограничений транзакций, но и критериев, установленных пользователями базы данных.

5. Предложить архитектуру СУБД реального времени с использованием модифицированного протокола планирования транзакций для улучшения качества предоставления услуг пользователям при реализации управления с обратной связью.

6. Создать программный комплекс для проверки динамических связей технологических схем и баз данных на основе системы кодирования Kraftwerk Kennzeichen System.

7. Создать компоненты распределенной специальной информационно-вычислительной системы управления с унифицированными механизмами межмодульного взаимодействия для оптимизации состава резервного оборудования и исключения программных конфликтов при обмене данными.

Объект исследования: распределенные системы управления базами данных.

Предмет исследования: математическое и программное обеспечение управления транзакциями с оперативным контентом на основе респределенного кэширования.

Методы исследования. При решении поставленных задач использовались теория вероятностей, теория принятия решений, а также методы объектно-ориентированного программирования.

Научная новизна работы. Получены следующие результаты, характеризующиеся научной новизной:

1. Механизм распределения данных о специальных транзакциях с оперативным контентом, отличающийся распределенным кэшированием в гетерогенных межобъектных интерфейсах в реальном масштабе времени, обеспечивающий минимизацию времени передачи данных.

2. Алгоритм локального кэширования хронологически запрошенных данных транзакций, отличающийся учетом собственной емкости межобъектного интерфейса и применением политикой вытеснения давно неиспользуемых данных, обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях.

3. «Жадный» алгоритм разделения и перенаправления запросов между кдиентами, межобъектными интерфейсами или облаком, отличающийся интеграцией информации о состоянии кэширования, предполагаемом размере данных и пропускной способности и обеспечивающий минимизацию задержки передачи данных транзакции.

4. Модифицированный протокол планирования транзакций в СУБД реального времени, учитывающий не только временные ограничения транзакций, но и критерии, установленные пользователями базы данных, обеспечивающий корректное определение приоритетов выполняемых транзакций.

5. Архитектура СУБД реального времени, отличающаяся применением модифицированного протокола планирования транзакций и улучшающая качество предоставления услуг пользователям при реализации управления с обратной связью.

Теоретическая и практическая значимость исследования заключается в разработке математического и программного обеспечения управления транзакциями с оперативным контентом на основе распределенного кэширования с использованием модифицированного протокола планирования транзакций, а также информационного и программного обеспечения для проверки динамических связей технологических схем и баз данных, использующего систему кодирования Kraftwerk Kennzeichen System.

Основное содержание работы

В первой главе исследуются особенности разработки математического и программного обеспечения математического и программного обеспечения управления транзакциями с оперативным контентом на основе распределенного кэширования с использованием модифицированного протокола планирования транзакций, и анализируется современное состояние проблемы управления ими. Отмечено, что повысить эффективность управления транзакциями можно путем модификации распределения данных о них в сети, создания алгоритмов и протоколов планирования транзакций с учетом дополнительных параметров, определяющих качество обслуживания. Исходя из этого, проанализированы существующие подходы к решению данных задач. Результат данного анализа потребовал формализации данных задач, а также алгоритмизации их решения с учетом особенностей, отраженных на рис. 0.1.

Вторая глава посвящена разработке новых алгоритмов и методов управления транзакциями в гетерогенных объектах распределенной сети.

Предлагается механизм распределения данных о специальных транзакциях в реальном времени на основе распределенного кэша.

Распределенный кэш в основном состоит из двух частей: одна - кэш межобъектного интерфейса (МОИН), а другая - кэш небольшой емкости конечного клиента. Облачная платформа отвечает за хранение всей информации о транзакциях на локальном клиенте, а межобъектный интерфейс отвечает за ускорение передачи данных о транзакциях и поддержание информации распределенного кэша. Представлен механизм распределения данных о специальных транзакциях с оперативным контентом, отличающийся распределенным кэшированием в гетерогенных межобъектных интерфейсах в реальном масштабе времени, обеспечивающий минимизацию времени передачи данных. Разработан

алгоритм локального кэширования хронологически запрошенных данных транзакций, отличающийся учетом собственной емкости МОИН и применением политикой вытеснения давно неиспользуемых данных и обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях.

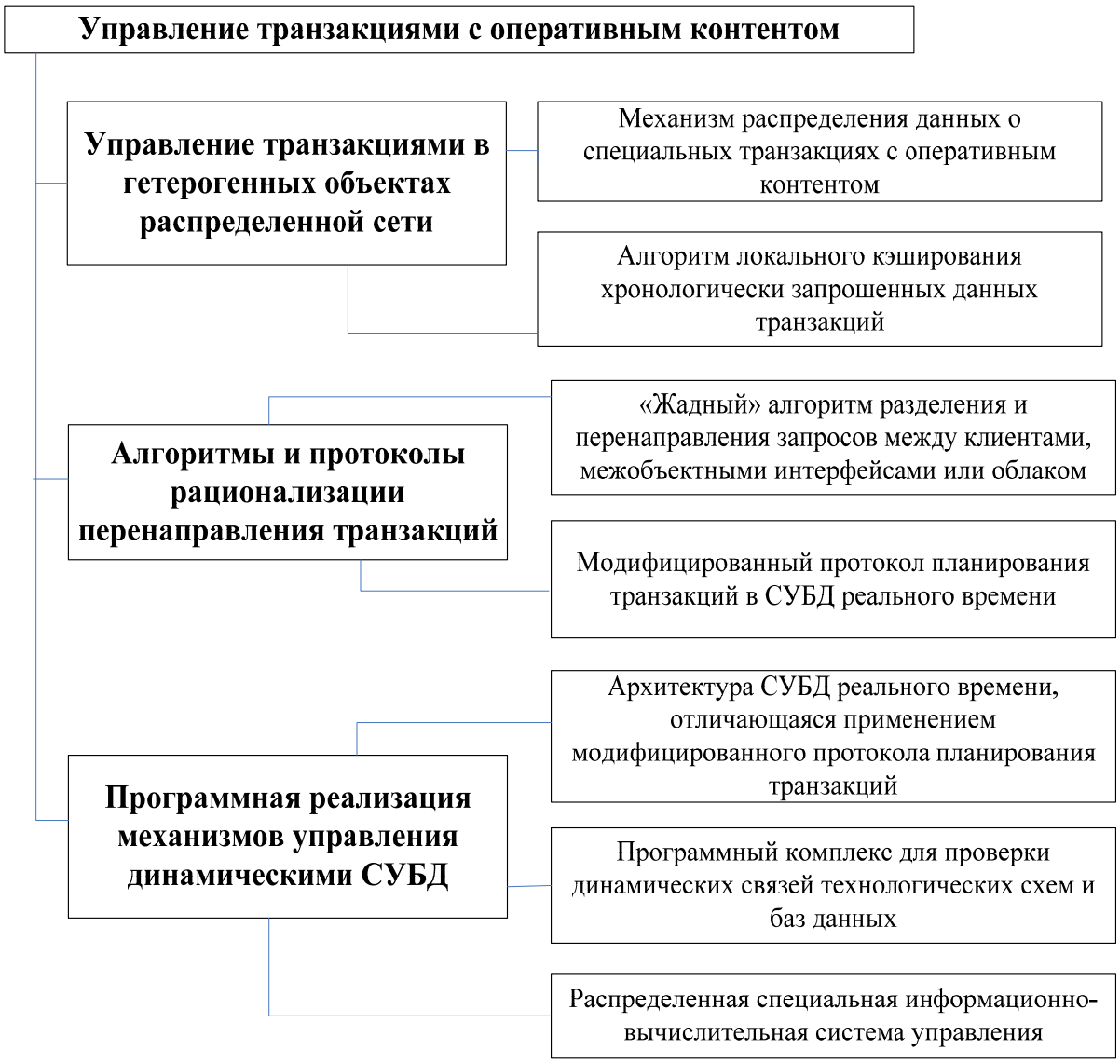

Рис. 0.1. Дизайн исследования

Третья глава посвящена описанию алгоритмов и протоколов рационализации перенаправления транзакций.

Для борьбы с фазами нестабильности в СУБД реального времени (РВ) из-за непредсказуемого поступления транзакций пользователей

предложена архитектура, основанная на обратной связи, которая рассматривается как базовая архитектура для управления QoS в СУБД РВ.

В предлагаемой архитектуре рассматриваются транзакции с жестким предельным сроком, когда, если транзакция пропустит свой предельный срок, она будет прервана и станет бесполезной для системы. Будем различать два типа транзакций в реальном времени в зависимости от объекта данных, к которому осуществляется доступ: обновленные транзакции и транзакции пользователей. Обновленные транзакции - это те, которые периодически выполняются для обновления данных в реальном времени. Однако пользовательские транзакции, которые являются апериодическими, состоят из набора операций чтения как с данными в реальном времени, так и с данными, не относящимися к реальному времени, и операций записи только с данными, не относящимися к реальному времени.

Основным критерием эффективности, рассматриваемым в предлагаемой архитектуре, является уровень удовлетворенности пользователей системы. Он представляет коэффициент успешности (SR) транзакций, которые квалифицируются как критические для пользователей. SR определяется как параметр QoS, измеряющий процент транзакций, которые соответствуют своим срокам. Представлен модифицированный протокол планирования транзакций в СУБД реального времени, учитывающий не только временные ограничения транзакций, но и критерии, установленные пользователями базы данных, обеспечивающий корректное определение приоритетов выполняемых транзакций, а также разработана архитектура СУБД реального времени, отличающаяся применением модифицированного протокола планирования транзакций и улучшающая качество предоставления услуг пользователям при реализации управления с обратной связью.

В главе 4 представлены особенности программной реализации механизмов управления динамическими СУБД, основанные на результатах предшествующих глав.

Описан программный комплекс для проверки динамических связей технологических схем и баз данных, использующий систему кодирования Kraftwerk Kennzeichen System, позволяющий получать идентификаторы для типовых объектов энергоблока и выявляющий идентификаторы, которые не соответствуют базам данных.

Создана распределенная специальная информационно-вычислительная система управления, в которой шлюзовое прокладное программное обеспечение предусматривает до 40 протоколов обмена со смежными модулям, а также позволяет оптимизировать состав резервного оборудования и избежать программных конфликтов при обмене данными.

В заключении представлены основные результаты работы.

1. Управление транзакциями в гетерогенных объектах распределенной сети

1.1. Проблемы управления транзакциями в гетерогенных объектах распределенной сети

Передача данных о специальных транзакциях предъявляет чрезвычайно высокие требования к передаче в режиме реального времени. Однако объем данных для таких транзакций продолжает расти. Традиционный механизм сквозной передачи данных стал трудно соответствовать требованиям управления в режиме реального времени. В статье предлагается эффективный механизм передачи данных о специальных транзакциях, основанный на кэшировании в распределенной сети. Внедряя эластичное пространство для хранения на межобъектном интерфейсе, обеспечивая интеллектуальную передачу данных о транзакциях в облаке и внедряя пассивное кэширование на клиенте, он эффективно решает требования к передаче данных о транзакциях в реальном времени.

Система оперативного управления большими базами данных - это система с частой обработкой и чрезвычайно высокой пропускной способностью обмена данными.

Скорость обработки системы напрямую связана с эффективностью работы и оценкой клиентами [1.1]. В объективных условиях вся система глобальной сети должна поддерживать возможности быстрого реагирования в режиме реального времени в то же время в условиях большого трафика, чтобы обеспечить бесперебойную работу всей инфокоммуникационной среды.

Однако в традиционном режиме передачи данных из облака клиенту, поскольку общедоступный Интернет не может быть гарантирован, максимально предельная задержка передачи не может быть гарантирована,

а пользовательский интерфейс оставляет желать лучшего. Для большого объема данных в реальном времени требуется очень большое количество данных в реальном времени [1.2, 1.3]. Поэтому традиционная архитектура не подходит. Требуется новый и эффективный механизм передачи.

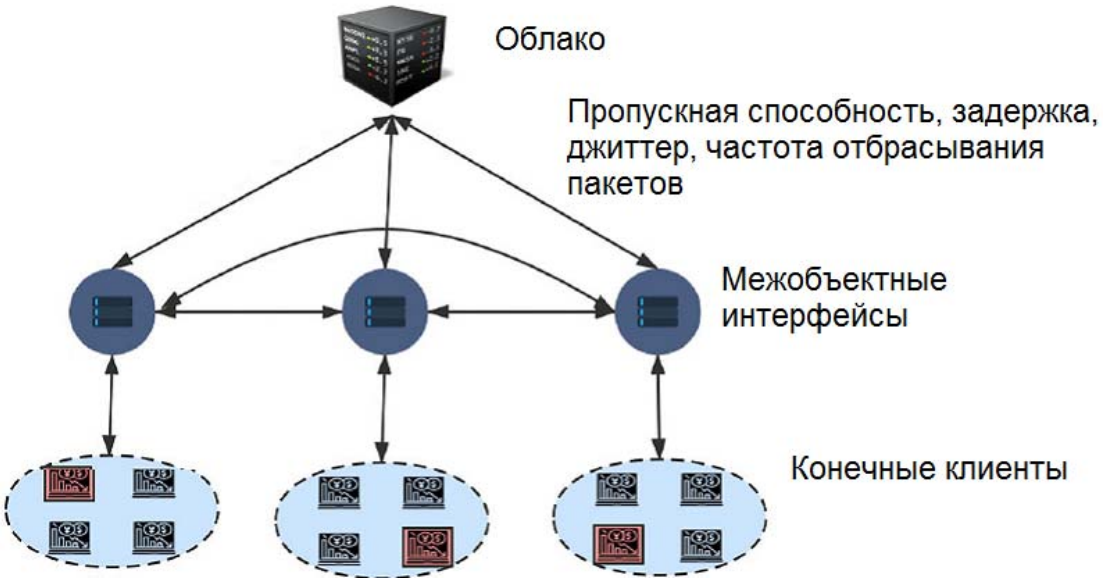

Рис. 1.1. Распределенная передача данных в реальном времени на основе межобъектного интерфейса

С развитием технологий было развернуто большое количество периферийных платформ, нарушающих традиционную сквозную сетевую архитектуру передачи данных и открывающих возможности для решения будущих задач сетевой передачи с чрезвычайно низкой задержкой [1.4]. По сравнению с традиционным облаком, межобъектный интерфейс (МОИН) ближе к пользователю.

Хотя его вычислительные возможности и возможности хранения данных значительно уступают облачным, межобъектный интерфейс обладает очевидными преимуществами в вычислительной мощности и хранении по сравнению с клиент-серверной идеологией [1.5]. В то же время межобъектный интерфейс может в режиме реального времени отслеживать динамическое состояние сети от конечных пользователей до

облака и принимать глобальные оптимальные решения для конечных клиентов [1.6]. Кроме того, благодаря сотрудничеству между межобъектными интерфейсами требования к передаче данных в облако могут быть сведены к минимуму.

Предлагается механизм распределения данных о специальных транзакциях в реальном времени на основе распределенного кэша. Распределенный кэш в основном состоит из двух частей: одна - кэш межобъектного интерфейса, а другая - кэш небольшой емкости конечного клиента.

1. Облако анализирует популярность различных акций в соответствии с историческими запросами “от-до” соответствующей платформы МОИН. Основываясь на вычислении возрастающей популярности и динамическом состоянии сети, облако отправляет более популярные данные о транзакциях на соответствующий межобъектный интерфейс, когда сеть имеет достаточную доступную пропускную способность.

2. Межобъектный интерфейс сообщает информацию о своих кэшированных данных другим МОИН, сохраняя при этом информацию о кэшированном содержимом локальных клиентов. Получив запрос от клиента, межобъектный интерфейс выбирает оптимальную стратегию передачи в соответствии со статусом сети и информацией о распределенном кэше. Он может перенаправить запрос в локальный кэш, другие терминалы, другие МОИН или удаленное облако. Цель состоит в том, чтобы достичь наименьшего времени завершения передачи.

3. Клиент кэширует соответствующие хронологически запрошенные данные транзакции в зависимости от своей емкости в соответствии с политикой вытеснения давно неиспользуемых данных (LRU - Least Recently Used [2.7]). Для того чтобы оценить эффективность схемы, произведено сравнение время выполнения запросов на данные о специальных транзакциях. Результаты показывают, что предложенные

схема значительно улучшила время завершения передачи данных, что хорошо решает требования к передаче данных о специальных транзакциях в режиме реального времени.

1.2. Задачи управления транзакциями для повышения качества обслуживания в СУБД реального времени

Приложения все чаще характеризуются манипулированием большими объемами данных и временными ограничениями, в связи с которыми представляются данные и методы лечения. СУБД РВ (Системы управления базами данных в реальном времени) являются подходящим формализмом для обработки таких приложений. Однако СУБД РВ часто проходит через периоды перегрузки после неожиданного поступления пользовательских транзакций. В такие периоды транзакции с большей вероятностью пропускают свои предельные сроки, и это напрямую влияет на QoS (качество обслуживания), предоставляемое пользователям. Таким образом, задача заключается в том, чтобы предложить модифицированный протокол планирования для оптимизации выполнения транзакций без превышения их предельных сроков. Он заключается в назначении приоритетов транзакциям на основе как их предельных сроков, дат прибытия, так и уровней приоритета, определенных пользователями. Кроме того, экспериментально показано, что такой подход может максимизировать количество успешных транзакций, в частности тех, которые классифицируются как критические для пользователей. Полученные результаты сравниваются с традиционными подходами к планированию.

В современных приложениях реального времени, связанных с информационными системами, основная цель состоит в том, чтобы

эффективно обрабатывать большие объемы данных, соблюдая при этом временные ограничения, накладываемые на них. Действительно, обычная система управления базами данных (СУБД) подходит для эффективного управления большими объемами данных. Однако такая система не учитывает временные ограничения. С другой стороны, системы реального времени имеют соответствующие механизмы для управления этими временными ограничениями, но не для больших объемов данных. Таким образом, появились СУБД реального времени (СУБД РВ).

Можно определить СУБД РВ как комбинацию обычной СУБД и системы реального времени. Ее цель состоит как в обработке больших объемов данных при сохранении их логической согласованности, так и в обеспечении соблюдения временных ограничений [1.7]. Когда транзакции, отправляемые пользователями, поступают с разной частотой, такая система переживает периоды нестабильности, в течение которых система становится перегруженной. В периоды перегрузки наблюдается нехватка ресурсов, так что транзакции существенно превышают свои предельные сроки. Поэтому необходимо лучше управлять имеющимися ресурсами, чтобы обеспечить хорошее качество обслуживания (QoS), удовлетворяя наиболее важные транзакции, которые сталкиваются с необходимыми ограничениями.

Поскольку такое управление необходимо для любого процесса выполнения транзакций, планирование рассматривается в нескольких исследовательских проектах. Это полезно для определения времени, в течение которого транзакции должны выполняться в системе. При получении транзакции планировщик решает запустить ее немедленно, отложить или полностью отклонить транзакцию. В контексте реального времени такой механизм значительно помог бы транзакциям справиться с их временными ограничениями, а затем улучшить качество обслуживания, предоставляемое пользователям. Широко обсуждалось управление

качеством обслуживания в СУБД РВ. Работы в основном основаны на методах планирования управления с обратной связью [1.8, 1.9]. Например, в [1.10] использовалась архитектура управления с обратной связью для учета полученных данных в реальном времени. Эта же архитектура была адаптирована в [1.11] для управления качеством обслуживания в мультимедийных системах. Такая же идеология используется в [1.12] для управления QoS в пространственных больших данных в реальном времени.

В работе решается задача повышения качества обслуживания в СУБД РВ, удовлетворяя максимум важных запросов пользователей. Предлагается протокол, объединяющий три критерия для определения приоритетов транзакций. Это позволяет, в частности, учитывать уровни критичности транзакций, определенные заинтересованными пользователями. Таким образом, гарантируется, что максимальное количество транзакций, квалифицируемых как критически важные для пользователей, будет выполнено без нарушения их сроков. После этого применяется предлагаемый протокол в архитектуре СУБД РВ, основанной на планировании управления с обратной связью, которое подходит для поддержания надежного поведения системы в периоды нестабильности.

1.3. Разнородная рабочая нагрузка и обработка транзакций

Сегодня облачные вычисления являются наиболее широко используемой системой из-за ряда преимуществ для конечных пользователей. В ИТ (информационных технологиях) организациях облачные вычисления являются наиболее важной областью для работы. Существуют различные типы услуг, такие как SaaS, IaaS и PaaS, предоставляемые облачными вычислениями в зависимости от потребностей конечных пользователей. Вместо того чтобы использовать собственные ресурсы для хранения и управления данными, организации

начали использовать системы хранения данных облачных вычислений. Облачное хранилище (например, Amazon S3) становится популярной услугой, и большинство предприятий переносят свои рабочие нагрузки на данные в облако. Популярность системы облачных вычислений резко возросла, поскольку она арендует вычислительные ресурсы, выставляет счета с оплатой по мере использования и мультиплексирует множество пользователей на одной и той же физической инфраструктуре. Пользователям облачных вычислений предоставляется иллюзия бесконечных вычислительных ресурсов, так что уровень потребления ресурсов по требованию может быть увеличен или уменьшен.

Облачные вычисления предлагают видение практически бесконечного пула вычислительных, хранилищ и сетевых ресурсов, где могут быть развернуты приложения. Это популярное решение для предоставления ресурсов по требованию и динамического предоставления ресурсов. Подготовка и обслуживание облачных ресурсов выполняются с помощью методов управления ресурсами (RM). Методы RM отвечают за отслеживание свободных ресурсов и назначение ресурсов из свободного пула входящим задачам. Наряду с растущими требованиями современных приложений и рабочих нагрузок облачные вычисления приобрели известность во всей ИТ-индустрии. Современные веб-приложения растут по разным параметрам, таким как количество пользователей, сложность, объем данных, и многие люди подключаются к Интернету с помощью различных устройств.

Современные веб-приложения требуют большей производительности и емкости от своих систем хранения и управления данными, поэтому обычно используются системы баз данных. По мере того как веб-приложения становятся все более популярными, они могут демонстрировать экспоненциальный рост со стороны пользователей по всему миру. Таким образом, для сокращения времени отклика многие веб-

приложения используют центры обработки данных, чтобы предоставить пользователям доступ к центру обработки данных. Веб-приложения, которые имеют дело с облачными системами хранения данных, содержат разнородную рабочую нагрузку для развертывания в облаке.

С этой разнородной рабочей нагрузкой необходимо обращаться осторожно, иначе могут возникнуть такие проблемы, как длительная задержка планирования, нехватка ресурсов для задач с низким приоритетом, которые могут значительно снизить производительность приложения. Для обработки этой рабочей нагрузки характеристика гораздо важнее. При характеристике рабочей нагрузки разнородная рабочая нагрузка делится на несколько классов задач со схожими характеристиками с точки зрения ресурсов и целей производительности. Важно учитывать неоднородность рабочей нагрузки. Рабочая нагрузка обычно состоит из различных приложений с различными приоритетами и потребностями в ресурсах. Если не учитывать неоднородную рабочую нагрузку, это приведет к длительной задержке планирования, истощению, влияя на производительность приложения.

Современные приложения сталкиваются с такими проблемами, как характеристика рабочей нагрузки, распределение ресурсов и безопасность. Для удовлетворения этих потребностей предлагается новая структура, которая взаимодействует с гетерогенными системами баз данных и обеспечивает характеристику рабочей нагрузки с использованием алгоритма кластеризации K-means, распределение ресурсов путем предоставления виртуальных машин посредством запроса в облако и безопасность с использованием алгоритма AES для шифрования и дешифрования.

В работе транзакции рассматриваются в виде гетерогенных файлов от пользователей в качестве входных данных и эффективно развертываются в облачной системе хранения данных и в гетерогенных

системах баз данных в качестве выходных данных. Транзакционные операции, такие как вставка, обновление, удаление, выполняются с данными пользователей в облачных системах хранения и в гетерогенных системах баз данных. Операции транзакционных запросов выполняются в системах баз данных. Предлагаемая система пытается представить новый подход к более безопасной, энергоэффективной, масштабируемой системе управления транзакциями, основанной на терминологии и методах планирования ресурсов в гетерогенных распределенных системах баз данных.

В работе рассматривается проблема управления транзакциями, которая заключается во взаимодействии с гетерогенными системами баз данных, а также в распределении и планировании вычислительных ресурсов с учетом характеристик рабочей нагрузки путем обеспечения безопасности пользовательских данных.

Оценка производительности предлагаемой системы проводится на основе изменяющихся рабочих нагрузок и с точки зрения пропускной способности, энергоэффективности, загрузки процессора, времени планирования, времени отклика. Производительность системы измеряется и сравнивается с планированием на основе контейнеров (CBS), механизмов динамического распределения ресурсов для энергосбережения (DPRA) и технологии управления энергетическими ресурсами с учетом SLA (SLA).

1.4. Задачи поиска и устранения неисправностей в базе данных

Развитие технологий и применение отраслевых инноваций на практике изменили рабочий процесс, связанный с устранением неисправностей в системах баз данных. Использование удаленного аппаратного оборудования существенно облегчает создание и

развертывание нового программного обеспечения, использующего базу данных. Традиционно независимые поставщики программного обеспечения развертывают отдельные копии приложений в оперативной среде. В настоящее время большинство таких вендоров переходят на другую модель. Они размещают комбинированный программный сервис, который работает для их клиентов коллективно, в публичной облачной среде, используя вместо этого тип "платформа как услуга". Таким же образом, ответственность за обслуживание теперь передается от конечного клиента поставщику услуг компьютерного облака. Однако, в то же время, вводится новый уровень абстракции, что серверы баз данных больше не доступны для разработчиков и администраторов баз данных. Это существенно уменьшает некоторые возможности поддержки производительности на должном уровне.

Другим очень важным аспектом является масштаб «платформы/базы данных как службы». Традиционно поиск и устранение неисправностей в базе данных выполнялся для каждого клиента и для каждого конкретного случая. В настоящее время такая высокотребовательная задача в облачном масштабе требует, как значительных вычислительных, так и инженерных мощностей.

1.4.1. Описание проблемы

Системы управления реляционными базами данных превратились в крупные, часто очень сложные программные продукты. Эти системы позволяют пользователям проектировать произвольно сложную логику, что иногда приводит к низкой производительности приложений. Ранее такие проблемы решались путем сочетания хорошего выбора дизайна реляционных баз данных во время разработки, а также путем устранения неполадок с производительностью администраторами баз данных в процессе разработки. Однако, более сложные системы часто требуют более

глубоких человеческих знаний для их отладки и настройки. Имея в виду, что каждое приложение отличается, для того, чтобы эффективно управлять такими системами, чтобы они могли работать адекватно, как правило, требуется ручная настройка. В [1.14] сообщалось, что эксплуатационные расходы на крупномасштабную СУБД составляют почти 50% от общих инвестиций, в то время как администраторы СУБД тратят около 25% своего времени на ручную настройку.

1.4.2. Поддержка мониторинга

Существует несколько коммерческих решений, которые обеспечивают расширенную поддержку мониторинга через набор пользовательских интерфейсов, позволяющих администратору базы данных отслеживать поведение базы данных. Эти решения основаны на расширении возможностей администраторов или консультантов баз данных для эффективного управления производительностью систем. Тем не менее, многие проблемы с производительностью выходят за рамки навыков среднего администратора баз данных. Даже для высококвалифицированных администраторов может потребоваться несколько часов, чтобы определить проблему, найти правильное решение и применить соответствующие исправления. При применении к системам хостинга общедоступных облачных баз данных с миллионами баз данных и приложений, размещенных в одной инфраструктуре, существующие методы просто не могут справиться с этой задачей.

В то же время первопричина наблюдаемых проблем может быть на стороне приложения или может быть вызвана неоптимальным использованием реляционной базы данных. Эта закономерность также наблюдалась в работах [1.15] и [1.16]. В таких случаях дальнейший анализ показывает либо отсутствие проблем со стороны системы, либо чистое увеличение нагрузки с точки зрения приложения. Для того, чтобы

отличить реальные проблемы от проблем, вызванных со стороны пользователя, требуется исследование со стороны инженеров поддержки, которое занимает значительное время и ресурсы в облачной среде. Было бы полезно иметь такую систему, которая могла бы различать эти ситуации и давать соответствующие советы инженерам поддержки, чтобы сократить время и стоимость эксплуатации. Следовательно, раннее обнаружение проблем и надлежащее оповещение позволяют пользователям сосредоточиться на исправлении фактической проблемы.

В области поиска и устранения неисправностей с реляционными базами данных существует несколько коммерческих решений, поставляемых Idera, Sentry, Dell, Amazon и Oracle.

1.4.3. Коммерческие решения

SQL Sentry Performance Advisor [1.17] - это решение для отслеживания и обслуживания инфраструктуры SQL-сервера. Данный продукт предназначен для локальной среды, но некоторые аспекты могут быть использованы и для облака. Пользователь может отслеживать различные метрики и все ключевые параметры SQL сервера. Можно исследовать, что происходило в последнее время, и настраивать оповещения на основе вручную заданных пороговых значений. Dell Spotlight [1.18] предоставляет доступ к интуитивно понятному пользовательскому интерфейсу с возможностью установки оповещений, позволяющих осуществлять непрерывный мониторинг систем на базе SQL-сервера.

Немного другое решение, называемое Idera SQL Diagnostic Manager [1.19], позволяет использовать некоторые базовые методы диагностики наряду с возможностями мониторинга. С помощью этого решения можно также отслеживать состояние базовой аппаратной инфраструктуры, в которой размещены серверы баз данных. В то же время, можно

отслеживать состояние на высоком уровне, в котором можно контролировать выполнение всех запросов или обрабатываемую процедуру. Используя события, обрабатываемые SQL-сервером, этот менеджер диагностики выявляет, какой именно запрос является причиной блокировки. Одним из ограничений данного решения является то, что пользователю необходимо указать набор параметров и их пороговые значения, чтобы сигнализировать системе о возможном возникновении проблемы. Diagnostic Manager содержит функционал, называемый SQL Doctor, который является инструментом для выявления и решения проблем производительности. Также этот инструмент дополнен Web-интерфейсом, который позволяет пользователям отслеживать производительность своего SQL-сервера из любой точки мира.

Oracle Enterprise Manager 13c Cloud Control [1.20] в основном ориентирован на настройку, управление и поддержку корпоративных облачных решений. Двумя основными компонентами, представляющими интерес с точки зрения данной работы, являются Automatic database diagnostic monitor [1.21] и Automatic SQL tuning advisor [1.22], которые направлены на обнаружение и оптимизацию рабочей нагрузки.

Как уже упоминалось ранее, в компании Azure уже разработаны утилиты для автонастройки баз данных SQL, такие как советники баз данных [1.23], [1.24]. Они также включают в себя плановую автоматическую коррекцию, основной целью которой является автоматическая оптимизация и улучшение производительности всех БД на платформе. По этой причине в Microsoft утверждают, что СУБД Azure SQL является "первым интеллектуальным облачным сервисом баз данных" [1.25].

Наряду с решениями от крупных корпораций существует рынок с набором более мелких продуктов и скриптов, которые могут быть использованы в качестве отправной точки в исследованиях для обычного

администратора баз данных. Популярный инструментарий в виде базовой экспертной системы, ориентированной на сервер Microsoft SQL, выполнен в [1.26]. После развертывания на конкретном экземпляре SQL-сервера этот инструментарий запускает отслеживание определенной статистики. Таким образом, администратор БД может точно определить, что происходит с БД во время выполнения.

1.4.4. Академические решения

В отличие от коммерческих решений, академические предложения научного сообщества в основном ориентированы на реляционные базы данных с открытым исходным кодом.

С введением парадигмы облака, которая поддерживается сложной инфраструктурой, развернутой по всему миру, вновь возникла проблема обслуживания сети, поиска и устранения неисправностей. На этот раз в несколько иной форме. Раньше проблемы возникали только в сложной сетевой инфраструктуре. Однако, поскольку приложения и сервисы развертываются в удаленных местах, проблема поиска и устранения неисправностей и обслуживания приложения становится сегодня трудоемкой и дорогостоящей операцией.

Одна из первых реализаций по поиску неисправностей относится к 1988 году, когда поиск неисправностей в инфраструктуре представлял собой серьезную проблему. Решение было построено на базе экспертной системы, чтобы помочь пользователю, не имеющему опыта работы в сетевом домене. Наблюдаемая проблема и сложность, возникающая при поиске и устранении неисправностей, а также общие принципы, применяемые в этом подходе, применимы к любому виду поиска и устранения неисправностей сложных систем и не являются специфичными для сетевого домена. Хотя экспертные системы зародились несколько десятилетий назад, они не устарели и до сих пор считаются эффективным

методом поиска и устранения неисправностей. Аналогичная экспертная система была разработана с целью поиска и устранения неисправностей в сети AT&T [1.15], в основу которой была положена экспертная система, основанная на правилах [1.14].

Автоматизация процесса решения заявленных пользователями проблем с целью снятия давления со стороны инженеров поддержки была решена в [1.16] с помощью методов машинного обучения. Когда речь зашла об использовании машинного обучения в реляционных базах данных, интересная попытка была предпринята в [1.14] с главной целью найти правильный набор параметров конфигурации для открытых реляционных баз данных, таких как MySQL и PostgreSQL.

Описание проблемы и поиск путей устранения неполадок с производительностью базы данных - это сложная задача, еще более подчеркивающая то, насколько требовательным может быть понимание и выполнение анализа первопричин проблемы. Масштабы этой проблемы еще выше и значительно сложнее в среде, где размещены сотни тысяч баз данных с различными рабочими нагрузками, например, в облачной системе управления реляционными базами данных.

О последних усилиях в этой области в масштабе облаков сообщают исследователи из CISCO Tetration Analytics. Они построили механизм анализа декларативных корневых причин для данных временных рядов под названием ExplainIt [14]. В наблюдаемом облачном сервисе прибор для мониторинга непрерывно собирает миллионы событий из различных сервисов. По сути, это те же самые сигналы, что и те, которые публикуются и передаются SQL-сервером. Работа со схожими, но разными временными рядами и нахождение фактической корреляции между ними считаются довольно сложной задачей.

1.4.5. Системы автоматического поиска и устранения неисправностей

Система автоматического поиска и устранения неисправностей в базе данных (ADTS) состоит из трех основных компонентов:

- научные модели данных
- экспертная система
- инфраструктура

Каждая база данных генерирует различный набор статистики. Эти данные могут быть использованы для наиболее четкого понимания характера нагрузки, выполняемой на БД. Компонент "Научные модели данных" содержит набор различных правильно спроектированных научных моделей статистических данных, предназначенных для обнаружения специфического поведения, основанного на определенном наборе сигналов. Применяемый подход с использованием традиционных научных моделей статистических данных имеет основное преимущество в условиях, когда сотни различных сигналов поступают из более чем миллиона баз данных, в то время как помеченный набор данных крайне узок и имеет менее пятисот наблюдаемых случаев. Аналогичный подход с набором классификационных моделей в сложной среде был рассмотрен в [1.28]. Несколько иной подход был представлен в [1.27], где основное внимание уделялось автоматическому выявлению проблемы. Здесь проблемные периоды времени определяются пользователем системы, а не самой системой. Ответственность системы в этом случае заключается в поиске корреляции с другими сигналами и предоставлении потенциальных путей для дальнейшего исследования.

Как и во многих современных крупных системах, основанных на искусственном интеллекте, существует необходимость фильтрации выходных данных, получаемых с помощью научных моделей данных, для того чтобы определить наиболее точную причину неполадок. Для этого

была разработана экспертная система, а ее правила были тщательно проработаны и оценены. Экспертная система, основанная на правилах, принимает окончательное решение о том, какие сигналы считать важными и публиковать для пользователей, а что не имеет отношения к делу и должно быть проигнорировано. Все разработанные модели, а также экспертная система работают поверх облачной инфраструктуры Microsoft Azure.

Модели для автоматического поиска и устранения неисправностей в БД

Для достижения наилучших результатов обнаружения и понимания поведения были определены два типа статистических моделей:

- Общие модели
- Классифицированные модели

Общие модели предназначены для обнаружения непредвиденных ситуаций в поведении системы баз данных. Влияние неожиданного поведения представлено параметром серьезности (Severity), атрибутом, характеризующим влияние дефекта на работоспособность системы. Данные научные модели предназначены не для получения высокоточных результатов, а скорее для высокой детализации действий внутри системы. По сути, это означает, что система будет обнаруживать всевозможные аномалии, как безопасные, так и вредоносные. Например, при таком подходе обнаружение происходит как при чистом увеличении нагрузки со стороны приложения, так и при серьезном снижении производительности, вызванной различными факторами. Эти две ортогональные ситуации дополнительно анализируются и дифференцируются системой на основе моделей отсечения серьезности и классификации. Снижение влияния дефекта на работоспособность системы указывает на то, что все случаи, которые обнаруживаются выше определенного порога, далее

обрабатываются системой, в то время как случаи, которые находятся ниже этого порога, рассматриваются как ложные тревоги. Точное значение порога тщательно определяется на основе анализа данных, проведенного по всему объему выходных данных типовых моделей.

Классифицированные модели позднее применяются для каждого обнаружения, полученного из общих моделей. С точки зрения науки о данных, эти модели характеризуются чрезвычайно высокой точностью. Это означает, что для каждого обнаружения, сделанного с помощью классификационной модели, мы можем быть уверены, что такое поведение имело место. В то же время, эти модели не гарантируют, что они смогут уловить все случаи такого поведения. Выходные данные классифицированных моделей в данном случае являются подмножеством выходных данных общей модели, так как они отфильтровывают только те случаи, которые признаны релевантными. Этот двухступенчатый конвейер, состоящий из общей и классифицированной моделей, увеличивает вероятность того, что реальная проблема будет обнаружена и надлежащим образом классифицирована.

Общие модели

Пять общих моделей были выявлены как наиболее значимые в ходе обширного анализа телеметрии в облачной среде баз данных Azure SQL за все время. Все модели сравнивают определенные значения, собранные в текущем интервальном периоде, со значениями, собранными в базовом периоде (период, в котором база данных гарантированно не испытывала никаких проблем).

В модели Query duration запрос считается перегруженным при выполнении следующих условий. Во-первых, для данного запроса суммируется среднее истекшее время, дисперсия истекшего времени и количество исполнений для базового и текущего периода. Затем по

заданному набору значений вычисляется Welch T-test [1.29]. Если результат выполненного теста, называемый t-значением, меньше 0,01, то запрос считается неоптимальным. Для уменьшения количества ложных срабатываний была выбрана альфа-версия 0.01. Серьезность неоптимального выполнения запроса вычисляется как разница между истекшим временем в текущем интервале и базовым периодом, умноженным на количество выполненных запросов.

Модель Timeouts Query Level учитывает только запросы, тайм-аут которых получен на этапе выполнения запроса в текущем интервале. Коэффициент суммарного таймаута рассчитывается как общее количество запросов, поделенное на общее количество выполненных запросов. Если коэффициент в текущем интервале превышает 3 базовых стандартных отклонения от базового среднего значения, то он рассматривается как регрессия.

Так как есть запросы, обрабатываемые SQL-сервером, которые так и не дошли до фазы выполнения, и таким образом не попали в хранилище запросов, то модель уровня базы данных Timeouts Database Level предназначена для обнаружения аномалий таймаута на уровне базы данных. Основным критерием перегруженности является увеличение количества прерванных запросов (что является специфическим сигналом, который SQL сервер отображает в журнале) на уровне всей базы данных. Если коэффициент суммарного таймаута на уровне БД в текущем интервале более 3-х базовых стандартных отклонений от базового среднего значения, то этот интервал рассматривается как неоптимальный и модель добавляет его в очередь на дальнейшую обработку.

Модель исключений отслеживает увеличение числа исключений. Модель концептуально похожа на модель Timeout Database Level. Единственное отличие связано с отслеживаемым соотношением. В этой модели отслеживаемое соотношение вычисляется как количество

исключений на общее количество запросов.

Наконец, модель Compile Time похожа на модель Query Duration, но она отслеживает другой сигнал. В этой модели отслеживается длительность только одного этапа (компиляции) вместо общей длительности процесса выполнения запроса. Как только обнаружена перегрузка во время компиляции, такой интервал посылается на дальнейший анализ по категориальным моделям, чтобы понять, что именно продлило фазу компиляции.

Классифицированные модели

После обширного анализа различных случаев, выявленных общими моделями, они группируются по одиннадцати классифицированным моделям. Проведенный анализ включал в себя различные типы рабочих нагрузок из совершенно разных баз данных и клиентов. Группировка выполнялась на основе окончательного анализа первопричин, определенного командой инженеров и подтвержденного заказчиком.

Первая модель классификации под названием Hitting Resource Limits характеризуется сильной обнаруженной корреляцией между перегруженным запросом и лимитами ресурсов, которые достигаются на процессоре. Тип ожидания в SQL-сервере соответствует конкретному типу ресурса, занимаемой задачи, выполняющейся на SQL-сервере.

Модель Plan Regression Category отслеживает изменение плана выполнения запроса, которое может быть соотнесено с перегрузкой, сообщаемой общими моделями. Если из типовых моделей наблюдается запрос, план которого изменяется в текущем интервале по сравнению с базовым периодом, включая увеличенное время исполнения, то это рассматривается как нарушение плана.

Модель Failover Category является одной из самых простых. Если отказ базы данных произошел при переходе с одной машины облачного

сервиса на другую, из-за пустых кэшей и загрузки на новую машину, то эта база данных может быть подвергнута вмешательству на короткий период времени. Это принято считать отклонением всей системы.

Модель Workload Increase/Pileup отслеживает эмпирический отпечаток рабочей нагрузки. Если существует корреляция между перегрузкой, наблюдаемой общими моделями, и изменением отпечатка рабочей нагрузки, то увеличение количества запросов рассматривается как увеличение рабочей нагрузки, в то время как увеличение количества таймаутов рассматривается как стек рабочей нагрузки.

Модель Hitting Worker Limits аналогична модели Hitting Resource Limits за исключением того, что также отслеживается количество активных пользователей. В SQL Server сотрудники отвечают за обработку запросов, поступающих от клиента. Если наблюдаемая перегрузка может быть соотнесена с количеством активных работников, которые начали достигать лимитов, установленных SQL Server, то интервал помечается как Hitting work limits.

Модель Memory Pressure/Pileup похожа на модель Workload increase/Pileup. Модель обнаруживает интенсивное использование общей памяти и определяет, что запросы ожидают выделения новых ресурсов памяти (тип ожидания resource_semaphore). Если общее время ожидание увеличивается и может быть соотнесено с перегрузками, обнаруженными общими моделями, то этот интервал помечается либо как Memory Pressure, либо как Memory Pileup.

Модель Locking анализирует типы ожидания заблокированных общих ресурсов. Она обнаруживает случаи, когда есть несколько запросов, которые выполняются одновременно над одним и тем же ресурсом. Это может произойти как умышленно, когда клиент явно блокирует ресурс в БД, так и неявно, когда есть два параллельных запроса, изменяющих равный набор или подмножество записей в БД.

Increase Maximum Degree of Parallelism соответствует случаю, когда существует корреляция между неоптимальными запросами, найденными по обобщенным моделям и типом ожидания CXPACKET. Данный тип ожидания отслеживает накладное время, необходимое для синхронизации различных потоков, выполняющих параллельное выполнение запроса.

Модель Memory Contention характеризует рабочую нагрузку с запросами, которые обращаются к одному и тому же ресурсу БД на уровне страницы памяти. Если это происходит в базе данных temp, которая является системной базой данных общего назначения в SQL Server, используемой клиентом для хранения временных результатов, то она рассматривается как Temp DB contention.

Missing Index Category появляется, когда запрос, обнаруженный по обобщенным моделям, также обнаруживается в фазе категоризации, где его выполнение влечет за собой доступ к тому же неиндексированному ресурсу БД. В таких случаях результаты считываются из таблицы на диске, что является ресурсозатратной операцией и приводит к дополнительным накладным расходам на операции ввода-вывода.

Категория "Slow Client" подходит для случаев, когда соответствующий тип ожидания увеличивается и может быть соотнесен с аномалией, обнаруженной ранее по общим моделям. Аномалия в этом случае является признаком проблемы с пропускной способностью сети между клиентским приложением и базой данных в облачном сервисе.

В целом, общий набор статистических моделей состоит из пяти общих и одиннадцати классифицированных моделей, которые были построены с использованием реальной рабочей нагрузки в производственной среде Azure SQL. Для различных случаев, отобранных общими моделями и далее обработанных классифицированными, может случиться так, что результаты этих двух фаз будут выглядеть в целом похожими, в то время как конечный анализ первопричины и взаимосвязь

между причиной и следствием различны. Это может произойти, когда точно один и тот же набор моделей обнаружит две или более ситуации, которые отличаются только степенью серьезности, сообщаемой классифицированной моделью. Степень влияния дефекта на работоспособность модели соответствует уровню корреляции между выходными данными общей и классифицированной моделей.

Для автоматического поиска и устранения неисправностей в БД очень важно различать такие случаи, чтобы точно определить первопричину проблемы. После детального анализа был сделан вывод о том, что выходные данные научных моделей должны быть далее обработаны экспертной системой, которая делает возможным точное разграничение между конкретными случаями, включая их различную степень серьезности.

1.4.6. Экспертная система

Экспертная система, основанная на правилах, предназначена для разграничения различных выходов, полученных из общей и классифицированной моделей. Тщательный анализ специфики экспертной системы был проведен до начала разработки системы [1.30].

В системах такого типа каждое правило состоит из предшествующей и последующей частей, которые традиционно представлены в виде набора утверждений IF-THEN. В этом случае в качестве фактов рассматриваются категории, выявляемые конкретным набором моделей. Последующая часть состоит из результата, который должен быть возвращен, если он соответствует правилу. Правила представлены в формате XML со строгим определением схемы, как показано на рис. 1.2.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ExpertSystemRules" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="">
  <!-- Type definitions -->
  <xs:complexType name="ClassificationResultType">
    <xs:all>
      <xs:element type="xs:string" name="Class" minOccurs="1" maxOccurs="1"/>
      <xs:element type="xs:string" name="Description" minOccurs="1" maxOccurs="1"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="RuleType">
    <xs:all>
      <xs:element type="xs:string" name="Fact" minOccurs="1" maxOccurs="1"/>
      <xs:element type="xs:string" name="Value" minOccurs="1" maxOccurs="1"/>
      <xs:element type="xs:string" name="Operator" minOccurs="1" maxOccurs="1"/>
      <xs:element type="xs:string" name="Type" minOccurs="1" maxOccurs="1"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="OrType">
    <xs:choice minOccurs="2">
      <xs:element name="And" type="AndType"/>
      <xs:element name="Rule" type="RuleType"/>
      <xs:element name="Not" type="NotType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="AndType">
    <xs:choice minOccurs="2">
      <xs:element name="Or" type="OrType"/>
      <xs:element name="Rule" type="RuleType"/>
      <xs:element name="Not" type="NotType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="NotType">
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element name="And" type="AndType"/>
      <xs:element name="Or" type="OrType"/>
      <xs:element name="Rule" type="RuleType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="ExpressionType">
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element name="Not" type="NotType"/>
      <xs:element name="And" type="AndType"/>
      <xs:element name="Or" type="OrType"/>
      <xs:element name="Rule" type="RuleType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="ClassificationRuleSetType">
    <xs:sequence>
      <xs:element name="Expression" type="ExpressionType" minOccurs="1" maxOccurs="1" />
      <xs:element name="Result" type="ClassificationResultType" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <!-- ExpertSystemRules definition -->
  <xs:element name="ExpertSystemRules">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ClassificationRuleSet" type="ClassificationRuleSetType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Рис. 1.2. XML-схема экспертной системы на основе правил

Правила экспертной системы в этой схеме могут содержать один или несколько наборов правил, в каждом из которых есть выражение и результат. Выражение представляет собой либо одно правило, либо комбинацию правил, которые комбинируются с логическими операторами AND, OR и NOT.

В такой экспертной системе может случиться, что некоторые правила накладываются друг на друга, что означает, что одно правило является подмножеством другого. Система построена таким образом, что результаты агрегируются в множество и, если нет перекрывающихся правил, они возвращаются в виде отдельных результатов. Из рис. 1.3.б и 1.3.в видно, что набор различных фактов может быть сопоставлен с разными результатами по правилу, и что за конечный результат всегда принимается множество, если подмножество и множество перекрываются. С другой стороны, на рис. 1.3.г и 1.3.д представлены различные крайние случаи, которые недопустимы в этой системе. Если имеется несколько наборов результатов правил, которые покрывают факты из других наборов результатов правил, образуя тем самым цепочку перекрывающихся результатов, то возникает предупреждение для инженерной группы. В случае, если ни одно из существующих правил не совпадает с выходом из общей и категориальной моделей, такой случай помечается как "Неизвестно" и возникает другой тип предупреждения. Оповещение предупреждает о том, что подобная ситуация должна быть дополнительно исследована и разрешена путем определения новых или переопределения существующих правил в системе.

1.4.7. Инфраструктура

Ранее определенная система требует соответствующей инфраструктуры. Полная телеметрия хранится в структурированной базе данных in-memory, предоставляемой внутренней командой Microsoft,

которая работает внутри Microsoft Azure. Этот источник телеметрии анализируется облачными службами (бегунами), в которых каждый «бегун» ассоциируется с регионом, в котором расположены базы данных Azure SQL.

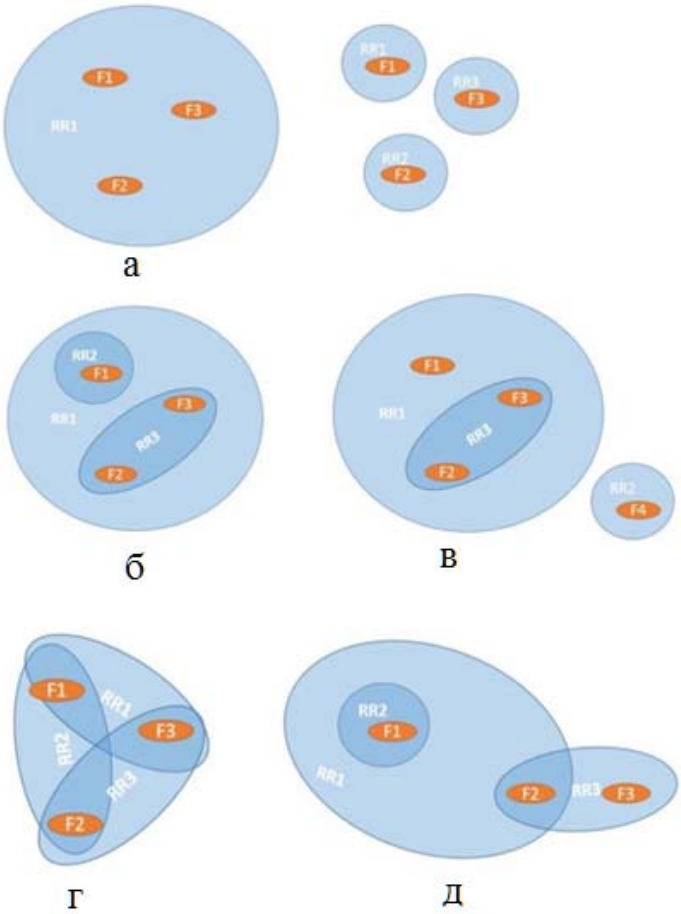

Рис. 1.3. Объединение выходов универсальных и категориальных моделей

Эти «бегуны» параллельно выполняют определенные аналитические модели и хранят выходную информацию во внутренней реляционной базе данных. В зависимости от нагрузки и количества регионов, эти «бегуны» могут легко масштабироваться. Разбиение может выполняться по группам регионов, по одному региону или по подмножеству баз данных в одном регионе. Внутренняя база данных географически реплицирована для повышения доступности и надежности. Это является важным требованием,

так как все другие базы данных SQL компании Azure, которые могут испытывать проблемы с производительностью, зависят от этой базы данных. Наконец, конечные результаты выдаются в виде журнала диагностики [3.31] и могут быть получены конечным пользователем по различным каналам связи в соответствии с его выбором. Высокоуровневая структура этой системы показана на рис. 1.4.

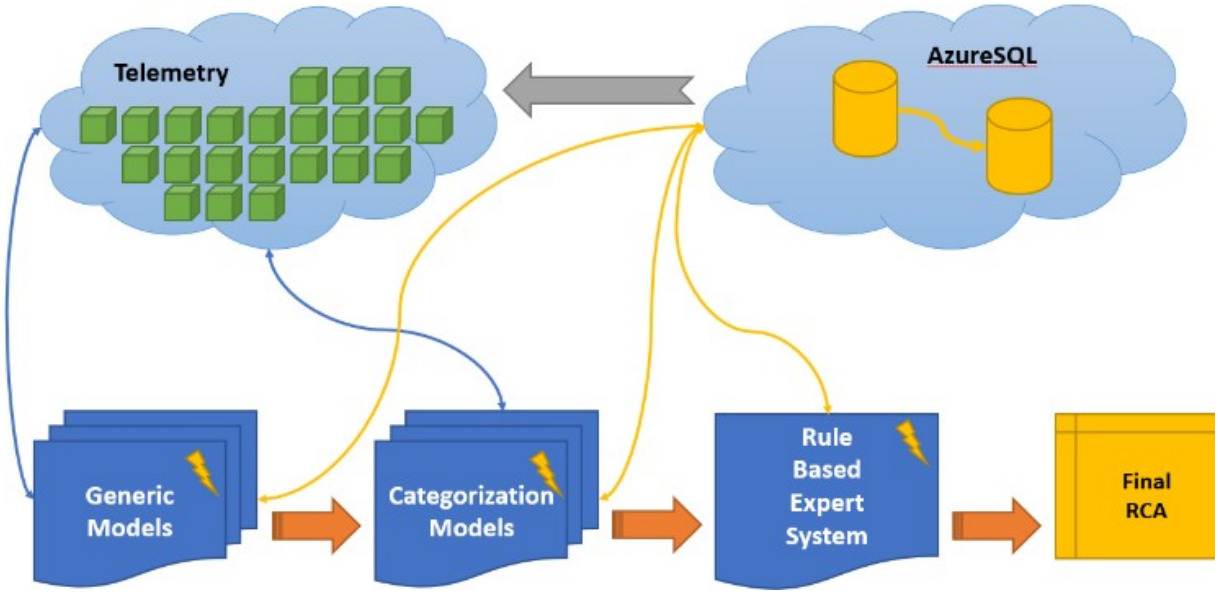

Рис. 1.4. Системная инфраструктура высокого уровня

1.5. Постановка задач работы

Проведенный анализ проблем управления транзакциями в гетерогенных объектах распределенной сети показал, что необходимо решить следующие задачи:

1. Разработать механизм распределения данных о специальных транзакциях с оперативным контентом, обеспечивающий минимизацию времени передачи данных.

2. Разработать алгоритм локального кэширования хронологически запрошенных данных транзакций, обеспечивающий уменьшение время

задержки передачи данных о специальных транзакциях

3. Создать алгоритм разделения и перенаправления запросов между кдиентами, межобъектными интерфейсами или облаком, дополнительно учитывающий интеграцией информации о состоянии кэширования, предполагаемом размере данных и пропускной способности.

4. Модифицировать протокол планирования транзакций в СУБД реального времени для учета не только временных ограничений транзакций, но и критерии, установленные пользователями базы данных.

5. Предложить архитектуру СУБД реального времени с использованием модифицированного протокола планирования транзакций для улучшения качества предоставления услуг пользователям при реализации управления с обратной связью.

6. Создать программный комплекс для проверки динамических связей технологических схем и баз данных на основе системы кодирования Kraftwerk Kennzeichen System.

7. Создать компоненты распределенной специальной информационно-вычислительной системы управления с унифицированными механизмами межмодульного взаимодействия для оптимизации состава резервного оборудования и исключения программных конфликтов при обмене данными.

Список источников к главе 1

1.1. Xiaolin L. Real time regression analysis in internet of stock market cycles// Cognit Syst Res. 2018; 52: 371-379.

1.2. Zhao Z., Min G., Gao W., Wu Y., Duan H., Ni Q. Deploying МОИН computing nodes for large-scale IoT: a diversity aware approach// IEEE Internet Things J. 2018; 5(5): 3606-3614.

1.3. Brzeszczynski J., Ibrahim B.M. A stock market trading system based on foreign and domestic information// Expert Syst Appl. 2019; 118: 381-399.

1.4. Mach P., Becvar Z. Cloud-aware power control for real-time application offloading in mobile МОИН computing// Trans Emerg Telecommun Technol. 2016; 27(5): 648-661.

1.5. de Assuncao M.D., da Silva V.A., Buyya R. Distributed data stream processing and МОИН computing: a survey on resource elasticity and future directions// J Netw Comput Appl. 2018; 103: 1-17.

1.6. 2.6. Ao W.C., Psounis K. Fast content delivery via distributed caching and small cell cooperation// IEEE Trans Mobile Comput. 2018; 17(5): 1048-1061.

1.7. Dar S., Franklin M.J., Jonsson B.T., Srivastava D., Tan M. Semantic Data Caching and Replacement// Proc. of the 22 VLDB Conf., Mumbai-Bombay, India, 1996, p. 330-341.

1.8. Aranha R.F.M., Ganti V., Narayanan S., Muthukrishnan C.R., Prasad S.T.S., Ramamritham K. Implementation of a real-time database system// Inf. Syst. 1996, 21, 55–74.

1.9. Amirijoo M., Hansson J., Son S.H. Specification and management of QoS in real-time databases supporting imprecise computations// IEEE Trans. Comput. 2006, 55, 304–319.

1.10. Lu C., Stankovic J.A., Son S.H., Tao G. Feedback control real-time scheduling: framework, modeling, and algorithms// Real-Time Syst. 2002, 23,

85–126.

1.11. Katet M., Bouazizi E., Sadeg B. Prise en Compte des Donnrees Drerivrees Temps Rreel dans Une Architecture de Contrˆole par Rretroaction// Revue rElectronique des Technologies de lInformation (e-TI), 2007, vol. 11.

1.12. Zeddini B., Duvallet C., Sadeg B. Une Approche Qualitre de Service dans Les Syst`emes Multimredias Distribures// Revue rElectronique des Technologies de l'Information (e-TI), 2007, vol. 4.

1.13. Hamdi S., Bouazizi E., Faiz S. QoS management in real-time spatial big data using feedback control scheduling// ISPRS Ann. Photogrammetry Remote Sens. Spat. Inf. Sci. 2015, II–3/W5, 243–248.

1.14. D. V. Aken, A. Pavlo, G. J. Gordon and B. Zhang, "Automatic Database Management System Tuning Through Large-scale Machine Learning," / 2017 ACM International Conference on Management of Data, Chicago, Illinois, USA, 2017.

1.15. P. H. Callahan, "Expert systems for AT&T switched network maintenance," / AT&T Technical Journal, vol. 67, no. 1, pp. 93-103, 1988.

1.16. V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellamanickam, S. Keerthi, S. Herbert and S. Dhulipalla, "Learning a Hierarchical Monitoring System for Detecting and Diagnosing Service Issues," / 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 2015.

1.17. SQL Sentry, "Sql Sentry preformance advisor," SQL Sentry, 2016. Режим доступа: http://www.sqlsentry.com/products/performance-advisor/sql-server-performance.

1.18. DELL software / "Spotlight on SQL Server Enterprise Edition," 2016.

1.19. Idera, "Idera SQL diagnostic manager," Idera, 2016. Режим доступа: https://www.idera.com/productssolutions/sqlserver/sqldiagnosticmanager/resources.

1.20. "Overview of Oracle Enterprise Manager Cloud Control 13c," Oracle, 2016. Режим доступа: https://docs.oracle.com/cd/E63000_01/EMCON/overview.htm#EM CON109.

1.21. "Automatic Performance Diagnostics," Oracle, 2017. Режим доступа: https://docs.oracle.com/database/121/TGDBA/pfgrf_diag.htm#TG DBA026.

1.22. "Automatic SQL tuning," Oracle, 2018. Режим доступа: https://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm #CHDJDFGE.

1.23. S. Steve, "SQL Database Advisor," Microsoft, 2016. Режим доступа: https://azure.microsoft.com/en-us/documentation/articles/sql-database-index-advisor/.

1.24. D. Dundjerski, S. Lazic, M. Tomasevic and D. Bojic, "Improving schema issue advisor in the Azure SQL database," / 25th Telecommunications Forum TELFOR, Belgrade, Serbia, 2017.

1.25. Microsoft, "Azure SQL Database: The world's first intelligent cloud database service," Microsoft, 22 August 2017. Режим доступа: https://techcommunity.microsoft.com/t5/Microsoft-Ignite-Content-2017/Azure-SQL-Database-The-world-s-first-intelligent-cloud- database/td-p/98549.

1.26. O.Brent, "sp_AskBrent," 2016. Режим доступа: https://www.brentozar.com/askbrent/.

1.27. V. Jeyakumar, O. Madani, A. Parandehgheibi, A. Kulshreshtha, W.Zeng, N. Yadav, "ExplainIt! -- A declarative root-cause analysis engine for time series data," / SIGMOD'19 2019 International Conference on Management of Data, Amsterdam, Holland, 2019.

1.28. T. Raeder, B. Dalessandro, F. Provost, "Design principles of massive, robust prediction systems" / 18th ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China, 2012.

1.29. B. L. Welch, “The Generalization of ‘Students’ problem when

several different population varlances are involved’, / Biometrika, vol. 34, Issue 1-2, pp. 28–35, 1947.

1.30. M. S. Hossain, S. Rahaman, A.-L. Kor, K. Andersson and C. Pattinson, "A Belief Rule Based Expert System for Datacenter PUE Prediction under Uncertainty," / IEEE Transactions on sustainable computing, vol. 2, no. 2, pp. 140-151, 2017.

1.31. J. Kemnetz and C. Watson, "Overview of Azure Diagnostic Logs," Microsoft, 21 August 2017. Режим доступа: https://docs.microsoft.com/en-us/azure/monitoring-and- diagnostics/monitoring-overview-of-diagnostic-logs.

2. Управление транзакциями в гетерогенных объектах распределенной сети

2.1. Механизм распределения данных о специальных транзакциях с оперативным контентом в реальном времени на основе кэширования в гетерогенных объектах распределенной сети

Предлагается механизм распределения данных о специальных транзакциях в реальном времени на основе распределенного кэша. Распределенный кэш в основном состоит из двух частей: одна - кэш межобъектного интерфейса (МОИН), а другая - кэш небольшой емкости конечного клиента.

1. Облако анализирует важность различных акций в соответствии с историческими запросами “от-до” соответствующей платформы МОИН. Основываясь на вычислении возрастающей важности и динамическом состоянии сети, облако отправляет более важные данные о транзакциях на соответствующий межобъектный интерфейс, когда сеть имеет достаточную доступную пропускную способность.

2. Межобъектный интерфейс сообщает информацию о своих кэшированных данных другим МОИН, сохраняя при этом информацию о кэшированном содержимом локальных клиентов. Получив запрос от клиента, межобъектный интерфейс выбирает оптимальную стратегию передачи в соответствии со статусом сети и информацией о распределенном кэше. Он может перенаправить запрос в локальный кэш, другие терминалы, другие МОИН или удаленное облако. Цель состоит в том, чтобы достичь наименьшего времени завершения передачи.

3. Клиент кэширует соответствующие хронологически запрошенные данные транзакции в зависимости от своей емкости в соответствии с политикой вытеснения давно неиспользуемых данных (LRU -

Least Recently Used [2.7]). Для того чтобы оценить эффективность схемы, произведено сравнение время выполнения запросов на данные о специальных транзакциях. Результаты показывают, что предложенные схема значительно улучшила время завершения передачи данных, что хорошо решает требования к передаче данных о специальных транзакциях в режиме реального времени.

2.1.1. Предложенная архитектура

Предпосылкой работы с большими БД является получение данных в режиме реального времени. Поэтому управления данными постоянно стремится к более быстрой скорости обновления данных. Поэтому вопрос о том, как повысить эффективность передачи и обеспечить передачу в режиме реального времени, стал серьезной проблемой в этой области. В соответствии с характеристиками данных о специальных задачах, общие данные по управлению сложными системами в реальном времени могут быть наложены на предыдущие исторические данные за короткий период времени T для поддержки точного анализа. Основываясь на этой функции, в статье предлагается механизм передачи данных о специальных транзакциях в реальном времени, основанный на распределенном внутрисетевом кэшировании.

Как показано на рис. 1.1, облачная платформа отвечает за хранение всей информации о транзакциях на локальном клиенте, а межобъектный интерфейс отвечает за ускорение передачи данных о транзакциях и поддержание информации распределенного кэша.

Пользовательские терминалы запрашивают данные о транзакциях, кэшируя часть данных о транзакциях.

Облачная платформа отвечает за хранение информации о специальных транзакциях, генерируемой всеми клиентами, включая информацию в режиме реального времени и историческую информацию.

Облачная платформа разделяет данные о транзакциях на независимые блоки данных в соответствии с двумя измерениями идентификатора контента и периода времени, и каждый блок данных представлен как <идентификатор контента, время>. Установка периода времени должна определяться в соответствии с частотой специальных транзакций и задержкой передачи по сети. Устанавливаем период времени равным 10 с. Например, <00000001, 3 марта 2021 10:12:20> означает, что срок действия данных о транзакциях по контенту “00000001” истекает в 10:12:20 3 марта 2021 года. Каждый клиент транзакции будет запрашивать информацию о транзакциях для разных контентов в разное время. Облачная платформа должна анализировать различные запросы пользователей различных межобъектных интерфейсов и заранее передавать соответствующую информацию об контентах на МОИН.

Платформа МОИН отвечает за кэширование соответствующих важных исторических данных о специальных транзакциях. Однако объем всех исторических данных транзакций настолько велик, что межобъектный интерфейс не имеет возможности кэшировать их все. Поэтому в предлагаемом решении клиент транзакций также кэширует часть запрошенных данных транзакций в соответствии со своей емкостью хранилища и отправляет соответствующую информацию о кэшированных данных на свой межобъектный интерфейс. В то же время межобъектный интерфейс (МОИН) обменивается таблицами данных кэша с другими МОИН для поддержки совместного кэширования и реагирования. Кроме того, межобъектный интерфейс отвечает за измерение состояния сети в режиме реального времени для других МОИН, таких как пропускная способность сети и задержка передачи. Состояние сети между клиентами транзакций также будет отправлено на соответствующий межобъектный интерфейс. Эта информация о состоянии сети важна для интеллектуального принятия решений о запросах.

Как показано на рис. 2.1, после поступления запроса пользователя межобъектный интерфейс разумно разделяет соответствующий запрос контента на основе таблицы сведений о локальном распределенном кэше и соответствующего состояния сети и перенаправляет разделенные подзапросы на другие МОИН, локальные узлы транзакций или облако. Цель принятия решения состоит в том, чтобы свести к минимуму время, необходимое клиенту транзакции для получения полных данных. При реализации, чтобы избежать затрат времени во время создания сеанса передачи, МОИН поддерживает пул соединений с облаком, который используется для немедленной передачи данных о специальных транзакциях в режиме реального времени в любое время.

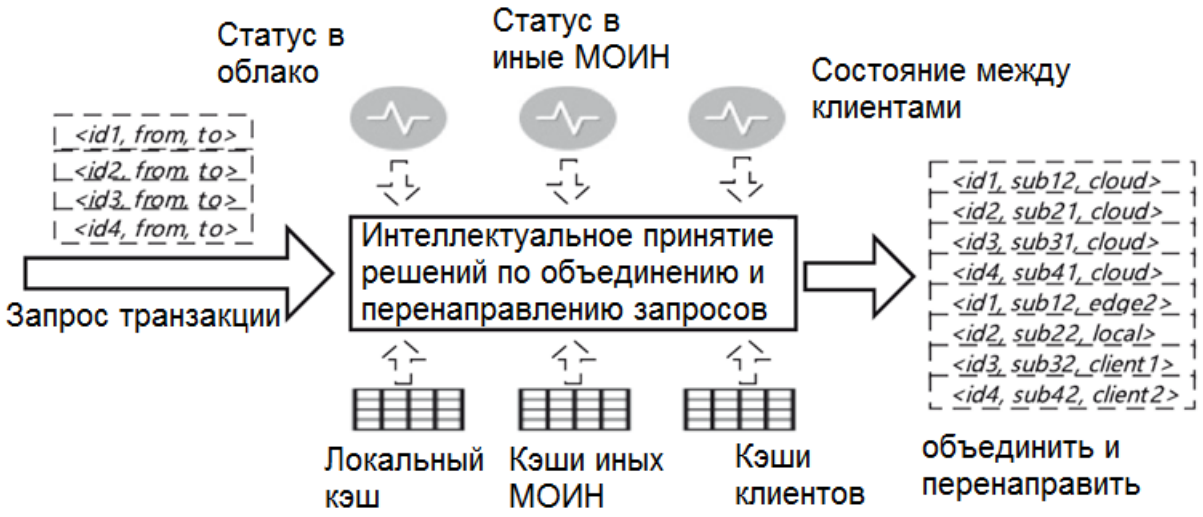

Рис. 2.1. Интеллектуальное решение на основе межобъектных интерфейсов: объединение запросов и перенаправление

Клиенты транзакций динамически запрашивают данные о транзакциях с контентами. Запросы от разных клиентов имеют соответствующую локальность, что в основном проявляется в их очевидном предпочтении знания о конкретных видах контентов. Каждый клиент может кэшировать определенный объем контента, а емкость кэша намного меньше, чем у межобъектного интерфейса. Но общее количество клиентов значительно превышает количество межобъектных интерфейсов.

Данные, запрошенные клиентом, кэшируются локально до тех пор, пока кэшированное содержимое не превысит емкость локального хранилища, и заменяются в соответствии с политикой LRU. Кроме того, как только запрос клиента будет перенаправлен другим клиентам, статус передачи между ними будет отправлен на локальную пограничную платформу в качестве основы для интеллектуального решения.

Таким образом, создан механизм распределения данных о специальных транзакциях с оперативным контентом, отличающийся распределенным кэшированием в гетерогенных межобъектных интерфейсах в реальном масштабе времени, обеспечивающий минимизацию времени передачи данных.

2.1.2. Субоптимальное решение

Передача данных на основе важности из облака в межобъектный интерфейс

Удаленное облако обычно имеет наибольшую задержку передачи на периферию или клиентам. Чтобы избежать передачи большого объема данных о транзакциях при поступлении запроса информации о транзакциях с акциями, облако заранее отправляет исторические данные о транзакциях с контентами на соответствующий межобъектный интерфейс. Таким образом, когда запрос действительно выполняется, передача данных в облако может быть значительно сокращена, что обеспечивает более быстрый ответ на запрос. Однако как ресурсы пропускной способности, так и емкость кэша межобъектных интерфейсов ограничены, поэтому очень важно, как передать соответствующие данные о специальных транзакциях на соответствующий межобъектный интерфейс. Как правило, более важная информация о транзакциях должна передаваться на соответствующий межобъектный интерфейс, чтобы максимизировать эффект, обеспечиваемый возможностями пограничного кэширования.

В статистике важности учитываются только идентификаторы контентов и не проводится различий между различными временными срезами. Предполагая, что d представляет идентификатор контента, контент запрашивается в момент времени t, затем запускается обновление его важности.

Предположим, что контент был запрошен в общей сложности k раз, и время запроса равно t_1, t_2,…, t_k, где $t_k = t$. После запроса в момент времени t его соответствующая важность p(d, k) вычисляется следующим образом:

$$\begin{aligned} p(d,k) &= \sum_{i=1}^{k} e^{-\lambda(t-t_i)} = e^{-\lambda(t-t_k)} + \sum_{i=1}^{k-1} e^{-\lambda(t-t_{k-1})} e^{-\lambda(t_{k-1}-t_i)} = \\ &= e^{-\lambda(t-t_k)} + e^{-\lambda(t-t_{k-1})} \sum_{i=1}^{k-1} e^{-\lambda(t_{k-1}-t_i)} = e^{-\lambda(t-t_k)} + e^{-\lambda(t-t_{k-1})} p(d,k-1) \end{aligned} \tag{2.1}$$

Как показывает (2.1), каждый запрос вносит свой вклад в важность, но вклад запроса в важность со временем экспоненциально уменьшается. Преимущество этого расчета важности заключается в том, что каждый расчет необходимо только постепенно обновлять по предыдущему результату, что значительно снижает стоимость расчета важности.

Однако после запроса данных о контентах создается высокая важность. Но по прошествии длительного периода времени он больше не запрашивается, тогда важность контента не может быть обновлена, что приводит к большому отклонению от фактической важности. Это может снизить эффективность кэширования, поскольку некоторые неважные данные будут смещены к краям. Поэтому в предлагаемой схеме будем периодически обновлять значения важности всех контентов экспоненциальным образом на ΔT. Таким образом, значение важности контента, который не запрашивался в течение длительного времени, будет постоянно снижаться, чтобы избежать неправильной оценки важности.

В реальной сетевой среде важность рассчитывается для каждого

межобъектного интерфейса, а запросы также классифицируются в соответствии с соответствующим МОИН. Как только облако обнаружит, что сеть имеет достаточную доступную пропускную способность, а межобъектный интерфейс располагает достаточным пространством кэша, облако выберет наиболее важную информацию о специальных транзакциях из списка важности, соответствующего граничным данным, и передаст ее на соответствующий межобъектный интерфейс.

Таким образом, представлен алгоритм локального кэширования хронологически запрошенных данных транзакций, отличающийся учетом собственной емкости МОИН и применением политикой вытеснения давно неиспользуемых данных и обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях.

Интеллектуальное решение для разделения и перенаправления запросов

Пусть $R=\{<d_1, S_1>, <d_2, S_2>, \ldots, <d_k, S_k>\}$ - запрос, где d_i есть идентификатор контента ID и S_i – время запроса.

Пусть $L=\{<Ld_i, LS_i>\}$ - локальный набор данных кэширования специальных транзакций. Пусть $E_j=\{<Ed_{ji}, ES_{ji}>\}$ - набор данных кэширования специальных транзакций на j-м МОИН. Пусть $C_m=\{<Cd_{mi}, CS_{mi}>\}$ - набор данных кэширования о специальных транзакциях у m-го клиента.

Пусть B(x, y) - пропускная способность между узлами x и y, где x и y могут быть клиентами, МОИН или облаком.

Для каждого $<d_i, S_i> \in R$, выберем клиента, МОИН или облако, которое может минимизировать задержку передачи данных транзакции, на основе информации о состоянии кэширования, предполагаемом размере данных и пропускной способности B(x, y). Решение о разделении и

перенаправлении запросов принимается с «жадностью», чтобы свести к минимуму общее время передачи.

Например, пусть есть клиент C и МОИН E, у которых есть данные о транзакциях по контенту $<d_i, S_i>$. Полосы пропускания от них до запрашивающего соответственно B_c и B_e. Предположим, что размер данных $<d_i, S_i>$ равен $Size(<d_i, S-i>)$. В настоящее время для C и E были назначены задачи передачи данных с размерами size(C) и size(E). Оценим время передачи C и E в соответствии с формулами $\frac{size(C)+Size(<d_i,S-i>)}{B_C}$ и $\frac{size(E)+Size(<d_i,S-i>)}{B_E}$. Для передачи $<d_i, S_i>$ будет выбран вариант с меньшим временем передачи.

Таким образом, описан «жадный» алгоритм разделения и перенаправления запросов между кдиентами, межобъектными интерфейсами или облаком, отличающийся интеграцией информации о состоянии кэширования, предполагаемом размере данных и пропускной способности и обеспечивающий минимизацию задержки передачи данных транзакции.

Численный анализ

Пусть общее количество контентов составляет 1000, размер данных каждого блока контентов (1 c) составляет 160 бит, запрос по умолчанию для данных каждого контентов составляет 7 дней, а соответствующий размер данных за 7 дней составляет около 1,2 Мбайт. Количество МОИН составляет 10, и под каждым МОИН находится 1000 клиентов транзакций. Пропускная способность между МОИН составляет 20 Мбит/с, пропускная способность передачи между облачными и платформами МОИН составляет 5 Мбит/с, а пропускная способность передачи между клиентами одного и того же МОИН составляет 50 Мбит/с. Каждый клиент генерирует

запрос на информацию о контентах в соответствии с законом Ципфа [8] с параметром распределения 1,07. МОИН задается емкостью 10 Мбайт, 20 Мбайт, 40 Мбайт и 80 Мбайт.

С помощью экспериментов сравниваются эффекты различных настроек МОИН - внедрения кэша МОИН и отказа от внедрения кэша клиента.

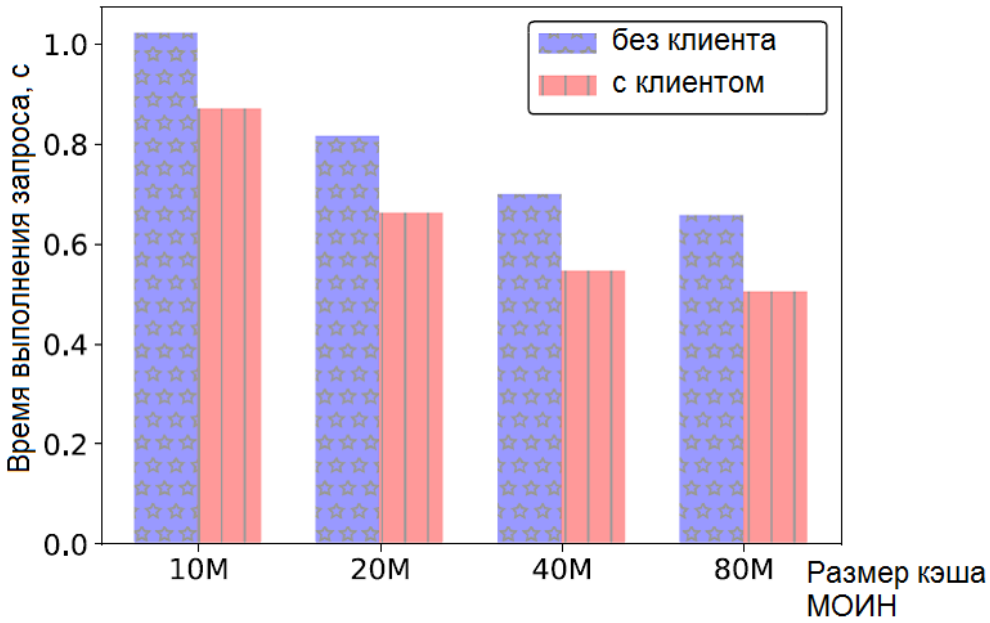

Рис. 2.2. Среднее общее время выполнения для каждого клиента

Если нет кэширования МОИН и кэширования клиента, данные о транзакциях с контентом будут отправлены непосредственно из удаленного облака. В условиях эксперимента размер данных составляет 1,2 Мбайт, а пропускная способность - 5 Мбит/с. Таким образом, время завершения передачи составляет 1,92 с. Если ввести, кэширование в МОИН, производительность значительно улучшится. Как показано на рис. 3, результаты с кэшированием в МОИН 10, 20, 40 и 80 Мбайт составляют соответственно 1,0025, 0,8015, 0,6740 и 0,6410 с. Если дополнительно ввести кэширование у клиента, результаты составят соответственно 0,8474, 0,6464, 0,5189 и 0,4859 с. Чтобы еще больше продемонстрировать стабильность нашей схемы, сгенерируем 50000 запросов и вычислим

среднее время завершения для каждой 1000 запросов. Как показано на рис. 4, результаты соответствуют общему времени завершения на рис. 2.2.

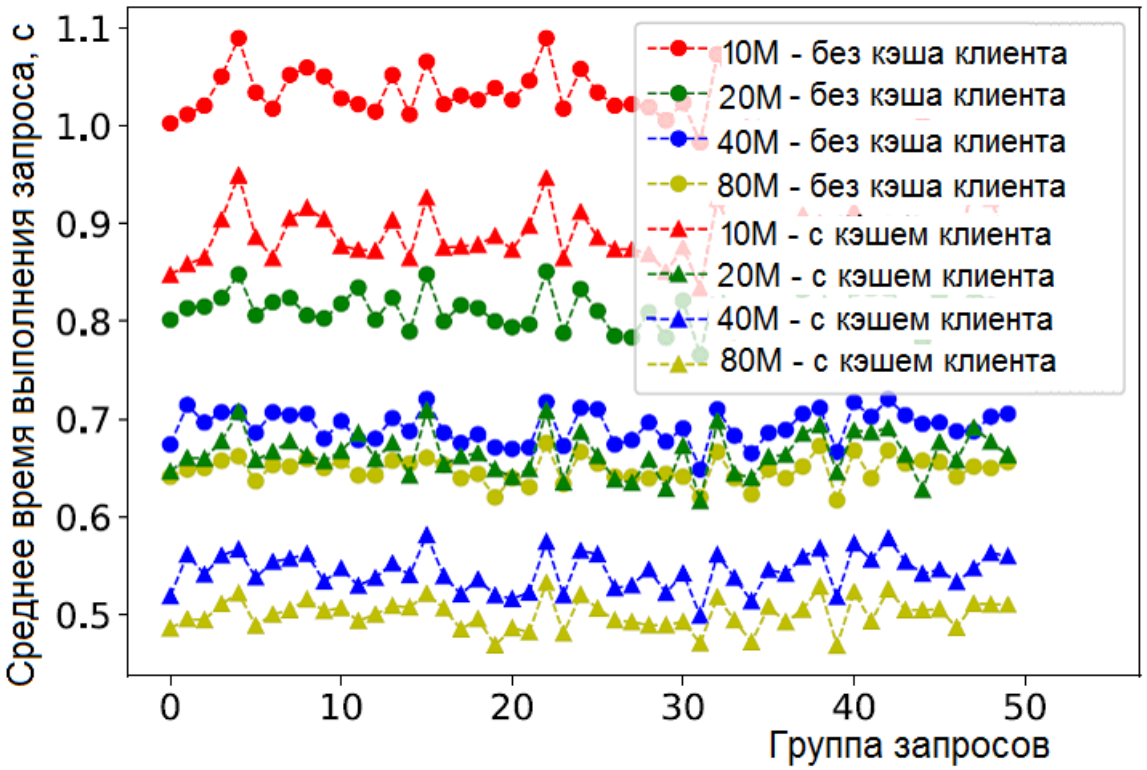

Рис. 2.3. Среднее общее время выполнения для каждой 1000 запросов

2.2. Экспериментальное исследование системы автоматического поиска и устранения неисправностей в базе данных

Современные приложения ориентированы на облачные сервисы для достижения лучшей производительности, географической репликации и снижения стоимости владения. Следуя современным концепциям облачных сервисов, данное исследование использует богатые телеметрические данные и отображает рабочую нагрузку, выполняемую с использованием базы данных SQL компании Azure. Основной целью данного исследования является потенциальное улучшение как сервиса, так и обслуживания клиентов с помощью контролируемой платформы.

Автоматическая система поиска неисправностей в базе данных предназначена для обнаружения проблем в реляционной облачной базе данных и обеспечения соответствующего анализа причин возникновения неполадок с целью сокращения времени и затрат на ручной поиск и решение данных проблем. Эта система была внедрена поверх платформы Microsoft Azure. Она основана на научных моделях общих и категориальных статистических данных, которые были разработаны и построены после тщательного анализа собранных телеметрических данных. Окончательная первопричина каждого текущего вопроса в сервисе Azure собирается после анализа результатов моделей с помощью экспертной системы. Результаты оценки показывают, что постоянное совершенствование инфраструктуры сократило время обработки примерно в 2 раза, в то время как количество интервалов увеличилось в два раза, что можно считать общим улучшением примерно в 4 раза.

2.2.1. Научные методы сбора данных и использование экспертных систем как инструменты повышения производительности баз данных

С началом эры машинных вычислений компьютерные системы претерпели ряд радикальных изменений. Первоначально центром вычислений любой компании или учреждения являлся локальный мейнфрейм, на котором размещались необходимые программные продукты. С развитием технологий и архитектур построения вычислительных машин мейнфреймы стали заменятся персональными компьютерами, работающими в общей сети. Основным преимуществом ПК перед мейнфреймом является гибкость использования вычислительных ресурсов, т.к. персональный компьютер имеет возможность запускать любые виды программного обеспечения, требуемого пользователем. Открывшиеся возможности повлекли за собой изменения в обработке массивов данных. Одним из таких изменений стало появление концепции

баз данных. Эта концепция разрешила вопрос о том, как хранить, модифицировать и потреблять большие объемы информации наиболее эффективно. Дальнейшее развитие сетевых технологий, в том числе и развитие сети интернет, позволило компаниям и вовсе отказаться от использования локальных серверов и внутренних сетей, им на смену пришли «облачные сервисы».

Базы данных, размещенные в «облаке», имеют ряд ограничений для пользователей в отличие от локальных. Одним из таких ограничений является отсутствие возможности использования собственных служб мониторинга систем баз данных и хост-машин. Данное ограничение обусловлено распределением нагрузок на вычислительные мощности дата-центров. Иначе говоря, системное администрирование систем баз данных уже не требует постоянного мониторинга, т.к. дата-центры автоматизированно выделяют требуемое количество ресурсов каждой конкретной системе в единицу времени. Наряду с требованиями по мониторингу, желательно также полностью разгрузить администраторов баз данных от таких действий, как первичная сортировка, поиск и устранение неисправностей и настройка базы данных. Таким образом, благодаря полной автоматизации сама система базы данных сможет достичь большей производительности. Научные методы сбора данных и использование экспертных систем представляются наиболее перспективным способом достижения данной цели.

2.2.2. Система автоматического поиска и устранения неисправностей

Система автоматического поиска и устранения неисправностей в базе данных (ADTS) состоит из трех основных компонентов [2.9-2.12]:

- научные модели данных;
- экспертная система;

- инфраструктура.

Каждая база данных генерирует различный набор статистики. Эти данные могут быть использованы для наиболее четкого понимания характера нагрузки, выполняемой на БД. Компонент "Научные модели данных" содержит набор различных правильно спроектированных научных моделей статистических данных, предназначенных для обнаружения специфического поведения, основанного на определенном наборе сигналов. Применяемый подход с использованием традиционных научных моделей статистических данных имеет основное преимущество в условиях, когда сотни различных сигналов поступают из более чем миллиона баз данных, в то время как помеченный набор данных крайне узок и имеет менее пятисот наблюдаемых случаев. Аналогичный подход с набором классификационных моделей в сложной среде был рассмотрен в [2.13]. Несколько иной подход был представлен в [2.14], где основное внимание уделялось автоматическому выявлению проблемы. Здесь проблемные периоды времени определяются пользователем системы, а не самой системой. Ответственность системы в этом случае заключается в поиске корреляции с другими сигналами и предоставлении потенциальных путей для дальнейшего исследования.

Как и во многих современных крупных системах, основанных на искусственном интеллекте, существует необходимость фильтрации выходных данных, получаемых с помощью научных моделей данных, для того чтобы определить наиболее точную причину неполадок. Для этого была разработана экспертная система, а ее правила были тщательно проработаны и оценены. Экспертная система, основанная на правилах, принимает окончательное решение о том, какие сигналы считать важными и публиковать для пользователей, а что не имеет отношения к делу и должно быть проигнорировано. Все модели, а также экспертная система работают поверх облачной инфраструктуры Microsoft Azure.

2.2.3 Результаты оценки

Оценка проводилась в среде баз данных Azure SQL с более чем миллионом баз данных за 8 месяцев (с паузой в 2 месяца в сборе данных). Эти базы данных имели различные рабочие характеристики и принадлежали разным клиентам, которые использовали их для различных целей.

Реализованное решение было проверено с нескольких точек зрения. Основные показатели улучшения можно увидеть с точки зрения клиентов, у которых общей целью было сокращение общей продолжительности выполнения запросов к базе данных в течение наблюдаемого периода времени. Другими важными факторами являются время, необходимое для обнаружения, обработки и сообщения о проблеме. Основным параметром, указывающим на размер проблемы, является количество обнаруженных интервалов, требующих обработки и анализа первопричин.

Оценка инфраструктуры

На первой итерации была создана базовая инфраструктура путем повторного использования существующей БД Azure SQL, в которой все выходные данные хранились в формате "ключ-значение". Этот формат был отмечен как одно из узких мест, и в ходе последующих итераций была смоделирована новая база данных для полного удовлетворения всех потребностей, требуемых этой системой. В начале в среднем на один интервал уходило 5 секунд, а в последующем наблюдаемом периоде времени время на обработку интервала сократилось до 2,5 секунд, хотя количество обработанных интервалов увеличилось в 2 раза, что представляет собой общее ускорение в 4 раза. Это улучшение объясняется инновационной моделью базы данных и повышением эффективности переработанных научных моделей данных. Соответственно, с точки зрения инфраструктуры очень важно отслеживать время, затрачиваемое на

обнаружение, обработку и публикацию результатов с точки зрения их сквозного использования [2.15, 2.16].

2.2.4. Оценка моделей и экспертной системы

Выходные данные моделей и экспертной системы также постоянно отслеживались и контролировались. Количество итераций в час показывает, сколько раз конкретная модель была выполнена в мире в течение одного часа, а количество обнаруженных интервалов - количество обнаруженных интервалов для уникальных баз данных. Эти два значения для типовых моделей показаны на рис. 2.4.

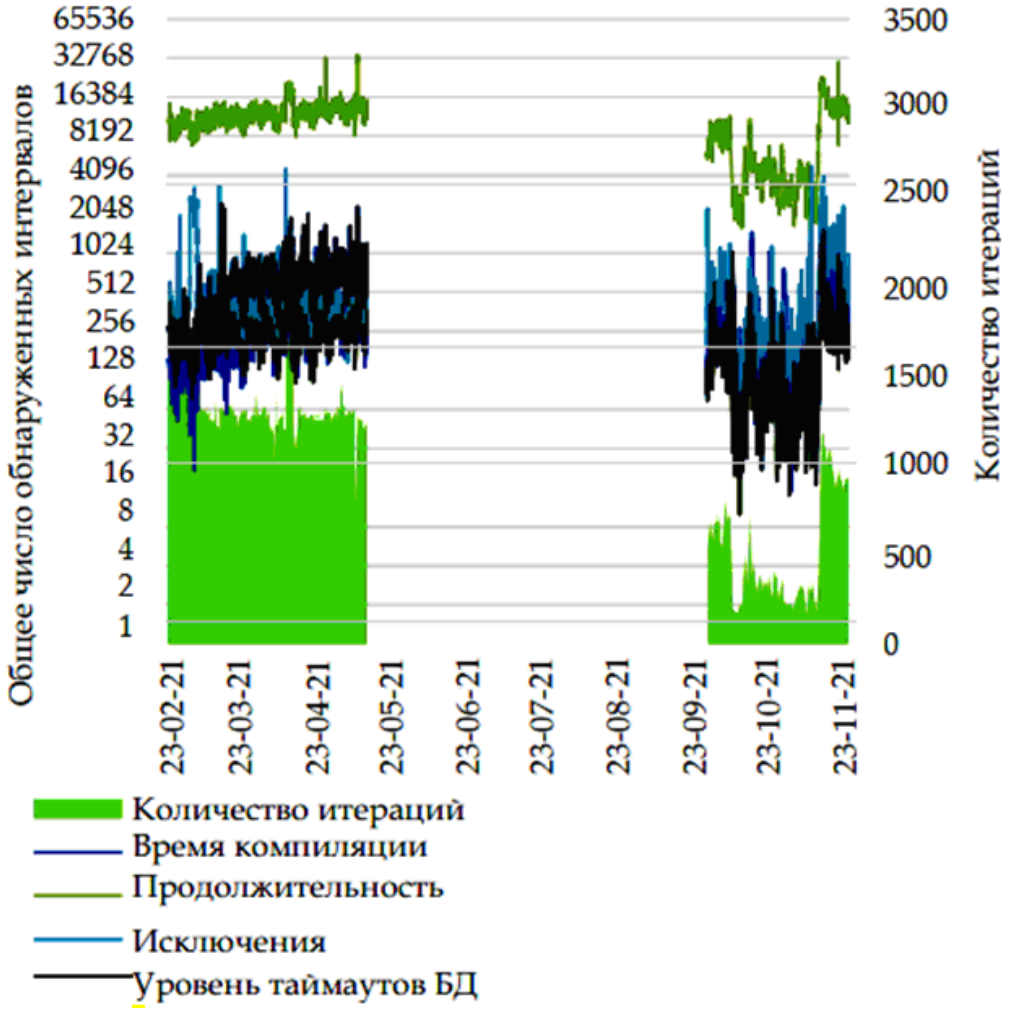

Рис. 2.4. Выполнение общей модели в течение оценочного периода

Очевидно, что в начале второго анализируемого периода (с 30

сентября) количество итераций было меньше, чем раньше. Это можно объяснить тем, что в то время был внедрен набор усовершенствований, которые повысили надежность и сократили время исполнения. Поскольку количество регионов и баз данных в сервисе после этого увеличилось, количество итераций вернулось примерно на 1000 в час. Что касается детектирования в час для данной модели, то можно заметить, что большинство обнаруженных интервалов происходит от модели Duration (более 8000 интервалов в час), в то время как другие типичные модели обнаруживают только от 200 до 1000 интервалов. Также видна корреляция между увеличением количества итераций и количеством обнаруженных интервалов для конкретной модели с 9 по 12 мая. После тщательного анализа телеметрии был сделан вывод, что один клиент с большим набором баз данных испытал существенное снижение производительности, на которое повлияли изменения, внесенные в базу данных и дизайн приложений, которые также были обнаружены системой автоматического поиска и устранения неисправностей в базе данных (ADTS).

После того, как обобщенная модель обнаруживает интервал с формой отклонения, дальнейшая обработка и анализ выполняется классифицированными моделями. На рис. 2.5 представлено количество обнаруженных категорий для заданного количества интервалов во временном масштабе.

Три наиболее частые из обнаруженных проблемы за обработанные интервалы попадают в общую группу проблем. Такие категории, как Hitting resource limit, Hitting worker limit, PileUp и Locking, могут представлять серьезную проблему для обычного администратора баз данных. Эти категории делятся на общие проблемы, которые требуют более глубоких технических знаний, чтобы точно определить, в чем заключается реальная проблема. Способность обнаруживать и объяснять такие случаи является дополнительным преимуществом данной системы.

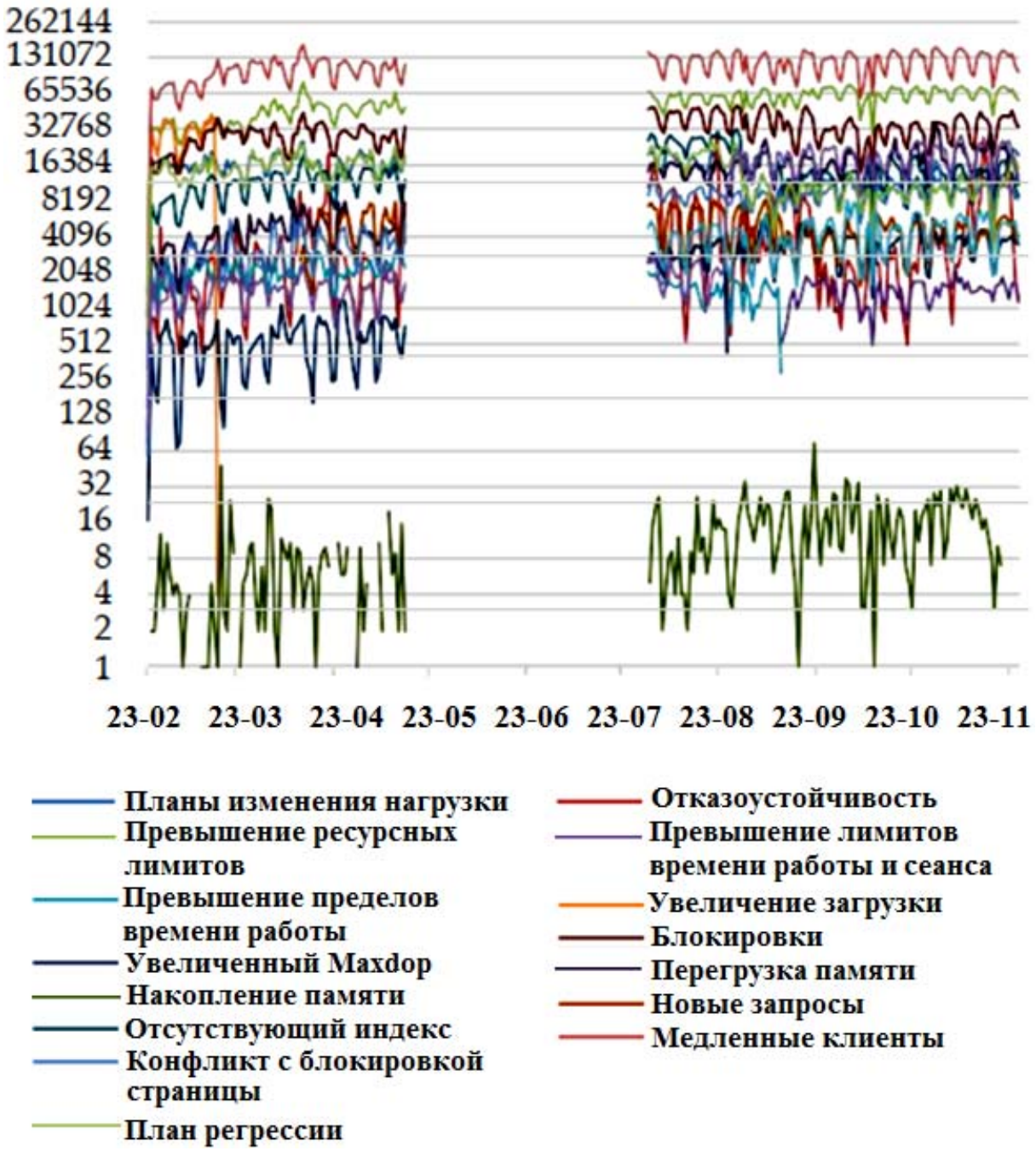

Рис. 2.5. Обнаружения классификационной модели в течение оценочного периода

Наконец, интервалы с обнаруженными категориями затем обрабатываются экспертной системой, чтобы получить конечную первопричину. С точки зрения экспертной системы, большинство интервалов (>85%) совпадают с правилом одной экспертной системы,

которое соответствует точно определенной первопричине. На рис. 2.6 представлено распределение количества приведенных в соответствие правил для всех наблюдаемых интервалов в течение 8 месяцев (более 45 млн. обработанных интервалов), как для исходного набора моделей, так и для обновленной и улучшенной версии моделей. Можно сделать вывод о высоком качестве правил экспертной системы, так как почти 99% наблюдаемых вопросов совпадает не более чем с 3 правилами. Этот результат дает достаточно точный анализ первопричин для конечных пользователей сервиса баз данных Azure SQL.

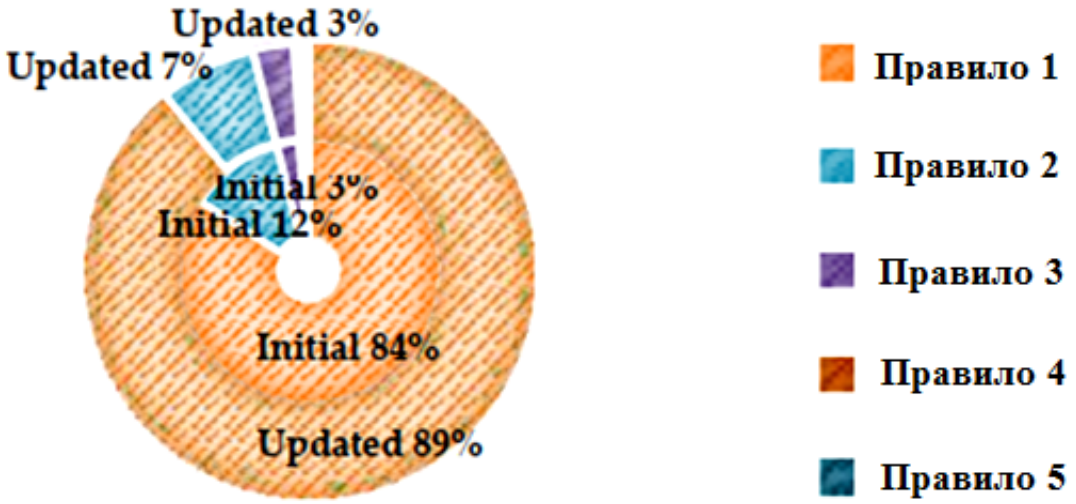

Рис. 2.6. Распределение количества приведенных в соответствие правил для всех наблюдаемых интервалов в течение 8 месяцев

2.2.5. Общее улучшение базы данных как сервиса

Общий опыт использования системы проиллюстрирован двумя примерами из реальной жизни. Два клиента с различными рабочими нагрузками были одними из первых пользователей автоматической системы поиска и устранения неисправностей. Рабочая нагрузка первого клиента (КЛ-1) была распределена по нескольким базам данных. Таким образом, поведение рабочей нагрузки варьировалось для разных пользователей клиентской платформы. В такой среде ADTS обнаружила

несколько ошибок в клиентском приложении и базе данных. О них было сообщено клиенту ADTS. После того, как клиент применил соответствующие исправления к приложению и базе данных были заметны улучшения в работе системы. Эти улучшения наблюдались не только с точки зрения платформы ADTS, но и с точки зрения клиента, который поделился положительным подтверждением с командой. Общее улучшение с точки зрения системы для КЛ-1 показано на рис. 2.7.

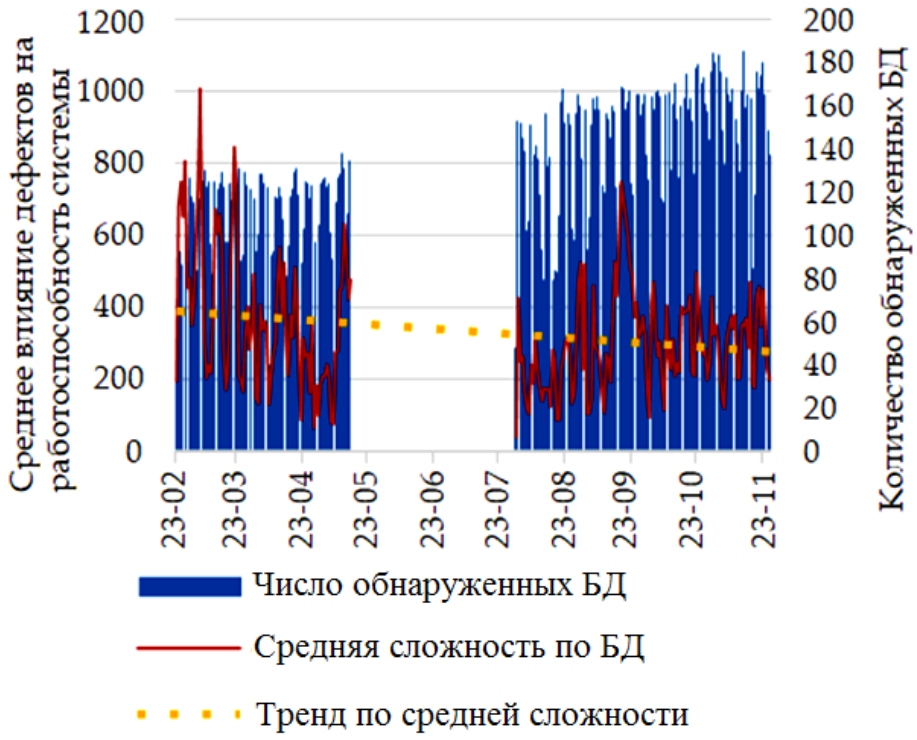

Рис. 2.7. Улучшение использования системы БД КЛ-1 в течение оценочного периода

Видно, что количество уникальных баз данных увеличилось, а среднее влияние дефектов на работоспособность системы со временем снизилось. Это означает, что наблюдаемые проблемы на более поздних этапах, когда клиент начал использовать ADTS, были не такими серьезными, как раньше, и большинство серьезных проблем были устранены в кратчайшие сроки.

Как и в случае с КЛ-1, второй клиент (КЛ-2) также имел архитектуру

с несколькими базами данных. Однако, в отличие от КЛ-1, у КЛ-2 не было разных пользователей своей платформы. Все рабочие нагрузки, которые выполнялись с этим набором баз данных, были одинаковыми. В данном случае проблемы возникали из-за неоптимального использования каждой БД приложением данного клиента. ADTS предоставил анализ первопричин того, что у КЛ-2 возникли проблемы с временными базами данных, или, другими словами, временная база данных использовалась чрезмерно. В таких случаях использование функции SQL-сервера, называемой «inmemory database», снижает нагрузку на временную базу данных, и пропускная способность базы данных становится намного выше. Общее улучшение для КЛ-2 представлено на рис. 2.8.

Рис. 2.8. Улучшение для КЛ-2 в течение оценочного периода

Наблюдается значительный всплеск 23 марта, о котором КЛ-2 был мгновенно проинформирован. Дальнейший анализ первопричин ADTS

показал, что всплеск был вызван несколькими ошибками в логике клиентских приложений, что накладывало неоптимальное использование базы данных и вызывало проблемы с производительностью для остальной части. Кроме того, в более поздний период после июля 2021 г. количество баз данных значительно увеличилось, что явилось результатом того, что клиент масштабировал свое решение, хотя и не все проблемы с производительностью баз данных были решены. Тем не менее, выявленные проблемы имели меньшее общее влияние дефектов на работоспособность системы. Клиент настроил свою рабочую нагрузку и подтвердил, что это было выполнено на основе информации, полученной от ADTS.

Из 2463 уникальных клиентов 1103 имели проблемы в 48117 базах данных, которые наблюдались и о которых ADTS сообщало. На рис. 2.9 представлено общее среднее влияние дефекта на работоспособность системы (красная линия), которое показывает общее улучшение для этих клиентов в заданный период времени.

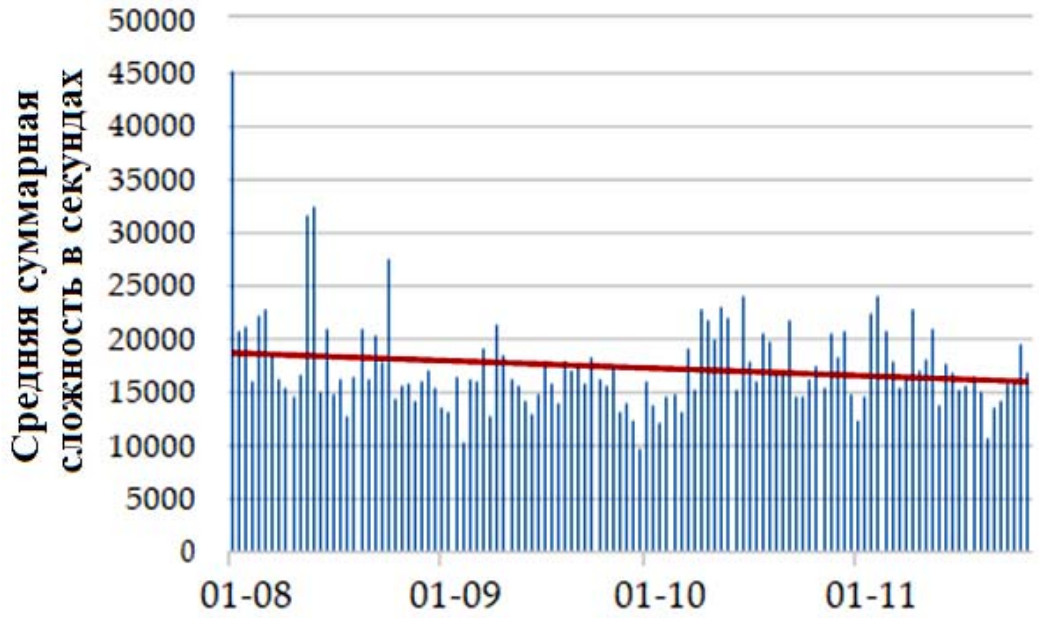

Рис. 2.9. Общие улучшения пользователей ADTS

Используя данные выборки во всех трех случаях (КЛ-1, КЛ-2 и

общие), были проведены два выборочных t-теста на среднее влияние дефекта на работоспособность системы временного ряда. В общем случае мы видим явное снижение среднего влияния дефектов на работоспособность системы до 11 октября, когда произошло обнаруженное этой системой ухудшение в SQL-сервере.

2.2.6. Результаты

Автоматическая система поиска неисправностей в базе данных предназначена для обнаружения проблем в реляционной облачной базе данных и обеспечения соответствующего анализа причин возникновения неполадок с целью сокращения времени и затрат на ручной поиск и решение данных проблем. Эта система была внедрена поверх платформы Microsoft Azure. Она основана на научных моделях общих и категориальных статистических данных, которые были разработаны и построены после тщательного анализа собранных телеметрических данных [2.17, 2.18]. Окончательная первопричина каждого текущего вопроса в сервисе Azure собирается после анализа результатов моделей с помощью экспертной системы. Результаты оценки показывают, что постоянное совершенствование инфраструктуры сократило время обработки примерно в 2 раза, в то время как количество интервалов увеличилось в два раза, что можно считать общим улучшением примерно в 4 раза.

2.3. Выводы

1. В перспективных сетях распределенное сетевое кэширование станет базовой возможностью. Предлагается эффективный механизм распределения данных о специальных транзакциях в реальном времени, основанный на динамических сетевых статусах в реальном времени. Этот механизм может эффективно избежать таких проблем, как высокая задержка, вызванная традиционной передачей в облаке, и может

значительно повысить эффективность передачи данных о специальных транзакциях, особенно для требований к операциям с контентами в режиме реального времени.

2. Современные приложения ориентированы на облачные сервисы для достижения лучшей производительности, географической репликации и снижения стоимости владения. Следуя современным концепциям облачных сервисов, данное исследование использует богатые телеметрические данные и отображает рабочую нагрузку, выполняемую с использованием базы данных SQL компании Azure. Основной целью данного исследования является потенциальное улучшение как сервиса, так и обслуживания клиентов с помощью контролируемой платформы.

3. Исследование выявило некоторые потенциальные улучшения. Постобработка извлеченной телеметрии увеличивает время обнаружения. В идеальном решении это время должно составлять не минуты, а секунды. Этого можно добиться путем запуска типовых моделей внутри или рядом с базой данных на платформе. Другим подходом, заслуживающим внимания, может быть введение набора быстрых моделей, которые будут уведомлять о существовании проблемы без детального анализа ее причин. С точки зрения, как заказчика, так и системы, еще одним полезным направлением могло бы стать предоставление заказчику возможности поделиться отзывами о полезности предоставленной информации и удовлетворенности ею.

4. Представлен механизм распределения данных о специальных транзакциях с оперативным контентом, отличающийся распределенным кэшированием в гетерогенных межобъектных интерфейсах в реальном масштабе времени, обеспечивающий минимизацию времени передачи данных

5. Описан алгоритм локального кэширования хронологически запрошенных данных транзакций, отличающийся учетом собственной

емкости межобъектного интерфейса и применением политикой вытеснения давно неиспользуемых данных, обеспечивающий уменьшение время задержки передачи данных о специальных транзакциях.

6. Представлен «жадный» алгоритм разделения и перенаправления запросов между кдиентами, межобъектными интерфейсами или облаком, отличающийся интеграцией информации о состоянии кэширования, предполагаемом размере данных и пропускной способности и обеспечивающий минимизацию задержки передачи данных транзакции. Этот алгоритм заключается в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Список источников к главе 2

2.1. Xiaolin L. Real time regression analysis in internet of stock market cycles// Cognit Syst Res. 2018; 52: 371-379.

2.2. Zhao Z., Min G., Gao W., Wu Y., Duan H., Ni Q. Deploying computing nodes for large-scale IoT: a diversity aware approach// IEEE Internet Things J. 2018; 5(5): 3606-3614.

2.3. Brzeszczynski J., Ibrahim B.M. A stock market trading system based on foreign and domestic information// Expert Syst Appl. 2019; 118: 381-399.

2.4. Mach P., Becvar Z. Cloud-aware power control for real-time application offloading in mobile computing// Trans Emerg Telecommun Technol. 2016; 27(5): 648-661.

2.5. de Assuncao M.D., da Silva V.A., Buyya R. Distributed data stream processing and МОИН computing: a survey on resource elasticity and future directions// J Netw Comput Appl. 2018; 103: 1-17.

2.6. Ao W.C., Psounis K. Fast content delivery via distributed caching and small cell cooperation// IEEE Trans Mobile Comput. 2018; 17(5): 1048-1061.

2.7. Dar S., Franklin M.J., Jonsson B.T., Srivastava D., Tan M. Semantic Data Caching and Replacement// Proc. of the 22 VLDB Conf., Mumbai-Bombay, India, 1996, p. 330-341.

2.8. Zipf G.K. Human Behavior and the Principle of Least Effort. - Addison-Wesley Press, 1949. 573 c.

2.9. Данилов А.Д., Синюков Д.С. Механизм распределения данных о специальных транзакциях с оперативным контентом в реальном времени на основе кэширования в гетерогенных объектах распределенной сети. Информационные технологии моделирования и управления. 2021;125(3):216-223.

2.10. Данилов А.Д., Синюков Д.С. Подход к управлению транзакциями в гетерогенных распределенных реплицированных системах

баз данных в реальном масштабе времени. Системы управления и информационные технологии. 2021;85(3):59-65.

2.11. Синюков Д.С., Данилов А.Д. Применение систем управления базами данных как сервиса в сложных информационных системах. Труды Всероссийской научной конференции «Достижения науки и технологий-ДНиТ-2021», Красноярск. 2021; http://ru-conf.domnit.ru/media/filer_public/42/d3/42d35c66-d78a-411c-b74d-9ef2f99ff7b6/3006-dnit-2021.pdf.

2.12. Sinyukov D.S. Problems of troubleshooting in databases. Modern informatization problems in simulation and social technologies (MIP-2022'SCT): Proceedings of the XXVII-th International Open Science Conference. Yelm, WA, USA. 2022;162-174.

2.13. Jeyakumar V., Madani O., Parandehgheibi A., Kulshreshtha A., Zeng W., Yadav N. ExplainIt! - A declarative root-cause analysis engine for time series data. SIGMOD'19 2019 International Conference on Management of Data, Amsterdam, Holland. 2019;333-348.

2.14. Raeder T., Dalessandro B., Provost F. Design principles of massive, robust prediction systems. 18th ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China. 2012;1357-1365.

2.15. Automatic Performance Diagnostics. Oracle. 2017; https://docs.oracle.com/database/121/TGDBA/pfgrf_diag.htm#TG DBA026.

2.16. Automatic SQL tuning. Oracle, 2018; https://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm #CHDJDFGE.

2.17. Oleinikova S.A., Kravets O.Ja., Aksenov I.A., Frantsisko O.Yu., Rahman P.A., Atlasov I.V. The general scheme of the genetic algorithm for solving the task scheduling problem for a multistage system and assigning time for jobs. International Journal on Information Technologies and Security. 2021;13(4):47-58.

2.18. Mustafayev V.A., Zeynalabdiyeva I.S., Kravets O.Ja. Control model of parallel functioning production modules as fuzzy Petri nets. Journal of Physics: Conference Series. 2021;2094: 022003. https://doi.org/10.1088/1742-6596/2094/2/022003.

3. Управление транзакциями в гетерогенных распределенных реплицированных системах баз данных в реальном масштабе времени

3.1. Архитектура СУБД реального времени, основанная на управлении с обратной связью

Для борьбы с фазами нестабильности в СУБД реального времени (РВ) из-за непредсказуемого поступления транзакций пользователей предложена архитектура, основанная на обратной связи, которая рассматривается как базовая архитектура для управления QoS в СУБД РВ.

3.1.1. Особенности данных

База данных, которые рассматриваются в предлагаемой архитектуре, содержит данные как в реальном времени и не в реальном времени. Данные в реальном времени - это данные датчиков, которые описывают объекты реального мира. Они должны регулярно обновляться, чтобы более точно отражать реальное состояние дел в мире. Каждый объект данных реального времени имеет интервал действия, после которого он устаревает, а затем становится непригодным для использования. Он также имеет метку времени, которая является датой его последнего обновления. Данные в реальном времени могут иметь отклонение от своего значения в реальном мире, называемое ошибкой данных (DE). Значение DE не должно превышать порогового значения, которое мы обозначаем MDE (максимальная ошибка данных). Данные, не относящиеся к реальному времени, находятся в обычных базах данных и не изменяются динамически со временем.

3.1.2. Особенности транзакций

Что касается данных, то в СУБД РВ транзакции в реальном времени

должны соответствовать ограничениям по времени, с которыми они связаны. В предлагаемой архитектуре рассматриваются транзакции с жестким предельным сроком [3.7], когда, если транзакция пропустит свой предельный срок, она будет прервана и станет бесполезной для системы. Будем различать два типа транзакций в реальном времени в зависимости от объекта данных, к которому осуществляется доступ: обновленные транзакции и транзакции пользователей. Обновленные транзакции - это те, которые периодически выполняются для обновления данных в реальном времени. Однако пользовательские транзакции, которые являются апериодическими, состоят из набора операций чтения как с данными в реальном времени, так и с данными, не относящимися к реальному времени, и операций записи только с данными, не относящимися к реальному времени.

3.1.3. Метрика оценки производительности

Основным критерием эффективности, рассматриваемым в предлагаемой архитектуре, является уровень удовлетворенности пользователей системы. Он представляет коэффициент успешности (SR) транзакций, которые квалифицируются как критические для пользователей. SR определяется как параметр QoS, измеряющий процент транзакций, которые соответствуют своим срокам, и имеющий следующую формулу:

$$\mathrm{SR} = 100 \times \frac{\#\,\mathrm{Timely}}{\#\,\mathrm{Late} + \#\,\mathrm{Timely}} (\%) \qquad (3.1)$$

где #Late и #Timely соответственно означают количество критических транзакций, которые пропустили свои предельные сроки, и количество успешных критических транзакций.

3.1.4. Управление QoS в СУБД РВ

QoS имеет основополагающее значение для оценки производительности человеко-машинных систем. В [3.8] предложена архитектура, использующая управление планированием на основе обратной связи для управления QoS в СУБД РВ. Она называется FCSA (архитектура управления планированием с обратной связью), и она позволяет контролировать поведение СУБД РВ, делая его более надежным в периоды нестабильности. На рис. 3.1 показана FCSA, которая будет кратко описана далее.

Рис. 3.1. Базовая архитектура СУБД РВ, основанная на управлении с обратной связью

Контроллер допуска отвечает за фильтрацию транзакций пользователей. Его функционирование контролируется контуром обратной связи для управления в периоды нестабильности системы. Количество принятых транзакций зависит от состояния системы, о котором сообщает монитор, и от параметров QoS, заданных администратором базы данных (DBA).

Принятые транзакции отправляются в очередь готовности для исполнения обработчиком транзакций. Обработчик транзакций состоит из диспетчера актуальности (FM), проверяющего актуальность данных в реальном времени перед доступом к транзакциям, контроллера параллелизма (CC), разрешающего конфликты данных доступа, возникающие между транзакциями, и базового планировщика (BS), определяющего время выполнения каждой транзакции. Монитор отвечает за измерение производительности системы на основе результатов выполнения, предоставляемых обработчиком транзакций.

Эти измерения относятся к контуру обратной связи и отправляются менеджеру QoS для информирования его о состоянии системы. Используя эти значения и справочные параметры системы, установленные администратором базы данных, менеджер QoS корректирует значения параметров QoS, которые будут отправлены контроллеру допуска и менеджеру QoD (качество данных). Этот параметр настроит значение MDE, а затем отправит его на контроллер актуальности, который используется для удаления транзакции обновления, обращающейся к данным, которые считаются устаревшими (DE<MDE).

Контур обратной связи, обеспечивающий балансировку системы, основан на принципе наблюдения и самоадаптации. Самоадаптация происходит на протяжении всего функционирования системы, чтобы постоянно корректировать ее рабочую нагрузку при наличии непредсказуемых запросов пользователей. Однако наблюдение

заключается в рассмотрении состояния системы, чтобы определить, соответствует ли оно заданным параметрам QoS, таким как коэффициент использования системы и коэффициент успешности транзакций. Система адаптирует свои настройки к соответствующим изменениям в поведении и, таким образом, справится с периодами нестабильности.

В [9] цикл обратной связи использовался для управления качеством обслуживания в крупномасштабных приложениях реального времени в архитектуре под названием DRACON (децентрализованная репликация и управление данными). Кроме того, в [10] эта методика адаптирована для управления QoS встроенных баз данных в реальном времени.

3.1.3. Управление транзакциями

Было предложено несколько политик планирования транзакций, среди которых есть онлайновые и автономные политики. Динамическое планирование выбирает следующий процесс, который будет запланирован в любое время выполнения приложения, в соответствии с информацией о транзакциях, инициированных в это время. Однако статическое планирование осуществляется заранее, поэтому оно не адаптируется к изменениям окружающей среды. Аналогичным образом, различаются упреждающие политики, позволяющие прерывать выполнение процесса, от тех, которые не являются упреждающими, когда процесс выполняется до тех пор, пока он не завершится или не будет заблокирован [3.11, 3.12].

Среди предлагаемых обычных протоколов планирования есть протокол First In First Out (FIFO), который позволяет запускать каждый процесс в соответствии с датой его прибытия. Таким образом, процесс избрания состоит в выборе первого прибывшего. Для процессов, поступающих в одно и то же время, FIFO случайным образом выбирает один из них. Однако в протоколе циклического перебора (RR) для каждого процесса, в течение которого ему разрешено выполняться, выделяется

квант времени Q. Любой процесс, потреблявший Q, должен покинуть процессор и быть поставлен в конец списка. Следовательно, все процессы имеют одинаковый приоритет. Однако, когда дело доходит до приложений реального времени, где транзакции ограничены по времени, эти протоколы не являются полезными.

Чтобы учесть временные ограничения транзакций в режиме реального времени, существуют иные политики планирования. Самый ранний предельный срок (EDF) [3.13] является одним из основных протоколов планирования, предлагаемых для контекста реального времени. Он заключается в присвоении наивысшего приоритета транзакции, имеющей ближайший предельный срок.

Таким образом, транзакции выполняются в порядке возрастания их предельных сроков, чтобы увеличить их возможности для успешного выполнения. В [3.14] расширен протокол EDF и предложена политика планирования GEDF (обобщенный самый ранний предельный срок). В GEDF распределение приоритетов основано на важности транзакций и их сроках. В [3.15] предложен адаптивный алгоритм совместного планирования в реальном времени, основанный на EDF. Это называется адаптивным первым совместным планированием с самым ранним предельным сроком (AEDF-Co), и он направлен на планирование транзакций таким образом, чтобы были соблюдены ограничения по срокам и был максимизирован QoD.

При использовании протоколов планирования, упомянутых выше, для принятия решения о следующей транзакции, которая должна быть выполнена, учитывается несколько критериев. Большинство из этих критериев в основном учитывают временные ограничения транзакций, которые должны выполняться в контексте реального времени, например, предельные сроки. Однако выбор пользователя никогда не учитывается при определении времени выполнения транзакций. Таким образом,

предлагается включить этот критерий в определение приоритетов транзакций, направленных на достижение удовлетворенности пользователей.

3.1.4. Модифицированный протокол управления транзакциями в СУБД РВ

Подход заключается в том, чтобы предложить модифицированный протокол планирования транзакций в СУБД РВ. Модификация заключается в учете не только временных ограничений транзакций, но и критериев, установленных пользователями базы данных. Функционирование этого протокола основано на использовании трех параметров для определения приоритетов выполняемых транзакций.

Первый параметр - это предельные сроки транзакции, которые должны быть соблюдены для ее совершения.

Второй параметр представляет время поступления каждой транзакции в систему.

Третий параметр служит показателем уровня критичности для каждой транзакции и отражает, в частности, ее важность для заинтересованного пользователя.

Поэтому считаем наиболее приоритетной транзакцию, имеющую как самый ранний предельный срок, самое раннее время прибытия, так и самый высокий уровень критичности. Таким образом, определим следующим образом интегральный приоритет каждой транзакции t_i:

$$Priority(t_i) = 1/DL(t_i) + 1/AT(t_i) + CR(t_i) \quad (3.2)$$

где $DL(t_i)$ представляет предельный срок транзакции t_i, $AT(t_i)$ - время ее прибытия, а $CR(t_i)$ - уровень ее критичности. Этот модифицированный протокол MRTS (смешанное планирование в реальном времени), следует принципу политики упреждающего планирования. Таким образом, транзакция с более низким приоритетом может быть прервана в пользу

другой транзакции с более высоким приоритетом. Однако упреждение может быть осуществлено только после проверки того, что оно не приведет к приостановке транзакции с пропущенным крайним сроком. Это означает, что приостановленная транзакция может быть возобновлена позже с момента ее приостановки.

Итак, представлен модифицированный протокол планирования транзакций в СУБД реального времени, учитывающий не только временные ограничения транзакций, но и критерии, установленные пользователями базы данных, обеспечивающий корректное определение приоритетов выполняемых транзакций.

Протокол MRTS применен в FCSA, который предлагается для управления QoS в СУБД РВ. Эта архитектура обеспечивает хорошую производительность для стабилизации системы в периоды перегрузки или недостаточного использования, вызванного неожиданным поступлением пользовательских транзакций. На рис. 3.2 показана архитектура FC-MRTS, содержащая модифицированный планировщик.

Задача предложенного подхода состоит в том, чтобы удовлетворить пользователей СУБД РВ за счет улучшения качества предоставляемых услуг, а именно за счет увеличения числа транзакций, которые соответствуют их срокам, особенно тех, которые квалифицируются как более важные для пользователей. Кроме того, совместно с FCSA снижается риск перегрузки системы, когда одновременно происходит значительное количество транзакций.

В дальнейших исследованиях планируется ввести в подход новый параметр, отражающий связи агрегирования, которые могут существовать между транзакциями. Поэтому, если транзакция t_i является субтранзакцией t_j, необходимо начать с запуска t_i, даже если она имеет более низкий приоритет. Реализация такого решения требует анализа взаимосвязей транзакций, которые обычно недостаточно очевидны.

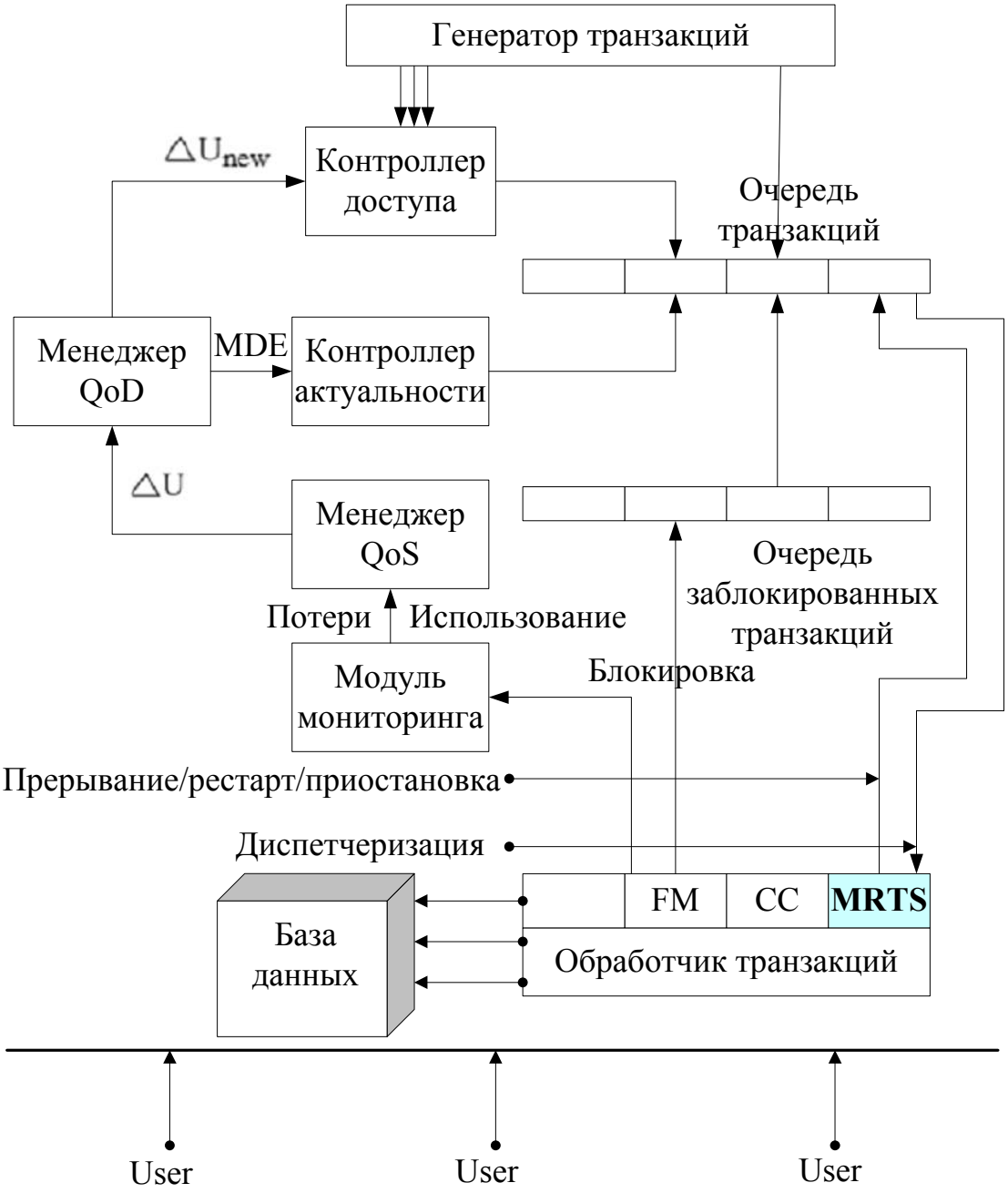

Рис. 3.2. Протокол MRTS в FCSA

Таким образом, разработана архитектура СУБД реального времени, отличающаяся применением модифицированного протокола планирования транзакций и улучшающая качество предоставления услуг пользователям при реализации управления с обратной связью.

3.1.5. Результаты имитационного моделирования

В этом разделе оценивается эффективность предложенного подхода в соответствии с результатами проведенного имитационного моделирования.

Принципы моделирования

Чтобы оценить предложенный подход к планированию, использован симулятор, основанный на FCSA и имитирующий поведение СУБД РВ. Проведен ряд экспериментов, в которых варьировали значения некоторых параметров. Системные параметры, используемые для генерации данных и транзакций, обобщены в табл. 3.1.

Таблица 3.1

Системные параметры

Параметр	Содержание	Значения
Duration (мс)	Длительность имитационного моделирования	10000
Validity (мс)	Время действия данных	[400, 800]
NbOfOperations	Число транзакций	[3, 10]
ReadOpTime	Время выполнения транзакции чтения	[1, 2]
WriteOpTime	Время выполнения транзакции записи	[1, 10]
MDE	Максимальное количество ошибок данных	3

Для разрешения конфликтов, возникающих между транзакциями, выбран протокол управления параллелизмом 2PL-HP (двухфазная фиксация с высоким приоритетом) [3.16]. Его принцип заключается в обеспечении того, чтобы выполнение транзакции с высоким приоритетом не затруднялось выполнением другой транзакции с более низким приоритетом.

Первый набор экспериментов состоит в моделировании поведения СУБД РВ с использованием протокола планирования EDF. Далее

используется протокол FIFO и эксперимент заканчивается оценкой предложенного протокола MRTS. В табл. 3.2 приведены значения коэффициентов удовлетворенности пользователей в зависимости от размера базы данных с использованием трех политик планирования, которые реализованы в симуляторе. Графическое представление табл. 3.2 приведено на рис. 3.3.

Таблица 3.2

Результаты моделирования трех планировщиков. Коэффициент удовлетворенности пользователей (%)

Объем данных	EDF	FIFO	MRTS
100	68	20	73
200	50	19	64
300	41	16	62
400	31	17	59
500	30	19	51

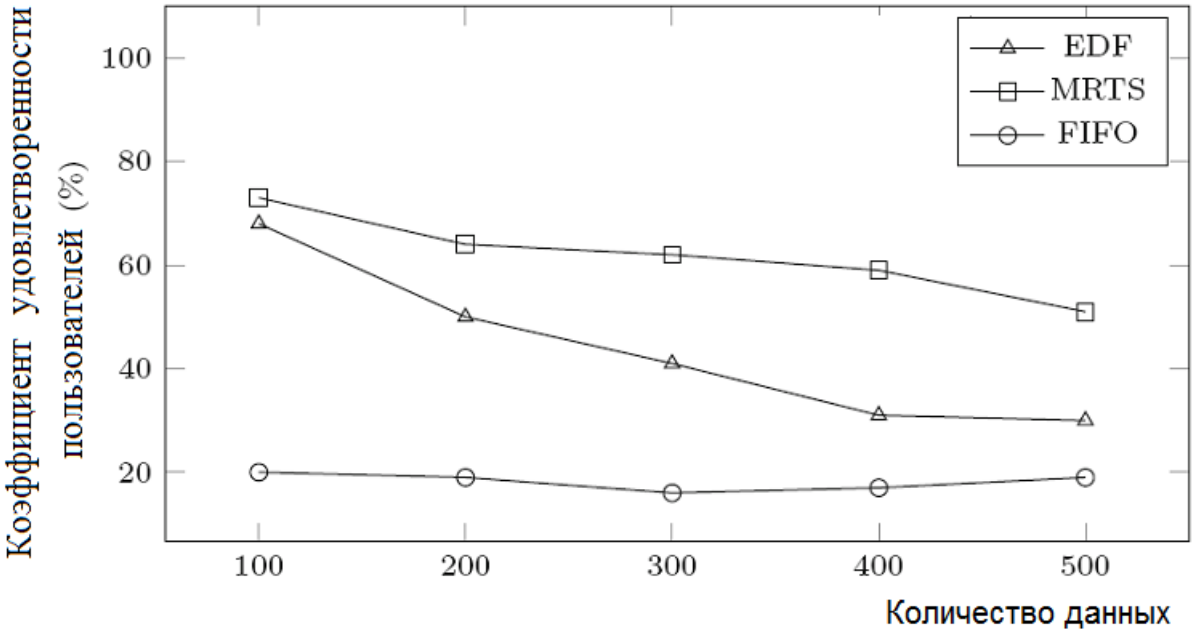

Рис. 3.3. Коэффициент удовлетворенности пользователей для трех планировщиков

В соответствии с рис. 3.3 показатели удовлетворенности пользователей политикой MRTS, предложенной в описанном подходе, всегда лучше, чем у планировщика EDF или FIFO. Это связано с тем, что предложенный подход учитывает, в дополнение к предельным срокам, даты прибытия транзакций, чтобы минимизировать время их ожидания для выполнения, и особенно уровни их критичности, определенные пользователями. Это не относится к другим планировщикам. Таким образом, пользователи более удовлетворены MRTS, которая продвигает их транзакции с наивысшими приоритетами.

Также отметим, что на коэффициент удовлетворенности пользователей влияет увеличение количества входящих транзакций в систему. В этом случае нагрузка на систему возрастает, а следовательно, возрастает и количество конфликтов при доступе к данным. Таким образом, отмечается снижение результатов у всех с трех планировщиков. Тем не менее, новый планировщик по-прежнему показывает наилучшие результаты. Планировщик FIFO по-прежнему имеет худшие результаты, поскольку он не учитывает предельные сроки транзакций, что делает его не очень подходящим в контексте реального времени.

3.2. Файловый подход к управлению транзакциями в гетерогенных распределенных реплицированных системах баз данных в реальном масштабе времени

3.2.1. Транзакции как гетерогенные файлы

В работе транзакции рассматриваются в виде гетерогенных файлов от пользователей в качестве входных данных и эффективно развертываются в облачной системе хранения данных и в гетерогенных системах баз данных в качестве выходных данных.

В работе после анализа гетерогенной рабочей нагрузки классификация задач выполняется с использованием алгоритма кластеризации k-средних. С использованием стандартной кластеризации k-средних разнородную рабочую нагрузку можно разделить на несколько классов задач со схожими характеристиками с точки зрения ресурсов и целей производительности. После анализа рабочей нагрузки можно обнаружить, что ее состав весьма неоднороден и динамичен с течением времени. В работе рассматривается неоднородность рабочей нагрузки.

В частности, сначала используется алгоритм кластеризации k-средних для разделения рабочей нагрузки на отдельные классы задач с аналогичными характеристиками с точки зрения требований к ресурсам и производительности. Система учитывает неоднородный характер рабочей нагрузки и выполняет следующие три операции:

1. Разделение задач на классы задач с использованием алгоритма k-средних, основная цель - понять характеристики рабочей нагрузки

2. Как только характеристика рабочей нагрузки будет получена, выделить ресурс, т.е. виртуальные машины, для описанной задачи.

3. Выполнить операции по обеспечению безопасности задачи перед загрузкой в облачные системы хранения.

Для тестирования идеи структуры используется облако Amazon AWS. Работа в основном сосредоточена на характеристике рабочей нагрузки с использованием кластеризации входной рабочей нагрузки, распределении ресурсов с помощью AWS EC2 и выполнении операций безопасности, т.е. шифрования и дешифрования назначенной работы. Базовая модель фреймворка показана на рис. 3.4.

Предлагаемый подход содержит три важные модели:

1) модель входа;

2) модель исполнения:

а) кластеризация;

б) распределение ресурсов;

в) безопасность, т.е. шифрование и дешифрование;

3) выходная модель.

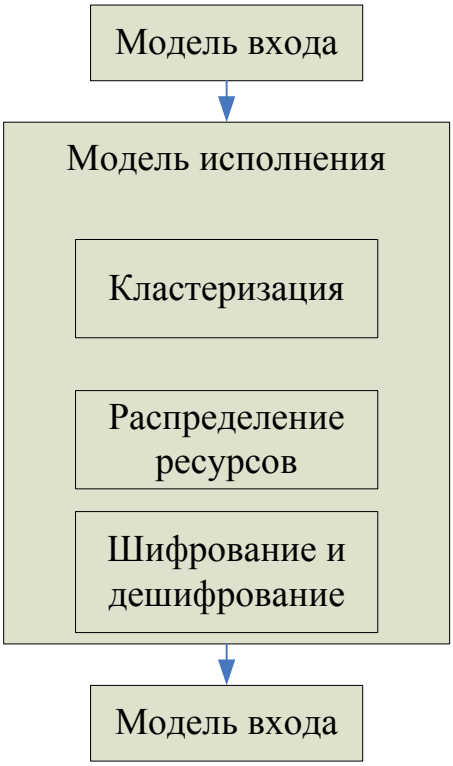

Рис. 3.4. Базовая модель фреймворка

Платформа взаимодействует с облаком Amazon AWS для использования различных сервисов, предоставляемых Amazon:

а) Эластичное вычислительное облако (EC2): EC2 используется для распределения ресурсов, т. е. виртуальных машин

б) DynamoDB: DynamoDB используется для создания облачной базы данных, в которой хранится важная информация о пользователях и файлах.

в) Простая система хранения (S3): Платформа выполняет операцию шифрования файлов, и эти зашифрованные файлы хранятся в S3. Поэтому для хранения используется S3, где для используемой платформы создается специальное пространство, называемое буфером.

Платформа использует базу данных MySQL для хранения информации, связанной с пользователями и файлами. В MySQL создается

база данных "транзакций", представляющая собой набор различных таблиц. Платформа использует алгоритм кластеризации k-средних для характеристики рабочей нагрузки и для определения классов задач. Эти классы формируются в зависимости от типа файла, размера файла и т.д. На основе этого создаются два кластера. Этим кластерам выделяются ресурсы, т. е. виртуальные машины, с помощью Amazon EC2 cloud.

3.2.2. Параметры эксперимента

Платформа реализована в среде Netbeans IDE 8.2. Он содержит различные файлы для кластеризации, распределения ресурсов и обеспечения безопасности. Платформа запрашивает у пользователей регистрационные данные. Вся эта информация о пользователе хранится как в базах данных, т.е. в базе данных MySQL, так и в базе данных Amazon AWS DynamoDB. Таким образом, сервер MySQL 5.1 устанавливается с помощью установочного файла mysql-essential-5.1.54-win32. Для использования базы данных MySQL с графическим пользовательским интерфейсом (GUI) используется сообщество SQLyog 13.1.1 - 32 бит. Облако Amazon AWS подключается к платформе с помощью файла учетных данных, который включает идентификатор aws_access_key_id и ключ aws_secret_access_key, т.е. идентификатор доступа и ключ безопасности для доступа к облаку AWS. Производительность платформы измеряется на компьютерной системе core i3, поддерживающей среду NetBeans с операционной системой Windows 10. Входные файлы в фреймворк берутся по указанному пользователем пути. Входные файлы имеют неоднородный тип, включая файлы .doc, файлы .ppt, файлы .txt, файлы изображений и видео. Фреймворк принимает входные файлы в диапазоне 01, 05, 10, 15, 20, 25, 30. Платформа кластеризует эти файлы, выделяет ресурсы и выполняет шифрование и дешифрование данных пользователей.

Этапы исполнения

Пошаговая процедура выполнения системного сценария показана на рисунке. На следующих нескольких рисунках показаны фактические рабочие этапы и процедура структуры управления транзакциями. Перед запуском фреймворка запускается первая база данных MySQL с помощью сообщества SQLyog, которое является графическим инструментом для базы данных MySQL СУБД. В MySQL создается база данных "транзакций", представляющая собой набор таблиц, таких как таблица регистрации, таблица шифрования и т.д. В то же время мы запустили Amazon AWS Cloud, используя имя пользователя и пароль. Фреймворк использует следующие сервисы Amazon AWS cloud.

1) EC2 (Эластичное вычислительное облако): Эта служба предоставляет ресурсы, т. е. виртуальные машины, требуемые платформой. Динамически платформа выделяет ресурсы (виртуальную машину) пользователю. Платформа дважды запрашивает виртуальные машины, чтобы EC2 выделил две виртуальные машины.

2) DynamoDB: Эта услуга предоставляется Amazon AWS cloud для создания базы данных. Это облачная база данных, используемая платформой. Это набор таблиц, таких как таблица регистрации пользователей, таблица хранилища ключей, таблица файлов и т.д.

3) S3 (Простая служба хранения): Эта услуга предоставляется Amazon AWS cloud для хранения. Огромное количество данных может храниться в облачных системах хранения, таких как S3.In при этом для пользователей платформы создается специальный буфер с именем "Транзакция2122". В этом буфере для каждого пользователя выделяется отдельное пространство. Имя папки указано в идентификаторе электронной почты каждого пользователя.

Оценка эффективности

Платформа реализована в Netbeans 8.2. Платформа использует

различные сервисы, предоставляемые облаком Amazon AWS. Платформа использует другую систему с другой конфигурацией для репликации. Другая система состоит из базы данных "MongoDB", которая используется платформой для хранения реплицированных значений. "MongoDB’ - это база данных NoSQL. Платформа показывает производительность в графическом виде после нажатия пользователем кнопки "Производительность". Все сгенерированные значения отображаются фреймворком в графическом виде. Производительность фреймворка оценивается с помощью нескольких параметров.

Параметры следующие:

1) пропускная способность;

2) скорость передачи данных;

3) время для шифрования;

4) время, необходимое для шифрования и без шифрования;

5) загрузка процессора;

6) время для загрузки

7) время задержки.

Проанализируем некоторые из этих параметров.

1) Пропускная способность: Это показатель того, сколько времени требуется для загрузки среднего количества файлов в облачные системы хранения.

Он рассчитывается как:

Average value = Total File Size/ Number of Files (3.3)

2) Загрузка ЦП: Это показатель времени, который показывает, какой процент ЦП используется платформой при загрузке файла. Его можно рассчитать как:

CPU Utilization= Usages ( ). cpuuti; (3.4)

Переменная cpuuti используется для определения нагрузки на процессор.

3) Время задержки: Это общее время, затраченное платформой с начала процесса загрузки до конца процесса загрузки. Метод с именем current Time Millis () используется для вычисления времени задержки. Он рассчитывается как:

Delay Time = (End Time – Start Time) / 1000 (3.5)

4) Скорость передачи данных: График скорости передачи данных показывает, сколько файловых данных в миллисекунду загружено в облачные системы хранения. Он рассчитывается как:

FSize = File Size / 1000 (3.6)

Datarate = FSize / Delay Time (3.7)

5) Время загрузки: Время загрузки - это время, необходимое для загрузки файла. Значение рассчитывается в миллисекундах. Метод вызывает current Time Millis ( ), используемое для определения требуемого времени. Его можно рассчитать как:

Uploading Time = (Stop upload – Start upload) / 1000 (3.8)

6) Время шифрования: Фреймворк использует его для шифрования одного файла. Значение рассчитывается в миллисекундах. Время шифрования рассчитывается как:

Encryption Time = [(EncEnd EncStart)] / 1000 (3.9)

3.2.3. Методология эксперимента

Платформа полностью реализована в среде Java Netbeans IDE. Эксперимент проводится на системе с машиной core i3 на платформе Windows 10. Подключение к облаку Amazon AWS осуществляется с помощью идентификатора пользователя и пароля. Затем различные службы, такие как EC2, DynamoDB, S3, будут использоваться платформой для использования и хранения необходимой информации в соответствующем месте. В то же время база данных MySQL используется с помощью программного обеспечения сообщества SQLyog. Сообщество

SQLyog содержит необходимую информацию о фреймворке. Это набор таблиц, который содержит важную информацию о структуре, такую как регистрационные данные пользователя, сгенерированные ключи для файлов, файлы в зашифрованных форматах и т.д. Входные файлы, предоставляемые фреймворку, различаются по диапазону 01, 05, 10, 15, 20, 25 и 30. В соответствии с этим кластеризация выполняется при вводе пользователем, и кластер создается с использованием типа файла, размера файла. Эти кластеры выделяются виртуальным машинам для загрузки. По запросу посредством кодирования, т. е. динамически сервис Amazon AWS EC2 предоставляет две виртуальные машины, необходимые для платформы для хранения файлов в облаке Amazon S3. База данных с именем "DynamoDB", которая присутствует в облаке, содержит ряд таблиц для хранения важной информации о файлах. Он содержит такую информацию, как загруженные файлы, зашифрованные файлы, различные ключи разных файлов, информацию об удаленных файлах в различных таблицах. Служба хранения Amazon S3 cloud используется для создания отдельного буфера для платформы. Каждому пользователю платформы выделяется отдельное пространство для хранения их файлов. Платформа принимает количество файлов от пользователя в качестве входных данных, затем шифрует файлы и хранит их в хранилище AWS S3 в соответствующей папке или пространстве пользователя. Администратор с именем "PKG" создается платформой, которая идентифицирует аутентифицированных пользователей в сети. Только после разрешения администратора каждый пользователь может работать с файлами в сети, и пользователь может загрузить файл, обновить файл или удалить файл. Пользователь не может удалить файл непосредственно в сети. Запрос на удаление сначала отправляется Администратору. Администратор проверяет подлинность пользователя, а затем пользователю предоставляется разрешение на удаление. Производительность платформы

проверяется с помощью таких параметров, как пропускная способность, загрузка процессора, Энергоэффективность, Время планирования и время отклика. Структура сравнивается с Методом планирования на основе контейнеров (CBS), методом динамического распределения ресурсов для энергосбережения (DPRA) и методом энергоэффективного управления ресурсами с учетом SLA для облачных сред (SLA).

3.2.4. Метод расчета энергоэффективности

В работе операция шифрования выполняется с файлами, указанными пользователем, при загрузке этих файлов в облачные системы хранения. Платформа шифрует файлы в качестве параметров загрузки, выбранных пользователем, таких как 01, 05, 10, 15, 20, 25 или 30 файлов. Необходимо зашифровать эти файлы, чтобы защитить данные от несанкционированного доступа.

Фреймворк отмечает количество запусков для разного количества файлов и время автономной работы. Разница в оставшемся времени автономной работы отмечается при загрузке файлов. Изменение времени автономной работы, деленное на количество файлов, дает время автономной работы, потребляемое в процентах.

Рассмотрим следующую таблицу, в которой содержится количество файлов, зашифрованных перед загрузкой в облачные системы хранения, и процент заряда батареи, оставшегося после выполнения операции шифрования. Производительность батареи измеряется, когда процент заряда батареи составляет 99%.

Как видно из табл. 3.3, для одного файла для шифрования доля заряда батареи уменьшена с 99 до 98%. Таким образом, потребление энергии для файла составляет 1%. Аналогично, для файлов 05 емкость уменьшена с 98 до 96%, поэтому коэффициент энергопотребления составляет 2%. Таким образом, рассчитывается энергоэффективность,

которая указывает процент батареи, используемой для работы шифрования.

Таблица 3.3

Примерная таблица для расчета энергоэффективности

Файлы	Остаток заряда батареи (%)
01	98
05	96
10	94
15	92
20	90
25	88
30	86

3.2.5. Оцениваемые параметры

Платформа имеет дело с разнородной рабочей нагрузкой путем кластеризации рабочей нагрузки, распределения ресурсов по кластерам и обеспечения безопасности пользовательских данных. Платформа сравнивается с планированием на основе контейнеров (CBS), Распределением ресурсов динамической подготовки (DPRA) и Энергоэффективным управлением ресурсами с поддержкой SLA для облачных сред (SLA). Основными параметрами, рассматриваемыми для оценки эффективности системы, являются следующие:

1) пропускная способность;

2) загрузка процессора;

3) энергоэффективность;

4) планирование времени;

5) время отклика.

3.2.6. Результаты эксперимента

Пропускная способность

Результаты после выполнения файлов в диапазоне, начиная с входных файлов 01, 05, 15, 20, 25, 30 собраны в табличной форме для CBS, DPRA, SLA и предлагаемого подхода. Входных файлов для фреймворка могут быть от 1 до 30. Платформа может загружать 30 файлов с шифрованием в облачную систему хранения. В табл. 3.4 и на рис. 3.5 показаны значения пропускной способности для CBS, DPRA, SLA и предлагаемого авторского подхода соответственно.

Таблица 3.4

Пропускная способность

Файлы	CBS	DPRA	SLA	Автор
1	34.6	37.6	38.2	44.9
5	35.6	37.2	28.2	42.9
10	33.8	35.5	31.2	49.6
15	33.1	34.2	25.8	36.6
20	33.5	36.2	25.7	37.9
25	36.7	38.6	33.2	40.4
30	43.6	45.8	38.3	54.6

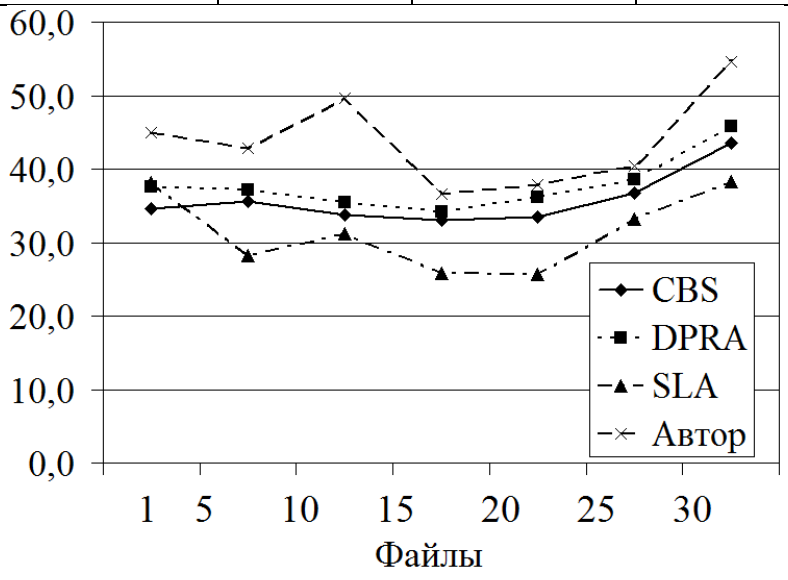

Рис. 3.5. Пропускная способность

Загрузка процессора

Это показатель того, какая доля процессорной мощности используется для загрузки файлов в облачные системы хранения. В табл. 3.5 показаны значения загрузки ЦП для CBS, DPRA, SLA и предлагаемого подхода соответственно.

Таблица 3.5

Загрузка ЦП

Файлы	CBS	DPRA	SLA	Автор
1	75.5	60.6	84.5	44.5
5	60.5	75.5	84.5	56.5
10	70.3	55.7	68.5	50.2
15	73.3	63.6	82.4	52.3
20	69.2	74.1	58.2	52.2
25	64.1	61.0	73.0	50.1
30	75.3	47.0	75.4	45.4

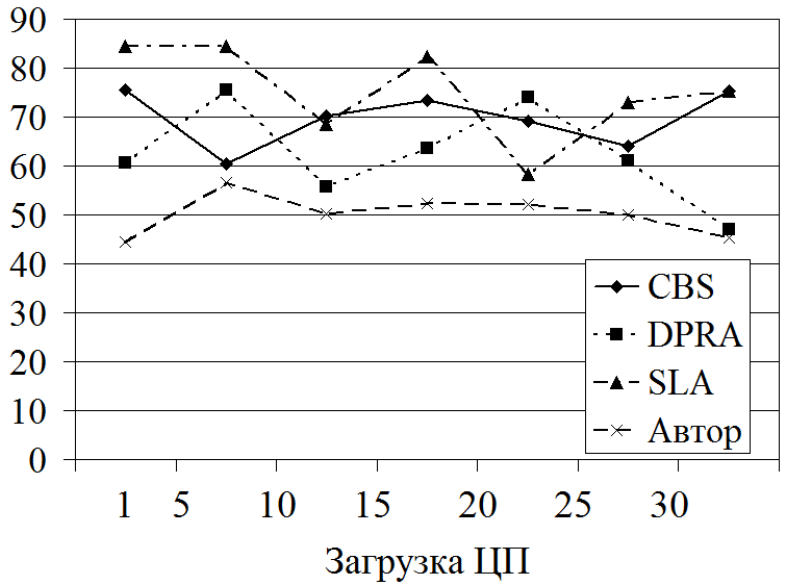

Рис. 3.6. Загрузка процессора

Энергоэффективность

Энергоэффективность рассчитывается для процесса шифрования, т.е.

сколько процентов заряда батареи используется для шифрования файлов с помощью разных методов. Как видно из табл. 3.6 и рис. 3.7, значение энергоэффективности для предлагаемого подхода меньше, чем у методов CBS, DPRA и SLA соответственно.

Таблица 3.6

Энергоэффективность

Файлы	CBS	DPRA	SLA	Автор
1	1.0	1.2	1.5	0.5
5	1.2	1.8	2.0	1.0
10	2.0	2.7	2.8	1.5
15	2.8	3.0	3.5	2.5
20	3.2	3.3	3.8	3.0
25	3.6	3.5	4.0	3.1
30	3.7	3.8	4.8	3.2

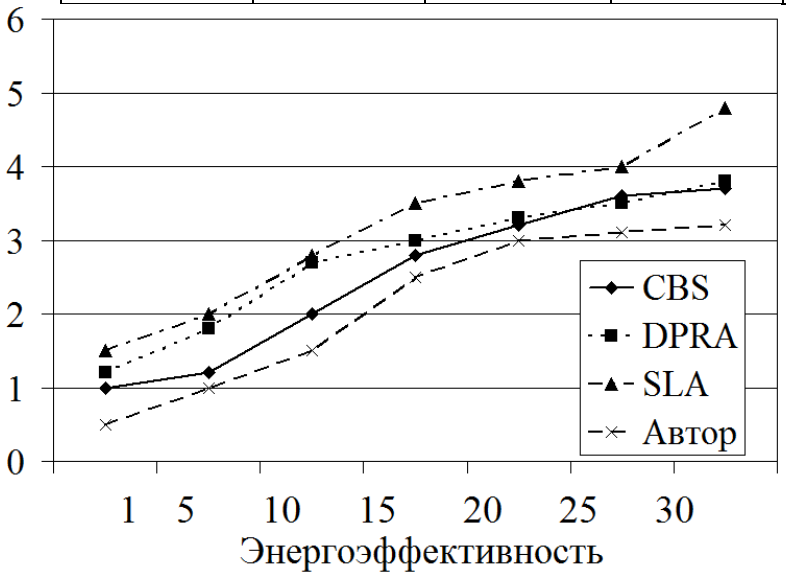

Рис. 3.7. Энергоэффективность

Время планирования

Время планирования - это время, в течение которого файл должен ждать, чтобы запланировать шифрование. Оно рассчитывается в

миллисекундах. В табл. 3.7 показаны значения времени планирования для CBS, DPRA, SLA и предлагаемого подхода соответственно.

Таблица 3.7

Время планирования

Файлы	CBS	DPRA	SLA	Автор
1	27.9	17.6	36.3	6.2
5	47.4	37.1	56.2	35.5
10	74.7	65.1	83.3	53.5
15	108.1	98.2	117.3	87.3
20	140.2	130.3	148.4	118.8
25	166.7	158.2	176.1	145.6
30	200.3	190.1	209.5	180.2

Как видно из приведенных в табл. 3.7 и на рис. 3.8 значений, предлагаемый подход требует меньше времени, чем сравниваемые системы, т.е. CBS, DPRA и SLA.

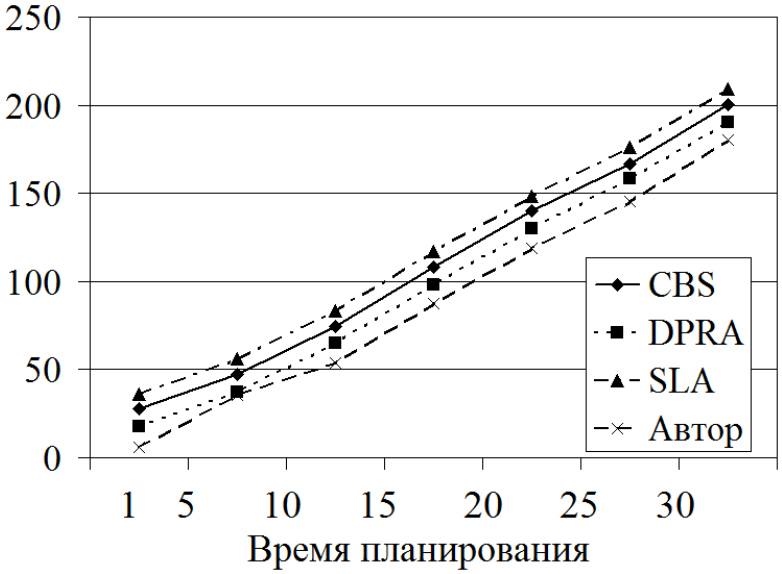

Рис. 3.8. Время планирования

Время отклика

Время отклика - это общее время, затраченное платформой на выполнение, рассчитывается в секундах. В табл. 3.8 показано время отклика для CBS, DPRA, SLA и предлагаемого подхода.

Таблица 3.8

Время отклика

Файлы	CBS	DPRA	SLA	Автор
1	179.8	169.8	189.3	154.7
5	175.4	150.1	165.4	115.3
10	300.4	300.7	285.3	245.4
15	237.5	257.6	297.5	222.4
20	328.6	338.5	299.7	279.5
25	319.1	399.6	320.1	296.1
30	429.6	403.3	441.3	383.6

Значения в табл. 3.8 указывают на то, что предлагаемый подход требует меньше времени, чем другие существующие подходы, т.е. CBS, DPRA и SLA. Соответствующий график представлен на рис. 3.9.

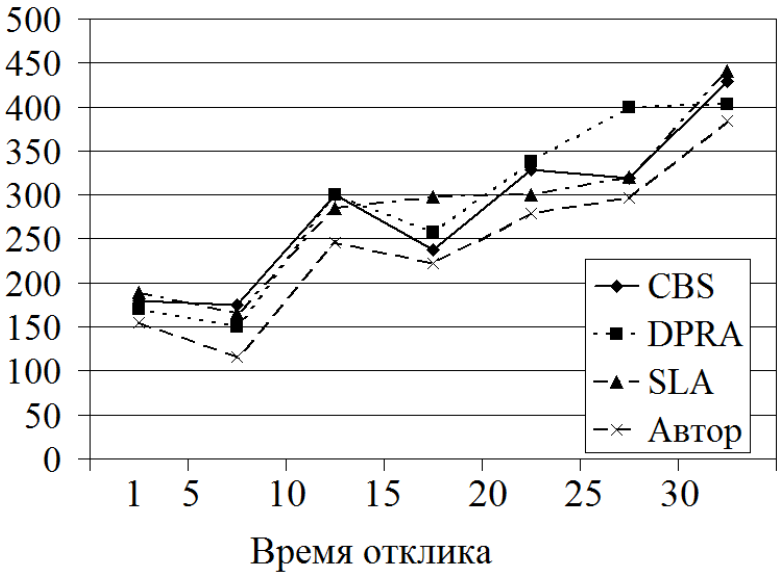

Рис. 3.9. Время отклика

3.3. Выводы

1. Предложена модифицированная политика планирования для улучшения качества обслуживания, предоставляемого пользователям. Модифицированная политика планирования объединена с FCSA, разработанной для управления QoS в СУБД РВ. Предложенный подход в основном основан на интеграции пользователя в процесс планирования, включая настройку предпочтений пользователя (критичность запросов). Эксперименты подтвердили вклад предложенного подхода в максимальное удовлетворение критических транзакций для пользователей. Подход способствовал соблюдению сроков выполнения этих транзакций, минимизируя время ожидания пользователей, учитывая даты их прибытия в систему.

2. В дальнейших исследованиях планируется уточнить параметры, учитываемые в процессе планирования, путем интеграции дополнительного параметра, который является связующим звеном агрегирования между транзакциями. Действительно, некоторые транзакции имеют общие субтранзакции. Приоритизация этих субтранзакций способствует повышению показателей успешности транзакций. Во-вторых, планируется реализовать протокол MRTS для пространственно-временных баз данных [3.17] в сочетании с циклом обратной связи. Наконец, было бы интересно ввести понятие онтологии при анализе запросов пользователей [3.18, 3.19]. Это позволит учесть семантику запросов для лучшего планирования последовательности их выполнения.

3. В работе транзакции рассматриваются в виде гетерогенных файлов от пользователей в качестве входных данных и эффективно развертываются в облачной системе хранения данных и в гетерогенных системах баз данных в качестве выходных данных. Транзакционные операции, такие как вставка, обновление, удаление, выполняются с

данными пользователей в облачных системах хранения и в гетерогенных системах баз данных. Операции транзакционных запросов выполняются в системах баз данных. Предлагаемая система пытается представить новый подход к более безопасной, энергоэффективной, масштабируемой системе управления транзакциями, основанной на терминологии и методах планирования ресурсов в гетерогенных распределенных системах баз данных.

4. Параметры оценки используются для проверки работы фреймворка с существующими системами. С точки зрения пропускной способности предлагаемый подход показывает чуть более высокие значения, чем существующие системы. Но по другим параметрам предлагаемый подход показывает явно лучшие результаты, чем существующие системы. Для оценки таких параметров, как загрузка процессора, энергоэффективность, время планирования, время отклика, предлагаемый подход намного лучше, чем в существующих системах. В табл. 3.4 – 3.8 показаны значения, полученные с помощью структуры, которые указывают на лучшую производительность предлагаемого подхода.

5. Представлен модифицированный протокол планирования транзакций в СУБД реального времени, учитывающий не только временные ограничения транзакций, но и критерии, установленные пользователями базы данных, обеспечивающий корректное определение приоритетов выполняемых транзакций.

6. Разработана архитектура СУБД реального времени, отличающаяся применением модифицированного протокола планирования транзакций и улучшающая качество предоставления услуг пользователям при реализации управления с обратной связью.

Список источников к главе 3

3.1. Aranha R.F.M., Ganti V., Narayanan S., Muthukrishnan C.R., Prasad S.T.S., Ramamritham K. Implementation of a real-time database system// Inf. Syst. 1996, 21, 55–74.

3.2. Amirijoo M., Hansson J., Son S.H. Specification and management of QoS in real-time databases supporting imprecise computations// IEEE Trans. Comput. 2006, 55, 304–319.

3.3. Lu C., Stankovic J.A., Son S.H., Tao G. Feedback control real-time scheduling: framework, modeling, and algorithms// Real-Time Syst. 2002, 23, 85–126.

3.4. Katet M., Bouazizi E., Sadeg B. Prise en Compte des Donnrees Drerivrees Temps Rreel dans Une Architecture de Contrˆole par Rretroaction// Revue rElectronique des Technologies de lInformation (e-TI), 2007, vol. 11.

3.5. Zeddini B., Duvallet C., Sadeg B. Une Approche Qualitre de Service dans Les Syst`emes Multimredias Distribures// Revue rElectronique des Technologies de l'Information (e-TI), 2007, vol. 4.

3.6. Hamdi S., Bouazizi E., Faiz S. QoS management in real-time spatial big data using feedback control scheduling// ISPRS Ann. Photogrammetry Remote Sens. Spat. Inf. Sci. 2015, II–3/W5, 243–248.

3.7. Xiong M., Ramamritham K. Deriving deadlines and periods for real-time update transactions// IEEE Trans. Comput. 2004, 53, 567–583.

3.8. Kang K.-D., Son S.H., Stankovic J.A., Abdelzaher T.F. A QoS-sensitive approach for timeliness and freshness guarantees in real-time databases// 14th Euromicro Conf. on Real-Time Systems. IEEE Computer Society, Vienna, Austria, 2002, pp. 203–212.

3.9. Kang W., Son S.H., Stankovic J.A. DRACON: QoS management for large-scale distributed real-time databases// JSW 2009, 4, 747–757.

3.10. Woochul K., Jaeyong C. QoS management for embedded databases

in multicorebased embedded systems// Mob. Inf. Syst. 2015, 14.

3.11. Cottet F., Delacroix J., Kaiser C., Mammeri Z. Ordonnancement Temps Rreel// Herm`es, 2000.

3.12. Liu C.L., Layland J.W. Scheduling algorithms for multiprogramming in a hard real-time environment// J. ACM 1973, 20, 46–61.

3.13. Buttazzo G.C. Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications. Real-Time Systems Series. Springer, New York, 2004.

3.14. Semghouni S., Amanton L., Sadeg B., Berred A. On new scheduling policy for the improvement of firm RTDBSs performances// Data Knowl. Eng. 2007, 63, 414–432.

3.15. Song H., Kam-Yiu L., Jiantao W., Sang H.S., Aloysius K.M. Adaptive coscheduling for periodic application and update transactions in real-time database systems// J. Syst. Softw. 2012, 85, 1729–1743.

3.16. Bernstein P.A., Hadzilacos V., Goodman N. Concurrency Control and Recovery in Database Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, 1987.

3.17. Jaziri W., Sassi N., Damak D. Using temporal versioning and integrity constraints for updating geographic databases and maintaining their consistency// J. Database Manag. 2015, 26, 30–59.

3.18. Jaziri W., Gargouri F. Ontology theory, management and design: an overview and future directions// Ontology Theory, Management and Design: Advanced Tools and Models, 2010, pp. 27–77.

3.19. Sassi N., Brahmia B., Jaziri W., Bouaziz R. From temporal databases to ontology versioning: an approach for ontology evolution// Ontology Theory, Management and Design: Advanced Tools and Models, 2010, pp. 225–246.

3.20. Hayes B. Cloud computing// Communication ACM, vol. 51, no. 7, p. 9, 2008.

3.21. Tiwari R.G., Navathe SB., Kulkarni G.J. Towards Transactional Data Management over the Cloud// 2010 Second Int. Symp. Data, Privacy, ECommerce, 2010, pp. 100–106.

3.22. Zhang Q., Zhani M.F., Boutaba R., Hellerstein J.L. Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud// IEEE transactions on cloud computing, vol. 2, no. 1, 2014.

3.23. Palanisamy B., Singh A., Liu L. Cost-effective Resource Provisioning for MapReduce in a Cloud// IEEE Transactions on Parallel and Distributed Systems, 2014.

3.24. Minas L., Ellison B Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers// Intel Press, 2009.

4. Программная реализация механизмов управления динамическими СУБД

4.1. Программный комплекс для проверки динамических связей технологических схем и баз данных

4.1.1. Задача проверки динамических связей между графическими объектами интерактивных схем технологического процесса и соответствующих переменненных баз данных

Безопасность эксплуатации потенциально опасных производств является первостепенной целью в промышленности. При управлении такими объектами одной из причин возникновения нарушений являются ошибочные действия оперативного персонала. Одним из инструментов решения данной проблемы является программная проверка результатов человеческой деятельности. Для реализации такой задачи на Нововоронежской АЭС был разработан программный комплекс (ПК), предназначенный для проверки динамических связей между графическими объектами интерактивных схем технологического процесса и соответствующих переменненных баз данных. В качестве методики была использована система кодирования Kraftwerk Kennzeichen System (KKS), позволяющая получать идентификаторы для типовых объектов энергоблока. Результатом работы ПК является информация о кодах KKS, которые не соответствуют базам данных. Организация предлагаемой базы данных реального времени сходна с организацией реляционной базы данных и позволяет обеспечить более 10 000 000 одиночных доступов в секунду. При этом имеется возможность получить файловое представление таблиц в виде дампов. При создании и редактировании видеокадров используются переменные базы данных: каждый динамический объект видеокадра связан со значением переменной. Программное обеспечение

для проверки динамических связей технологических схем и базы данных было реализовано на языке Python. В результате программы формируется список кодов видеокадра, который далее сравнивается со списком кодов базы данных. При обнаружении несоответствия кода динамической связи видеокадра коду базы данных выводится сообщение об ошибке. Внедрение разработанного программного продукта на Нововоронежской АЭС была повышена достоверность представления данных оператору от различных систем энергоблока. Получены новые возможности информационной поддержки оператора, облегчающие оценку состояния оборудования и принятия решения по управлению блоком. Предлагаемая методика использования кодов KKS для проверки динамических связей между графическими объектами интерактивных схем технологического процесса и соответствующих перемененных баз данных уменьшает нагрузку на оператора ядерного блока и снижает вероятность его ошибочных действий.

4.1.2. Характеристика системы управления технологическими процессами энергоблока

Персонал блочного пункта управления осуществляет мониторинг и управление энергоблоком с автоматизированных рабочих мест (АРМ), на которых отображаются видеокадры. Видеокадры являются интерактивными схемами технологического процесса. На них присутствуют как статические элементы без возможности управления, так и динамические, воздействие на которые приводит к изменению состояния оборудования. Такой подход к управлению энергоблоком реализован в рамках *системы верхнего блочного уровня* (СВБУ). СВБУ является подсистемой *автоматизированной системы управления технологического процесса* (АСУ ТП) [4.1-4.4].

Основными функциями СВБУ являются обеспечение:

- контроля и управления технологическим процессом;

- интеграция всей информации по энергоблоку от всех систем и подсистем АСУ ТП;

- дисплейного управления оборудованием систем нормальной эксплуатации и оборудованием систем безопасности в режимах, предусмотренных проектом;

- централизованного контроля и представления, как обобщенной, так и детализированной информации о состоянии энергоблока, отдельных параметрах технологического процесса и состоянии оборудования;

- контроля состояния барьеров безопасности энергоблока;

- необходимой информацией персонала различных подразделений АЭС, которым эта информация необходима в процессе работы;

- формирования сигнализации о нарушениях в работе энергоблока, отдельных систем, отдельного оборудования;

- необходимой информацией аварийного центра АЭС;

- возможности обмена информацией с другими подсистемами АСУ ТП [4.5].

Реализация возможности дисплейного управления обусловлена в том числе большим количеством оборудования и точек контроля технологического процесса. Для обеспечения функционирования СВБУ база данных АЭС содержит более 170000 переменных [4.6].

Пакет программ СВБУ называется Портал. Он применяется для сбора, хранения, обработки и предоставления информации операторам. Пакет программ обеспечивает возможность контроля и управления оборудованием энергоблока.

Программное обеспечение Портал состоит из нескольких компонентов (рис. 4.1):

- исполняющая система;

- система визуализации и регистрации технологических данных;

- система конфигурирования технологических данных [4.7].

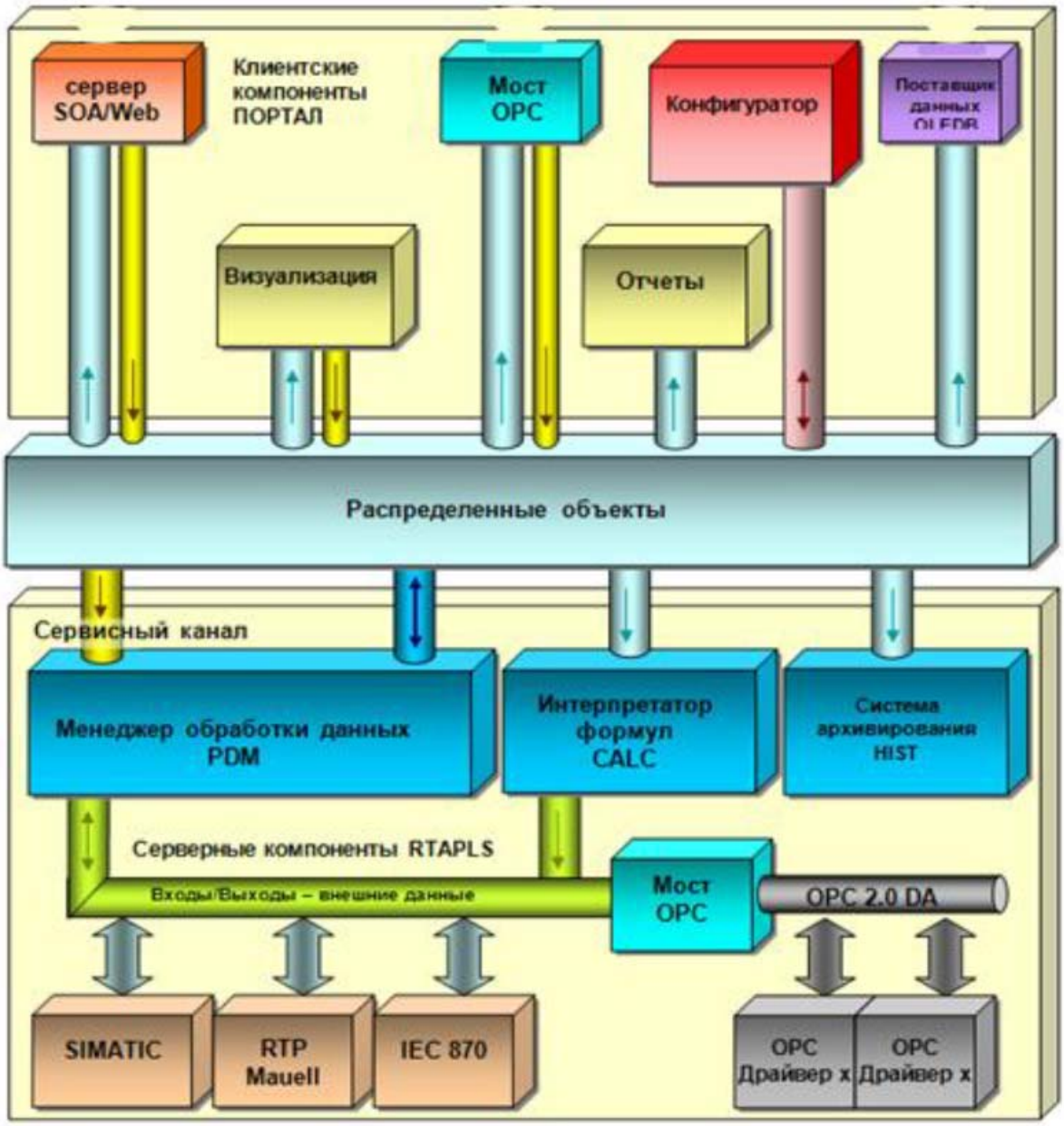

Рис. 4.1. Структура информационных потоков системы Портал

Исполняющая система включает в себя следующие компоненты:

- ядро (RTA), обеспечивающее работу в режиме реального времени;
- модель распределенных объектов (DBO);
- менеджер обработки данных технологического процесса (PDM);
- интерпретатор формул (CALC);
- система архивирования (HIST);
- информационно-отчетная система (REPORT);

- визуализация технологического процесса (VIEW);

- объектно-ориентированная система проектирования (RtOOS);

- драйверы (интерфейсы к системам ввода-вывода).

Вся система базируется на ядре, именуемом Архитектурой Реального Времени (RTA). Ядро предоставляет набор базовых служб. Технологии, используемые архитектурой реального времени:

- база данных в реальном времени (RtDb);

- взаимодействие процессов в реальном времени (RtIpC) – осуществляет коммуникацию данных между процессами, репликацию данных пользователям;

- управление резервированием в реальном времени.

4.1.3. База данных и переменных системы

Основными информационными единицами, в разрабатываемой системе, являются технологические данные, которые хранятся в базе данных реального времени [4.8].

Организация базы данных реального времени сходна с организацией реляционной базы данных. Она представляет собой совокупность связанных между собой таблиц (рис. 4.2).

Информационные единицы представлены в виде двоичных, аналоговых и текстовых технологических переменных. Переменные в одной таблице могут иметь несколько столбцов, которые описывают их конфигурацию. Переменные имеют уникальный идентификатор. Идентификатор обеспечивает взаимосвязь с другими таблицами. Способы взаимодействия между таблицами базы данных «one to one», «one to many». Например, в случае таблиц PLS_ANA_CONF и PLS_ANA_DYN уникальный идентификатор PVNr, который используется только в одной строке таблицы PLS_ANA_CONF («one to one»). В случае таблиц PLS_ANA_DYN и PLS_USERS уникальный идентификатор USERNR,

который используется в нескольких строках таблицы PLS_USERS («one to many»).

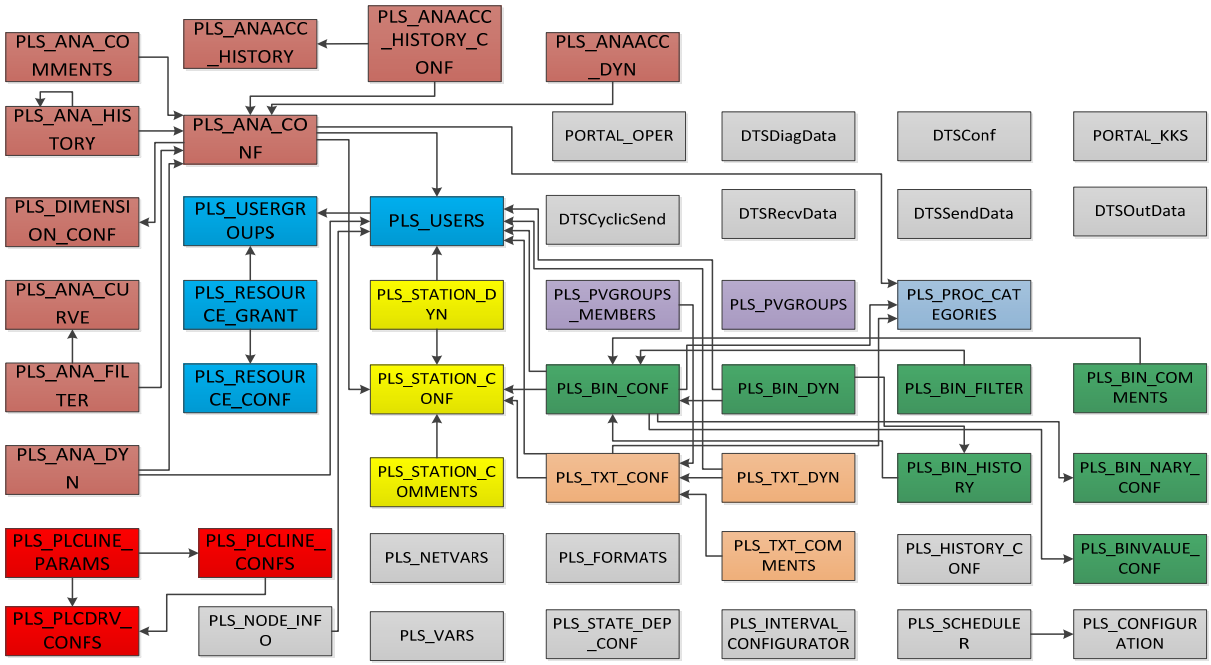

Рис. 4.2. Структура ОЗУ-резидентной базы данных реального времени

Рассматриваемая база данных характеризуется следующими свойствами:

- существует возможность обеспечить в реальном времени более 10 000 000 одиночных доступов в секунду;

- данные отображаются в таблицах со строками и столбцами;

- каждый из индивидуальных сетевых узлов в распределенной системе может хранить локальную копию таблиц базы данных. Репликация таблиц будет произведена автоматически в случае их изменения с помощью широковещательных протоколов;- специальная интеграция в язык программирования C++ путем поддержки типов данных из C++ и через процедурный, программный интерфейс;

- пусковые механизмы позволяют какому-либо приложению определять действия, которые будут активированы автоматически, когда

некое приложение меняет данные в базе данных;

- таблица базы данных может быть распределена на несколько серверов, каждый из которых отвечает за определенную часть данных [4.9].

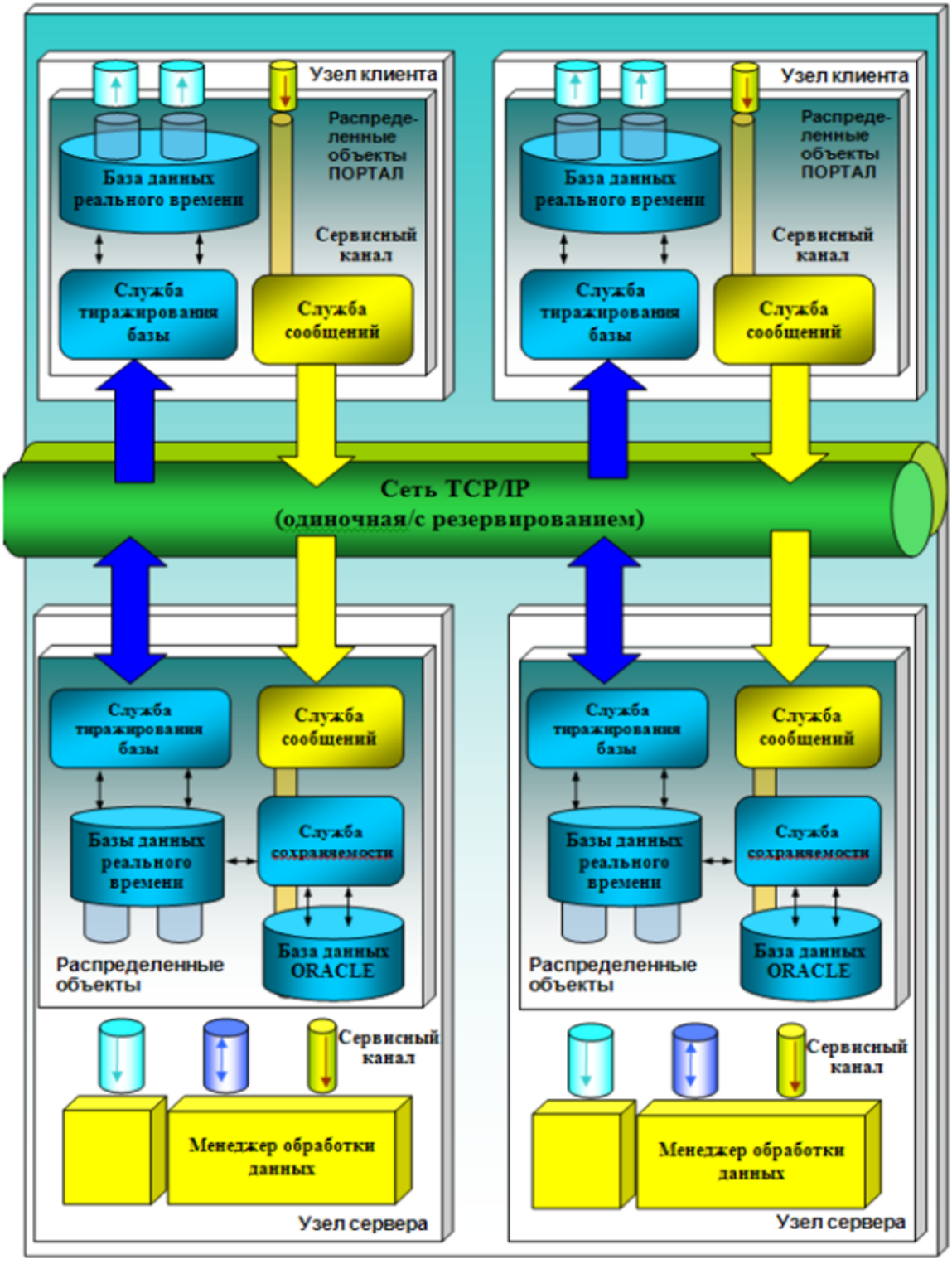

Рис. 4.3. Структурная схема хранения распределенных объектов

Схема хранения распределенных объектов приведена на рис. 4.3.

Таблицы базы данных располагаются в памяти компьютера для ускорения доступа к данным, однако имеется возможность получить файловое представление таблиц в виде дампов [4.10].

Дамп является файлом и представляет собой таблицу базы данных в текстовом формате. Одна строка файла соответствует одной строке таблицы, а значения соседних ячеек таблицы разделены символом «|», закомментированные строки начинаются символом «*». Первая строка файла содержит заголовок таблицы. Следующие строки определяют значения и конфигурации переменных, в соответствие с которыми они обрабатываются системе Портал.

База данных формируется на основании импортных файлов низовой автоматики.

4.1.4. Создание и редактирование интерактивных схем технологического процесса

Переменные в базе данных являются сигналами от оборудования и механизмов. Они несут в себе информацию о соответствующих параметрах технологического процесса. Переменные представлены в дискретном, аналоговом и текстовом видах. Они имеют свою конфигурацию, на основании которой исполняющая система осуществляет их обработку. Переменные имеют следующую классификацию:

- входные переменные, которые отображают состояние датчиков в технологическом процессе;

- выходные переменные, которые выдаются в процесс при каждом их изменении или сразу после перезапуска системы;

- производные переменные, которые вычисляются на основе других переменных;

- системные переменные, которые формируются ПО Портал;

- переменные приложения, которые создаются прикладными приложениями;

- переменные, которые вводятся и модифицируются оператором.

При создании и редактировании видеокадров используются переменные базы данных: каждый динамический объект видеокадра связан со значением переменной. На видеокадрах технологический процесс изображен в виде интерактивных схем, которые создаются в соответствии с нормативными документами. Информация предоставляется оператору в удобном и стандартизированном формате. Для создания, редактирования и отображения видеокадров используется система визуализации ПО Портал. Пример видеокадра представлен на рис. 4.4.

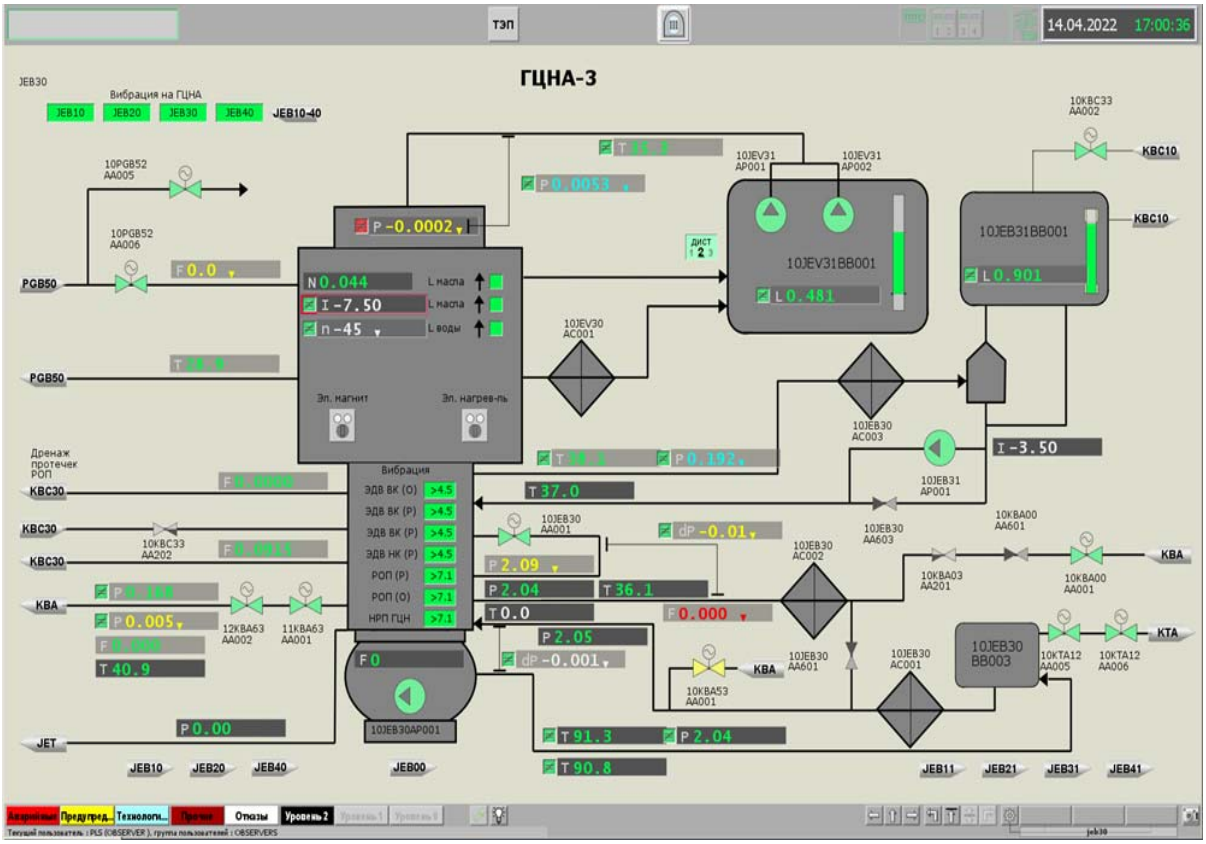

Рис. 4.4. Пример визуализации видеокадра при использовании платформы Портал

Система визуализации использует векторный графический стандарт SVG. Преимущество SVG состоит в том, что доступно множество

редакторов и конвертеров для многократного использования разработанных изображений. Формат SVG независим от платформы. Дополнительно система визуализации имеет встроенный обозреватель для отображения гипертекстового содержимого (HTML).

Система визуализации может содержать программный код, например, для управления навигацией между окнами. Реализация алгоритмов, специфических для прикладной системы, требует использования специализированного языка программирования.

Система визуализации состоит из исполняющей и прикладной части [4.7]. Исполняющая система визуализации запускается на каждой рабочей станции с графическим интерфейсом оператора и необходима для дисплейного управления. Графический интерфейс пользователя состоит из главного и нескольких дополнительных окон. Окно в свою очередь состоит из областей отображения, на которых могут быть представлены HTML-страницы или видеокадры.

Прикладная система используется для создания графических интерфейсов пользователя. К прикладной части относится редактор, который применяется для создания видеокадров, а также логических и арифметических связей и ссылок на источники данных. То есть в редакторе создаются динамические связи между графическими объектами и базой данных. Редактор работает с файлами формата SVG и способен реализовать часть его спецификации:

- объекты (circle – круг, line – линия, title – наименование и т.д.);

- события (OnClick – щелчок левой кнопкой мыши и т.д.);

- атрибуты (id – идентификация элемента, description – описание элемента и т.д.).

Прикладная система обеспечивает возможность создания и редактирования видеокадров в соответствии с требованиями технологического процесса. Работа с графическими схемами происходит в

специализированном редакторе ПО Портал. После создания графической схемы необходимо создать динамическую связь между графическим объектом и соответствующей переменной базы данных. Существует риск создания неправильной динамической связи вследствие большого количества графических объектов и переменных.

Для устранения данной проблемы был разработан программный комплекс, который проверяет корректность созданной динамической связи между графическим объектом технологической схемы и соответствующей переменной базы данных.

4.1.5. Программное обеспечение для проверки динамических связей технологических схем и базы данных

Языком программирования для написания кода программы был выбран Python. Выбор данного языка обосновывается следующим:

- кроссплатформенностью;

- возможностью дальнейшего развития;

- наличием множества библиотек с готовыми решениями для различных задач.

Алгоритм работы программы представлен на рис. 4.5.

В целях подтверждения корректности функционирования разработанного программного средства выполнялась его верификация путем сопоставления выходных данных программы с проверенными данными, где преднамеренно были созданы ошибки. Для этого в общую директорию были загружены дампы PLS_ANA_CONF и PLS_BIN_CONF базы данных с общим количеством 173315 переменных, а также десять видеокадров. В одном из видеокадров умышленно были созданы две динамические связи, коды которых отсутствуют в базе данных. Программа корректно обнаружила ошибки.

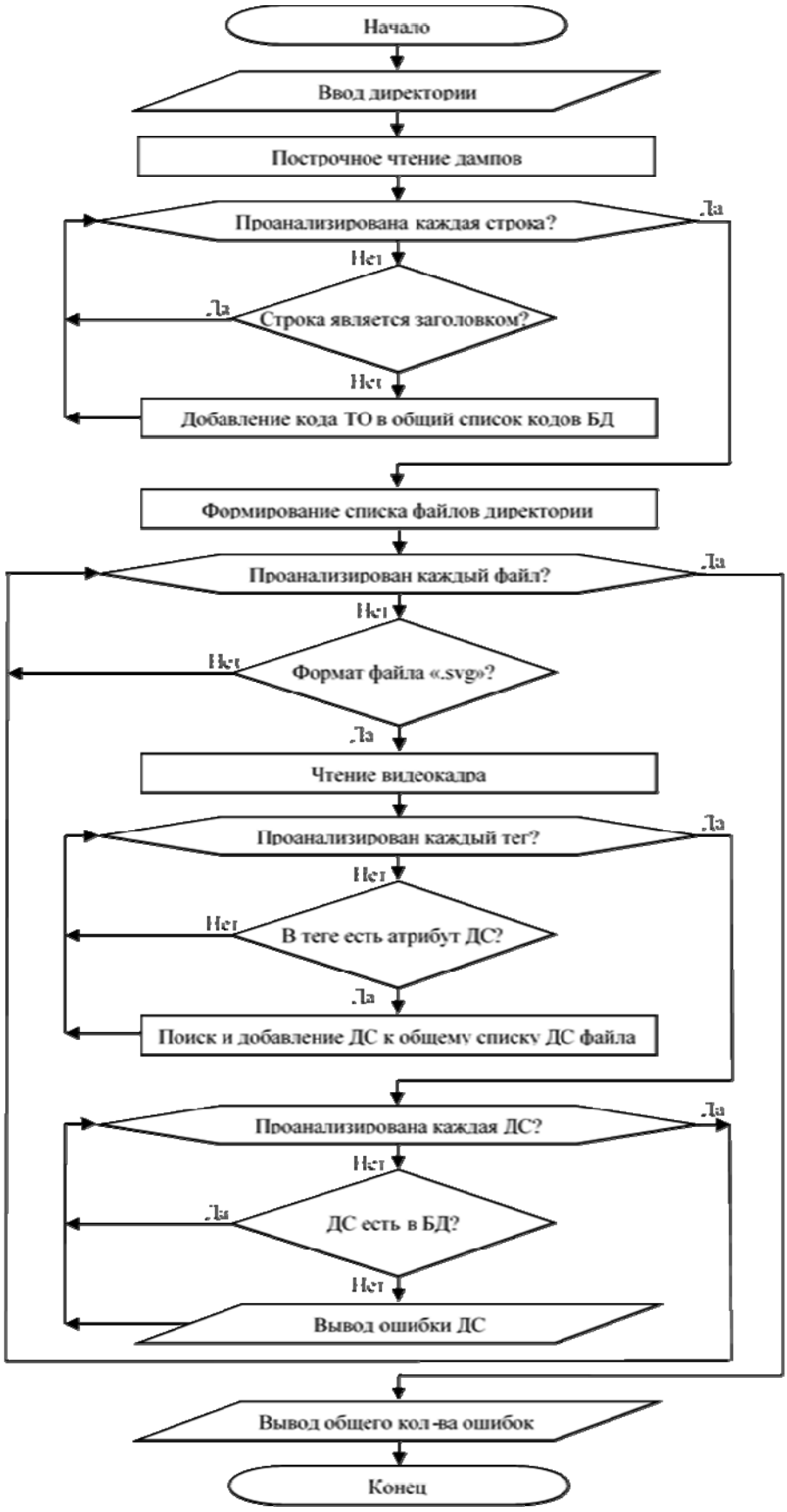

Рис. 4.5. Алгоритм работы программы

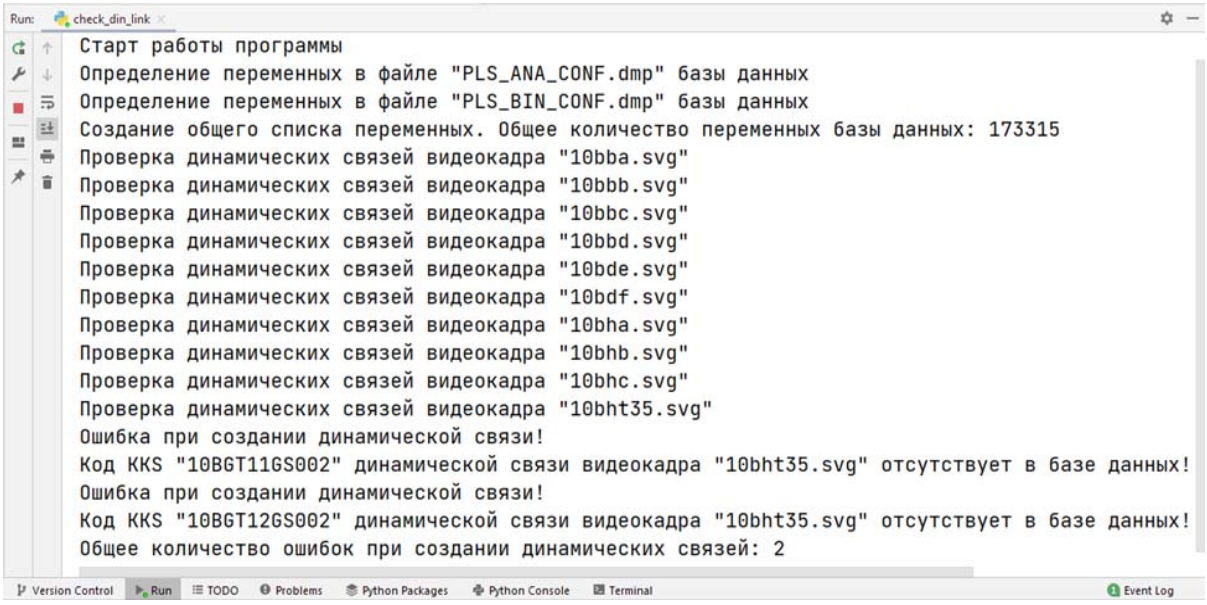

Рис. 4.6. Пример визуализации работы программного комплекса

На выходе программа выводит имена дампов базы данных, общее количество переменных, имена файлов видеокадров, в случае обнаружения ошибок – коды и имена видеокадров с ошибками, общее количество ошибок (рис. 4.6).

Таким образом, создан программный комплекс для проверки динамических связей технологических схем и баз данных, использующий систему кодирования Kraftwerk Kennzeichen System, позволяющий получать идентификаторы для типовых объектов энергоблока и выявляющий идентификаторы, которые не соответствуют базам данных.

4.2. Разработка распределенной информационно-вычислительной специальной системы управления

Ядерные блоки атомных электростанций имеют длительный срок эксплуатации, что приводит к ситуации, когда в процессе эксплуатации

технические и программные средства систем управления перестают отвечать текущим современным требованиям в плане надежности и безопасности их использования. В результате для продления срока действия ядерного блока требуется обязательное проведение модернизации информационно-вычислительной системы (ИВС) управления.

При этом потребовалось создать резервный щит управления (РЩУ), произвести разделение систем питания и систем контроля и управления технологических систем нормальной эксплуатации и систем безопасности. Кроме этого были сохранены традиционные способы управления и контроля: управление с помощью индивидуальных ключей/кнопок, а также индивидуальные приборы и табло сигнализации, установленные на панелях и пультах блочного и резервного щитов управления (БЩУ/РЩУ). Плюс дополнительно был разработан компьютерный способ визуализации параметров на видеокадрах информационных систем.

4.2.1. Характеристика информационно-вычислительной системы ядерного блока до модернизации

На энергоблоке №4 Нововоронежской АЭС эксплуатировалось несколько разрозненных информационных систем, представляющих информацию о состоянии технологического оборудования эксплуатирующему персоналу БЩУ. Например:

- система отображения технологической информации (СОТИ-М) на базе приборов Ш711/2-1;

- многоканальные приборы производства «Электроточприбор» для приема и обработки поочередно, а также выдачи до 4-х релейных каналов сигнализации по каждому параметру;

- программно-технический комплекс системы внутриреакторного контроля на базе аппаратуры СВРК-03-06 производства завода «Тензор»,

предназначенный для приёма и обработки сигналов от совокупности внутриреакторных датчиков, проведения тепло-гидравлических и нейтронно-физических расчётов для определения состояния активной зоны реактора и представления полученных данных оперативному персоналу БЩУ;

- система представления параметров безопасности на базе измерительных шкафов аппаратуры DAS и вычислительного уровня под управлением операционной системы OpenVMS, предназначенная для предоставления информации о состоянии технологических систем и расчёта состояния критических функций безопасности (КФБ) энергоблока.

Структурная схема информационно-вычислительной системы ядерного блока до модернизации представлена на рис. 4.7.

Недостатками построения и функционирования являются следующие моменты:

- системы СВРК и системы предоставления параметров безопасности (СППБ) имеют различные аппаратные и программные решения;

- в каждой системе предусмотрены собственные рабочие станции, работающие под управлением разного прикладного программного обеспечения и по разному реализовывающие подход к отображению информации оператору на видеокадрах, к работе с архивными данными и т.д.;

- СОТИ-М не имеет средств графического представления информации и архивных данных на БЩУ и использует мощности программного обеспечения СППБ;

- подсистемы АСУТП деинтегрированы и полностью автономны, не имеют общего информационного пространства.

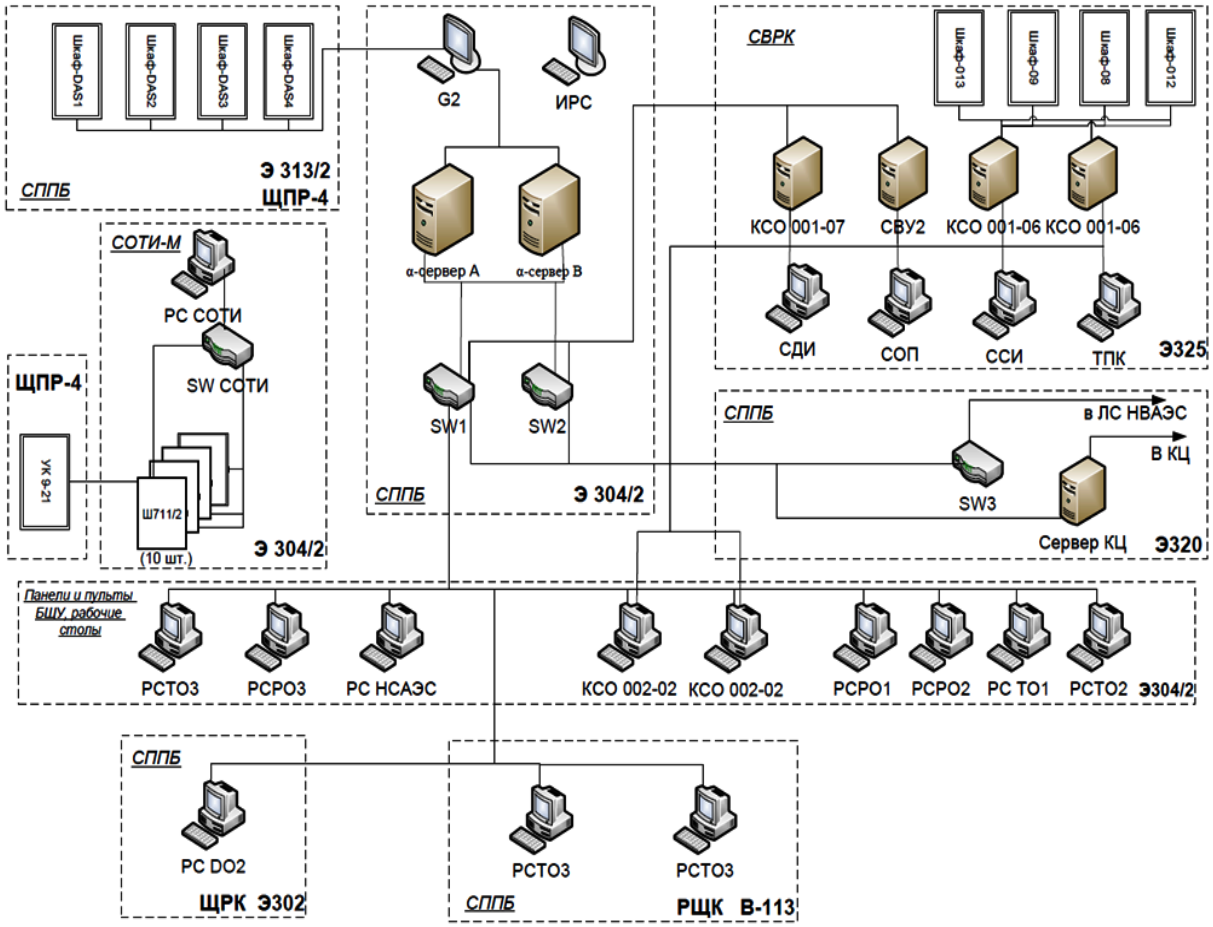

Рис. 4.7. Структурная схема информационно-вычислительной системы ядерного блока до модернизации: РС - рабочая станция, КСО - комплект специального оборудования, РО/ТО – реакторное/турбинное отделение, СВУ – сервер верхнего уровня, СДИ/СОП/ССИ – рабочая станция дежурного инженера/оперативного персонала/системного инженера

К моменту продления срока эксплуатации энергоблока №4 вышеперечисленные системы безнадёжно устарели и не отвечали современным требованиям, предъявляемым к современным средствам АСУ ТП.

4.2.2. Принципы построения внедряемой ИВС

При выборе оборудования для создания новой ИВС

модернизируемого энергоблока был реализован принцип унификации. Программное обеспечение всех компонентов программно-технического комплекса (ПТК) ИВС, включая функции СППБ и СВРК, реализовано на единых программных средствах. Представление значений параметров сигналов на всех рабочих станциях ПТК ИВС, интерфейсы взаимодействия, человеко-машинный интерфейс и навигация по видеокадрам идентичны, что учитывает требования по оптимальному взаимодействию системы «человек-машина». На рис. 4.8 представлена принципиальная схема построения ИВС на энергоблоке №4 после модернизации.

Система удовлетворяет требованиям по обеспечению надёжности на основе резервирования, независимости, разнообразия, с учётом отказов по общей причине.

Для этого ИВС была реализована в виде двухканальной информационной системы. Основной и дублирующий каналы измерения и обработки данных в ПТК ИВС функционируют одновременно в полном объеме. В серверах оперативного контура вычислительного комплекса ИВС, к которым относятся два сервера ИВС и два сервера СВРК, информация, полученная от двух каналов устройств связи с объектом: УСО ИВС и УСО СВРК обрабатывается совместно, что обеспечивает отсутствие влияния отказа в одном из каналов УСО на результаты работы ИВС.

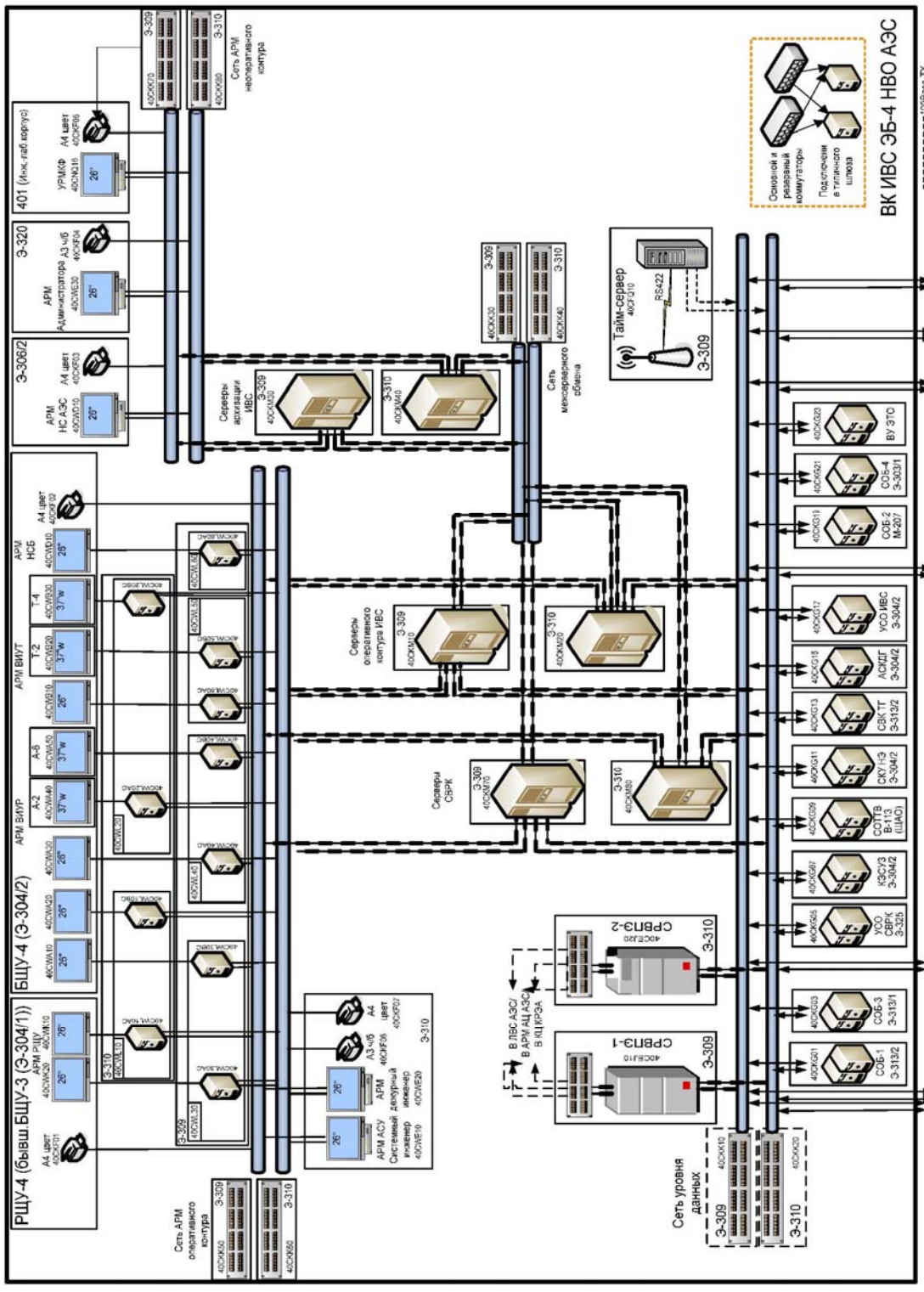

Рис. 4.8. Принципиальная схема информационно-вычислительной системы после модернизации энергоблока №4: СОБ - система обеспечения безопасности, СКУ – система контроля и управления, СВК – система контроля вибрации, АСК – автоматизированная система контроля, КЭСУЗ - комплект электрооборудования систем управления защитой

4.2.3. Функционал разработанной ИВС

Оборудование разработанной ИВС при режимах нормальной эксплуатации и нарушениях нормальной эксплуатации модернизируемого блока реализует следующий информационный функционал:

- прием информации от всех существующих, модернизированных и вновь созданных подсистем систем контроля и управления (СКУ) энергоблока (ЭБ) в соответствии со структурной схемой ИВС (рис. 4.8):

– представление актуальной и ретроспективной информации оперативному персоналу, а также экспертам аварийных центров АЭС и Концерна Росэнергоатом;

– представление оперативному персоналу световой и звуковой сигнализации при выходе параметров за проектные уставки;

– регистрация (архивирование) полученной информации, включая команды операторов и положение механизмов систем безопасности;

– вывод на печать зарегистрированной информации в различной форме;

– обеспечение единым временем оборудования ПТК ИВС и смежных подсистем СКУ ЭБ нижнего уровня автоматизации, от которых вычислительный комплекс ИВС принимает информацию;

– регистрацию информации при аварийных ситуациях (СРВПЭ);

– обеспечение функционирования СВРК и СППБ в объеме не менее чем в эксплуатировавшейся системе до модернизации с учётом изменений в системах безопасности и системах нормальной эксплуатации.

Функционал СВРК по обеспечению персонала достоверной информацией о состоянии основных параметров активной зоны реактора (АкЗ) и теплоносителя первого контура включает в себя:

– обработку результатов измерений;

– расчёт тепловой мощности активной зоны;

– расчёт теплофизических переменных в объёме активной зоны;

– расчёт поля энерговыделения;

– расчёт запасов до кризиса теплообмена;

– расчёт функционалов по объему АкЗ;

– расчёт выгорания и отравления в объёме АкЗ;

– расчёт покассетных и потвэльных распределений энерговыделения;

– расчёт коэффициентов внутрикассетной неравномерности энерговыделения;

– контроль отклонений технологических величин;

– компенсация запаздывания сигналов ДПЗ;

– учет влияния неполного перемешивания теплоносителя в головках кассет на показания термопреобразователя (ТП) на выходах кассет;

– контроль температуры (или энтальпии) теплоносителя на выходе из горячего канала.

Дополнительно к последней функции реализованы следующие опции:

а) компенсация запаздывания сигналов ДПЗ;

б) учет влияния неполного перемешивания теплоносителя в головках кассет на показания ТП на выходах кассет;

в) контроль температуры теплоносителя на выходе из горячего канала.

Функционал СППБ включает:

– своевременное уведомление оператора о нарушении КФБ, отображение сигнализации об изменениях состояния КФБ;

– представление оперативному персоналу электронного аналога деревьев состояния КФБ, определяющих выбор процедуры по

восстановлению КФБ;

– выдача оператору рекомендаций о применении той или иной процедуры по восстановлению КФБ (контроль состояния диагностических блоков (узлов) деревьев КФБ и контроль активации выходных ветвей деревьев КФБ);

– предоставление электронных интерактивных процедур восстановления КФБ;

– предоставление справочной или расчетной информации, необходимой для принятия решений и выполнения управляющих действий, определенных в определенных в «Руководстве по управлению за проектными авариями»;

– предоставление необходимой информации о состоянии рабочих механизмов, используемых при выполнении управляющих действий.

4.2.4. Автоматизированные рабочие места оперативного персонала

Автоматизированные рабочие места (АРМ) оперативного персонала обеспечивают представления всей совокупной технологической информации оперативному персоналу блока и сигнализацию о нарушениях технологического процесса. АРМ получают информацию от каждого сервера ИВС, работают независимо друг от друга и представляют информацию на мониторах в цикле приема данных от серверов.

В состав ИВС входят следующие *автоматизированные рабочие места оперативного персонала*:

– АРМ ведущего инженера управления реактором (ВИУР), состоящий из двух дисплеев, размещенных на панелях, и трех дисплеев, размещенных на столе БЩУ;

– АРМ ведущего инженера управления турбиной (ВИУТ), состоящий из двух дисплеев, размещенных на панелях, и одного дисплея, размещенного на столе БЩУ;

– АРМ начальника смены блока (НСБ), состоящий из одного дисплея, размещенного на столе БЩУ;

– АРМ РЩУ, состоящий из двух дисплеев, размещенных на РЩУ;

– АРМ начальника смены (НС) АЭС, состоящий из дисплея, размещённого на рабочем месте НС АЭС;

– АРМ системного администратора ИВС;

– АРМ системного инженера и АРМ дежурного инженера;

– УРМКФ (рабочее место контролирующего физика).

На мониторы оперативных АРМ выводятся следующие данные:

– технологическая информация (представление видеокадров, сигнализация, протоколирование технологических событий, вывод архивных данных);

– прогноз состояния РУ (с участием оперативного персонала).

На мониторы неоперативных АРМ выводится информация:

– диагностика состояния измерительных каналов;

– контроль и диагностика ПТК ИВС;

– контроль санкций доступа персонала.

Реализация решения использовать однотипные аппаратные и программные средства шлюзового оборудования, входящего в состав ИВС модернизированного энергоблока, позволяет оптимизировать состав резервного оборудования и избежать программных конфликтов при обмене данными «шлюз-сервер». Шлюзовое прокладное программное обеспечение энергоблока предусматривает до 40 протоколов обмена со смежными системами.

На сегодня на данном энергоблоке создана сетевая инфраструктура для подключения до 24 предусмотренных проектом ИВС систем.

4.2.5. Программное обеспечение разработанной ИВС

В качестве базовой платформы для разработанной ИВС был выбран программный комплекс «Круиз». ПО данного комплекса имеет модульную структуру и создается проектным путем (конфигурируется) на базе библиотеки программных средств «Круиз». Сконфигурированное ПО является децентрализованным и может функционировать как на одном компьютере, так и на многомашинном комплексе. По решаемым задачам и степени инвариантности к прикладным задачам ПО было реализовано четыре уровня.

Уровень 1 - операционная система (не является частью ПО "Круиз"). Чаще всего на оборудовании оперативного персонала используется ОС Linux, на оборудовании неоперативного персонала - различные версии Windows.

Уровень 2 - средства абстрагирования от операционной системы. Они, как и уровни 3 и 4, входят в состав ПО «Круиз». Наличие этого уровня позволяет использовать ПО полуфабрикатного режима (ПФР) с разными операционными системами без изменений в прикладных программах: для перехода к другой операционной системе достаточно заменить библиотеку программных средств уровня 2 и перекомпилировать ПО уровней 3 и 4. Программные средства уровня 2 пользовательского интерфейса не имеют и обслуживания не требуют.

Уровень 3 - базовые сервисы. Уровень реализует механизмы взаимодействия между прикладными программами. Сервисы реализуют свои функции путем вызова программ из библиотеки функций ПО "Круиз". В состав ИС входят следующие базовые сервисы:

– сервис запуска ПО обеспечивает управление остальными включенными в конфигурацию системы сервисами;

– сервис хранилища обеспечивает функционирование размещенных на жестком диске каждой ЭВМ ИС файлов (Хранилища

данных), содержащих информацию о конфигурации и настройках системы и необходимые для рестарта накапливаемые данные. Сервис выполняет, в том числе, синхронизацию Хранилищ разных ЭВМ;

– сервис диспетчера обеспечивает организацию взаимодействия и обмена данными между модулями диспетчера, выполняющими функции приема, обработки и передачи данных. Каждый модуль диспетчера может обмениваться данными с диспетчером через специализированный программный интерфейс. Диспетчер хранит перечни входных и выходных данных каждого включенного в конфигурацию модуля, и по приходу от данных от любого модуля раздает заинтересованным в них модулям. Включение в конфигурацию нового модуля может выполняться как при остановленном, так и при работающем диспетчере;

– сервис протокола сообщений обеспечивает функционирование протокола событий;

– сервис архива обеспечивает функционирование аппертурного архива;

– сервис самодиагностики обеспечивает контроль состояния оборудования.

Уровень 4 - прикладные программы и модули, выполненные в виде программных библиотек. Программные средства данного уровня комплексно разработаны по единым требованиям, что исключает проблемы совместимости нижнего и верхнего уровней разработанной платформы.

4.2.6. Обеспечение надежности разработанных ИВС

Надежность реализованных на единой платформе решений обеспечивается следующим:

– оборудование изготовлено серийно по техническим условиям под контролем надзорных органов;

– программные средства разработаны в соответствии изложенными в МЭК 60880 и 62138 требованиями процедуре разработки ПО систем, важных для безопасности;

– организация программной среды обеспечивает возможность резервирования оборудования;

– ряд важных для безопасности функций реализован в нескольких (не менее двух) вариантах в разных программных модулях, обеспечена возможность одновременной работы альтернативных алгоритмов и автоматического сравнения результатов;

– в системах с дублированной структурой для повышения надежности на основе разнообразия реализована мультиплатформенность ПО, позволяющая на основном и резервном оборудовании использовать разные операционные системы;

– прием данных от измерительной аппаратуры и обработка данных выполняются в жестком цикле по принципу "все всегда", что обеспечивает отсутствие зависимости загруженности (работоспособности) системы от режимов работы контролируемого объекта.

В ИВС модернизированного энергоблока используется российская операционная система Astra Linux «Смоленск», сертифицированная в системах сертификации средств защиты информации ФСТЭК России. В роли основного программного обеспечения выступает аттестованное в «Ростехнадзор» программное обеспечение «Круиз» 3.3. Разработчиком программного обеспечения является ООО «ИФ СНИИП АТОМ», что дает возможность дальнейшего всестороннего развития этой программной платформы. Внедрение отечественных программных продуктов хорошо отражает общее направление развития во всех отраслях Российского производства после 2014 г.

Модульная структура построения с возможностью гибкого распределения функций между компьютерами позволяет распределять

нагрузку и определять индивидуальные роли каждого сервера, шлюза и т.д. Клиент-серверная архитектура, используемая в классических решениях, ограничена вычислительными возможностями обработки данных в сервере.

Модульный подход позволяет иметь всего один дистрибутив программного обеспечения, а пакет настраиваемых «профилей» оборудования позволяет оперативно конфигурировать один «системный блок» под любые задачи, от шлюза до сервера. Эти свойства программного обеспечения дают возможность иметь универсальный резерв оборудования при ограниченном количестве ЗИП.

Таким образом, создана распределенная специальная информационно-вычислительная система управления, в которой шлюзовое прокладное программное обеспечение предусматривает до 40 протоколов обмена со смежными модулям, а также позволяет оптимизировать состав резервного оборудования и избежать программных конфликтов при обмене данными.

4.3. Выводы

1. В результате внедрения разработанного программного комплекса в систему управления Нововоронежской атомной электростанции была решена одна из проблем повышения эксплуатационной надежности оборудования и систем безопасности в режимах, предусмотренных проектом. Достигнуты цели по достоверности представления данных оператору от различных систем энергоблока.

2. Расширены возможности представления данных. Получены новые возможности информационной поддержки оператора, облегчающие оценку состояния оборудования и принятия решения по управлению блоком. Повышена надёжность и устойчивость работы системы верхнего

блочного уровня за счет расширения возможностей контроля динамических элементов при использовании интерактивных схем технологических процессов.

3. Разработан программный комплекс для проверки динамических связей технологических схем и баз данных, использующий систему кодирования Kraftwerk Kennzeichen System, позволяющий получать идентификаторы для типовых объектов энергоблока и выявляющий идентификаторы, которые не соответствуют базам данных. Предлагаемая методика использования кодов KKS для проверки динамических связей между графическими объектами интерактивных схем технологического процесса и соответствующих перемененных баз данных уменьшает нагрузку на оператора ядерного блока и снижает вероятность его ошибочных действий.

4. Разработан программно-технический комплекс, построенный на модульном принципе и объединивший различные функции вычислительных систем. Достигнуты цели по унификации представления данных оператору от различных систем, унифицировано представление данных различным группам пользователей. Расширены возможности представления данных. Получены новые возможности информационной поддержки оператора, облегчающие оценку состояния оборудования и принятия решения по управлению. Повышена надёжность и устойчивость работы оборудования за счет применения принципов дублирования, резервирования и разнообразия. Унифицированы технические средства программно-технического комплекса, что значительно упростило его эксплуатацию.

5. Создана информационно-вычислительная система, имеющая расширенные возможности интерфейса, гибкость конфигурации, большой набор вспомогательных инструментов и неограниченный потенциал по расширению как обрабатываемых параметров, так и функционала работы.

В распределенной специальной информационно-вычислительной системе управления шлюзовое прокладное программное обеспечение предусматривает до 40 протоколов обмена со смежными модулям, а также позволяет оптимизировать состав резервного оборудования и избежать программных конфликтов при обмене данными

Список источников к главе 4

4.1. Поваров В.П., Бакиров М.Б., Данилов А.Д. Автоматизированная система многопараметрического мониторинга параметров состояния энергетических установок АЭС. Воронеж: Научная книга, 2017. 276 с.

4.2. Поваров В.П., Бакиров М.Б., Данилов А.Д. Обработка данных в системе непрерывного мониторинга эксплуатационной повреждаемости критических элементов энергетических установок // Вестник Воронежского государственного технического университета. 2018. Т.14. № 1. С. 64-72.

4.3. Intellectual decision-making system in the context of potentially dangerous nuclear power facilities / A. Danilov, V. Burkovsky, S. Podvalny, K. Gusev, V. Povarov // MATEC Web of Conferences. 13. "13th International Scientific-Technical Conference on Electromechanics and Robotics "Zavalishin's Readings" - 2018". 2018. С. 2009.

4.4. Data support system for controlling decentralised nuclear power industry facilities through uninterruptible condition monitoring / V. Povarov, A. Danilov, V. Burkovsky, K. Gusev // MATEC Web of Conferences. 13. "13th International Scientific-Technical Conference on Electromechanics and Robotics "Zavalishin's Readings" - 2018". 2018. С. 2012.

4.5. Терехов Д.В., Данилов А.Д. Обработка потоков данных в информационных системах верхнего уровня управления процессами // Системы управления и информационные технологии. 2019. № 1 (75). С. 67-70.

4.6. Терехов Д.В., Сидоренко Е.В., Данилов А.Д. Тенденции развития АСУТП на Нововоронежской АЭС // Известия высших учебных заведений. Ядерная энергетика. 2017. № 3. С. 66-76.

4.7. Данилов А.Д., Терехов Д.В. Особенности архитектуры информационной системы реального времени // Системы управления и

информационные технологии. 2018. № 4 (74). С. 49-54.

4.8. Данилов А.Д., Синюков Д.С. Механизм распределения данных о специальных транзакциях с оперативным контентом в реальном времени на основе кэширования в гетерогенных объектах распределенной сети // Информационные технологии моделирования и управления. 2021. Т. 125. № 3. С. 216-223.

4.9. Поваров В.П., Терехов Д.В., Данилов А.Д. Особенности использования многоуровневой конфигурации специализированной информационной системы в задачах реализации принципа разнообразия систем безопасности 4-го блока Нововоронежской АЭС // Ядерная и радиационная безопасность. 2019. № S1. С. 41-45.

4.10. Терехов Д.В., Данилов А.Д. Проблема разработки принципов организации информационного обмена между иерархическими уровнями в многоуровневых АСУТП // Информационные технологии моделирования и управления. 2021. Т. 124. № 2. С. 151-157.

Заключение

Целью работы являлась разработка математического и программного обеспечения управления транзакциями с оперативным контентом на основе распределенного кэширования с использованием модифицированного протокола планирования транзакций.

В процессе выполнения исследования получены следующие основные результаты:

1. Разработан механизм распределения данных о специальных транзакциях с оперативным контентом, обеспечивающий минимизацию времени передачи данных.

2. Разработан алгоритм локального кэширования хронологически запрошенных данных транзакций, обеспечивающий уменьшение задержку передачи данных о специальных транзакциях

3. Создан алгоритм разделения и перенаправления запросов между клиентами, межобъектными интерфейсами или облаком, дополнительно учитывающий интегрированную информацию о состоянии кэширования, предполагаемом размере данных и пропускной способности.

4. Модифицирован протокол планирования транзакций в СУБД реального времени для учета не только временных ограничений транзакций, но и критериев, установленных пользователями базы данных.

5. Предложена архитектура СУБД реального времени с использованием модифицированного протокола планирования транзакций для улучшения качества предоставления услуг пользователям при реализации управления с обратной связью.

6. Создан программный комплекс для проверки динамических связей технологических схем и баз данных на основе системы кодирования Kraftwerk Kennzeichen System.

7. Созданы компоненты распределенной специальной

информационно-вычислительной системы управления с унифицированными механизмами межмодульного взаимодействия для оптимизации состава резервного оборудования и исключения программных конфликтов при обмене данными.

8. Элементы программного обеспечения зарегистрированы в ФИПС.

Список использованных источников

1. Вентцель Е.С., Овчаров Л.А. Теория случайных процессов и ее инженерные приложения. - М.: Наука. Гл. ред. физ.-мат. лит. - 1991.

2. Волков Д. Как оценить рабочую станцию // Открытые системы, №2, 1994, С. 44-48.

3. Волков Д., Французов Д., Новое поколение тестов SPEC. / Открытые системы, №4, 1996, С. 73-74.

4. Гамильтон С. Управление цепочками поставок с Microsoft Axapta. – М.: Альпина Бизнес Букс, 2005, 352 с.

5. Гешвинде Э., Шенинг Г.Ю. Разработка WEB-приложений на PHP и PostgreSQL. М.: ДиаСофт, 2003.- 608 С.

6. Ги К. Введение в локально-вычислительные сети. – М.: Радио и связь, 2000. – 190 с.

7. Гнеденко Б.В., Коваленко И.Н. Введение в теорию массового обслуживания, 2-е изд., М.:Наука, 1987. - 336 С.

8. Говорский А.Э., Кравец О.Я. Особенности взаимодействия подсистем при решении задач интегрального обслуживания неоднородного трафика // Системы управления и информационные технологии. 2008. Т. 31. № 1.1. С. 141-146.

9. ГОСТ 28195-89 Оценка качества программных средств. Общие положения. Введен 01.07.90. - 39 С.

10. ГОСТ 28806-90 Качество программных средств. Термины и определения. Введен 01.01.92. - 12 С.

11. ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. Введен 01.07.94. - 19 С.

12. Гребенникова Н.И., Синюков Д.С., Потудинский А.В. Программа мониторинга серверов баз данных с оповещением средствами

мессенджера: Свидетельство о регистрации программы для ЭВМ № 2022665036 от 09.08.2022. М.: ФИПС, 2022.

13. Гриневич А.И., Кулямин В.В., Марковцев Д.А., Петренко А.К., Рубанов В.В., Хорошилов А.В. Использование формальных методов для обеспечения соблюдения программных стандартов. – Тр. института системного программирования РАН. - 2006. - Т.10, с. 51-68.

14. Грязнов Н.Г., Димитриев Ю.К., Мелентьев В.А. Оптимизация отказоустойчивого вложения диагностического графа в тороидальные структуры живучих вычислительных систем// Автоматика и телемеханика. 2003. № 4. С. 133-152.

15. Данилов А.Д., Синюков Д.С. Механизм распределения данных о специальных транзакциях с оперативным контентом в реальном времени на основе кэширования в гетерогенных объектах распределенной сети// Информационные технологии моделирования и управления. №3(125), 2021, с. 216-223.

16. Данилов А.Д., Синюков Д.С. Подход к управлению транзакциями в гетерогенных распределенных реплицированных системах баз данных в реальном масштабе времени. Системы управления и информационные технологии. 2021;85(3):59-65.

17. Данилов А.Д., Синюков Д.С. Распределенная информационно-вычислительная система управления ядерным блоком// Завалишинские чтения'22: Сб. докл. XVII Междунар. конф. по электромеханике и робототехнике. - СПб., 2022. С. 33-37.

18. Данилов А.Д., Терехов Д.В. Особенности архитектуры информационной системы реального времени // Системы управления и информационные технологии. 2018. № 4 (74). С. 49-54.

19. Дастин Э., Рэшка Д., Джон П. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. - М.: Лори, 2003.

20. Дейт К. Введение в системы баз данных - 6-е изд. - Киев: Диалектика, 1998. - 784 С.

21. Дейтел Х., Дейтел П., Нието Т. Как программировать для Internet & WWW М.: Бином, 2002, - 1184 С.

22. Денисов А.А., Колесников Д.Н. Теория больших систем управления. Л.: Энергоиздат, 1982. - 288 С.

23. Диго С.М. Проектирование и использование баз данных. М.: Финансы и статистика, 1995. -208 С.

24. Дрожжинов В.И. От теста не уйдешь// Мир ПК, N 2, 1993.

25. Жожикашвили В.А., Вишневский В.М. Сети массового обслуживания. Теория и применение к сетям ЭВМ. М.: Радио и связь, 1988. – 192 С.

26. Журков А.П., Аминев Д.А., Гусева П.А., Мирошниченко С.С., Петросян П.А. Анализ возможностей применения подходов самодиагностирования к распределенной радиотехнической системе наблюдения// Системы управления, связи и безопасности. 2015. № 4. С. 114-122.

27. Задорожный В.Н. Модели и системы. Анализ научного мышления. - Омск, ОМГТУ, 1999. - 100 с.

28. Задорожный В.Н. Статистическое моделирование. - Омск: Изд-во ОмГТУ, 1996. - 92с.

29. Иванов П. Управление информационными системами: базовые концепции и тенденции развития // Открытые системы, №4, 1999, С. 37-43.

30. Камер Д. Компьютерные сети и Internet М.: Вильямс, 2002.- 640 С.

31. Кастаньетто Д. и др. Профессиональное PHP программирование. М.: Символ – Плюс, 2001. - 912 С.

32. Кениг Д., Штоян Д. Методы теории массового обслуживания.- М.: Радио и связь, 1981. – 127 с.

33. Киллелиа П. Тюнинг WEB-сервера. СПб.: Питер, 2003. - 528 С.

34. Клейнен Дж. Статистические методы в имитационном моделировании. - М.: Статистика, 1978. - Вып. 1. - 221 с.; Вып. 2 - 335 с.

35. Клейнрок Л. Вычислительные системы с очередями: Пер. с англ. М.: Мир, 1979 - 600 С.

36. Клейнрок Л. Теория массового обслуживания: Пер. с англ. М.: Машиностроение, 1979 – 432 С.

37. Когаловский М.Р. Энциклопедия технологий баз данных. М.: Финстат, 2002. - 800 С.

38. Кокс Д. Р., Смит У. Л. Теория очередей: Пер. с англ. М.: Мир, 1966 - 218 С.

39. Колесов Ю.Б., Сениченков Ю.Б., Визуальное моделирование, СПб.: Мир и Семья, 2000. - 256 С.

40. Конвей Р., Максвелл В., Миллер Л. Теория расписаний: Пер с англ. / Под ред. Г.П. Башарина. – М.: Наука, 1975 – 159 с.

41. Кравец О.Я., Барабанова И.А., Соляник С.А. Программа нечёткого к-поиска близких объектов. - Свидетельство о государственной регистрации программы для ЭВМ №2018666616 от 19.12.2018. М.: ФИПС, 2018.

42. Кравец О.Я., Соляник С.А. Алгоритмизация построения топологии системы управления потоками данных в ячеистых сетях// Системы управления и информационные технологии, №3(73), 2018. С. 66-69.

43. Кравец О.Я., Соляник С.А. Разработка процессной модели управления инновационной деятельностью при разработке наукоемких прикладных IT-решений в условиях импортозамещения в сфере информационных технологий// Информационные технологии моделирования и управления, №2(104), 2017. – С. 115-132.

44. Кравец О.Я., Соляник С.А. Резултат от интегрирането на специален математически и софтуер в практиката на метрологията// Парадигма. Електронно научно списание. Брой 2/2016. - С. 59-74. http://niiparadigma.ru/?page_id=1783.

45. Кравец О.Я., Соляник С.А. Управление вероятностными потоками в беспроводных ячеистых сетях на основе использования ориентированного ациклического графа// Системы управления и информационные технологии, №4(70), 2017. – С. 40-44.

46. Ланг К., Чоу Д. Публикация баз данных в Интернете. – СПб.: Символ-Плюс, 2004.

47. Ларионов А.М., Майоров С.А., Новиков Г.И. Вычислительные комплексы, системы и сети. Л.: Энергоатомиздат, 1987. - 256 С.

48. Лебедев А.Н., Чернявский Е.А. Вероятностные методы в вычислительной технике. М.: Высшая школа, 1986. - 312 С.

49. Левенчук А. Интранет предлагает решения для корпорации// Рынок ценных бумаг, №16, 1996.

50. Липский Н. Комбинаторика для программистов. М.: Мир, 1985. – 374 С.

51. Мазуркевич А., Еловой Д. PHP: настольная книга программиста. М.: Новое знание 2003. -480 С.

52. Майерс Г. Надежность программного обеспечения. - М.: Мир, 1980.

53. Мальцева С.В. Информационное моделирование WEB-ресурсов Интернет. М.: Глобус, 2003. - 216 С.

54. Матвеев В.Ф., Ушаков В.Г. Системы массового обслуживания. М.: Изд-во МГУ, 1984. – 240 С.

55. Мельтцер К., Михальски Б. Разработка CGI-приложений на Perl. М.: Вильямс, 2001. - 400 С.

56. Мещеряков Е.В., Хомоненко А.Д. Публикация баз данных в Интернете Спб.: BHV-Санкт-Петербург, 2001. - 560 С.

57. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. 2-изд. СПб: Питер-пресс, 2002. - 864 С.

58. Павловский Ю. Н. Имитационные модели и системы. М.: ФАЗИС, 2000. - 144 С.

59. Перегудов Ф.П., Тарасенко Ф.П. Введение в системный анализ. М: Высшая школа, 1989. - 367 С.

60. Петренко А., Бритвина Е., Грошев С., Монахов А., Петренко О. Тестирование сетевой инфраструктуры// Открытые системы, 09/2003.

61. Поваров В.П., Бакиров М.Б., Данилов А.Д. Автоматизированная система многопараметрического мониторинга параметров состояния энергетических установок АЭС. Воронеж: Научная книга, 2017. 276 с.

62. Поваров В.П., Бакиров М.Б., Данилов А.Д. Обработка данных в системе непрерывного мониторинга эксплуатационной повреждаемости критических элементов энергетических установок // Вестник Воронежского государственного технического университета. 2018. Т.14. № 1. С. 64-72.

63. Поваров В.П., Терехов Д.В., Данилов А.Д. Особенности использования многоуровневой конфигурации специализированной информационной системы в задачах реализации принципа разнообразия систем безопасности 4-го блока Нововоронежской АЭС // Ядерная и радиационная безопасность. 2019. № S1. С. 41-45.

64. Программный комплекс для проверки динамических связей технологических схем и баз данных /Д.С. Синюков, А.Д. Данилов, Д.А. Денисов, М.Е. Ушков// Вестник Воронежского государственного технического университета, 2022. Т. 18. № 4. С. 15-24.

65. Профессиональные стандарты в области информационных технологий. - М.: АП КИТ, 2008. - 616 с.

66. Разработка распределенной информационно-вычислительной системы управления ядерным блоком на Нововоронежской АЭС/ Д.С. Синюков, А.Д. Данилов, А.А. Самодеенко, А.А. Иванников// Вестник Воронежского государственного технического университета, 2021, т. 17, №6. – С. 20-27.

67. Ройс У. Управление проектами по созданию программного обеспечения. - М.: Лори, 2002.

68. Свами М., Тхуласираман К. Графы, сети и алгоритмы: Пер. с англ. М.: Мир, 1984. - 455 С.

69. Седова Н.А. Формирование лингвистических переменных для задач судовождения// Эксплуатация морского транспорта. – 2013. – №2(72). – С. 19-23.

70. Седова Н.А., Перечёсов В.С., Седов В.А. Удержание судна на курсе на базе нечеткой логики с учетом скорости судна// Автоматизация процессов управления. – 2013. – № 2. – С.74-79.

71. Синюков Д.С. Модифицированный протокол планирования для оптимизации выполнения транзакций без превышения их предельных сроков// Системы управления и информационные технологии, №3(89), 2021. – С. 89-94

72. Синюков Д.С. Потудинский А.В. Экспериментальное исследование системы автоматического поиска и устранения неисправностей в базе данных// Моделирование, оптимизация и информационные технологии. 2022;10(1). https://moitvivt.ru/ru/journal/pdf?id=1150. DOI: 10.26102/2310-6018/2022.36.1.030.

73. Синюков Д.С., Данилов А.Д. Применение систем управления базами данных как сервиса в сложных информационных системах. Труды

Всероссийской научной конференции «Достижения науки и технологий-ДНиТ-2021», Красноярск. 2021; http://ru-conf.domnit.ru/media/filer_public/42/d3/42d35c66-d78a-411c-b74d-9ef2f99ff7b6/3006-dnit-2021.pdf.

74. Синюков Д.С., Денисов Д.А., Ушков М.Е. Программа анализа динамических связей интерактивных схем и баз данных реального времени: Свидетельство о регистрации программы для ЭВМ № 2022665490 от 17.08.2022. М.: ФИПС, 2022.

75. Советов Б.Я., Яковлев С.А. Моделирование систем //3-е изд., М:Высшая школа, 2001. - 344 С.

76. Создание Intranet. Официальное руководство Microsoft. - СПб.: BHV, 1998.

77. Соляник А.А., Авсеева О.В., Кравец О.Я. Особенности управления процессами конкурентного проектирования программного обеспечения // Системы управления и информационные технологии. 2010. Т. 39. № 1.2. С. 308-312.

78. Соляник А.И., Кравец О.Я. Об аналитическом подходе к метамоделированию механизмов управления региональной социально-экономической системой // Системы управления и информационные технологии. №1.3(31), 2008. – С. 380-384.

79. Соляник А.И., Кравец О.Я. Пути и средства повышения эффективности управления качеством социально-экономической системы // Системы управления и информационные технологии. №2.2(32), 2008. – С. 304-308.

80. Соляник А.И., Кравец О.Я. Системно-структурное моделирование объекта управления на основе проектного подхода // Системы управления и информационные технологии, №4(46), 2011. – С. 43-45.

81. Соляник А.И., Соляник С.А. Формализация программной модели управления территориальной системой с использованием нечетких множеств// Системы управления и информационные технологии, №3(65), 2016. – С. 87-92.

82. Соляник С.А. Графовые инструменты управления пакетами без предварительного оповещения// Информационные технологии моделирования и управления, №3(111), 2018. – С. 234-239.

83. Соляник С.А. Способ определения диагностических способностей системы с полиномиальной оценкой трудоемкости по времени// Информационные технологии моделирования и управления, №6(102), 2016. – С. 437-445.

84. Соляник С.А. Управление потоками данных в беспроводных ячеистых сетях с однократной рассылкой по топологии графа// Системы управления и информационные технологии, №1(71), 2018. – С. 85-88.

85. Соляник С.А., Кравец О.Я. Исследование устойчивости процесса синтеза статических программных систем с блокировками// Экономика и менеджмент систем управления, №3.1(25), 2017. – С. 166-179.

86. Соляник С.А., Кравец О.Я. Модели и алгоритмы оперативного управления вероятностными процессами в многоуровневой специализированной системе// Информационные технологии моделирования и управления, №4(100), 2016. – С. 70-80.

87. Соляник С.А., Кравец О.Я. Размита моделиращи дейности лаборатории за калибриране като структурна предмет на териториално система, гарантираща единство на измеренията // Парадигма. Електронно научно списание. Брой 2/2016. - С. 145-158. http://niiparadigma.ru/?page_id=1783.

88. Соляник С.А., Кравец О.Я. Разработка процессной модели создания прикладных IT-решений для задач управления// Экономика и менеджмент систем управления, №3(25), 2017. – С. 79-86.

89. Соляник С.А., Кравец О.Я. Элементы синтеза статических программных систем с компонентами параллелизма// Информационные технологии моделирования и управления, №4(106), 2017. – С. 274-284.

90. Соляник С.А., Соляник А.И. Рационализация плановой диагностики дискретно-событийных систем в «холодном» режиме// Системы управления и информационные технологии, №4.1(66), 2016. – С. 184-188.

91. Стрелкова Е. Интеграция данных предприятия / Открытые системы, №4, 2003, С. 58-60.

92. Терехов Д.В., Данилов А.Д. Обработка потоков данных в информационных системах верхнего уровня управления процессами // Системы управления и информационные технологии. 2019. № 1 (75). С. 67-70.

93. Терехов Д.В., Данилов А.Д. Проблема разработки принципов организации информационного обмена между иерархическими уровнями в многоуровневых АСУТП // Информационные технологии моделирования и управления. 2021. Т. 124. № 2. С. 151-157.

94. Терехов Д.В., Сидоренко Е.В., Данилов А.Д. Тенденции развития АСУТП на Нововоронежской АЭС // Известия высших учебных заведений. Ядерная энергетика. 2017. № 3. С. 66-76.

95. Технология системного моделирования/ Е.Ф. Аврамчук, А.А. Вавилов, С.В. Емельянов и др. / Под ред. С.В. Емельянова. – М.: Машиностроение, 1988. – 320 с.

96. Томашевский В.Н., Жданова Е.Г. Имитационное моделирование в среде GPSS. Серия "Факультет". М.: Бестселлер, 2003. 400 С.

97. Томсетт Р. Радикальное управление ИТ проектами. М.: Лори, 2005.

98. Увайсов С.У., Иванов И.А., Кошелев Н.А. Методика обеспечения диагностируемости электронных средств космических аппаратов по ранговому критерию на ранних этапах проектирования// Качество. Инновации. Образование. 2012. № 1 (80). С. 60-63.

99. Уемов А.И. Системный подход и общая теория систем. М.: Мысль, 1978.- 272 С.

100. Управление качеством продукции. Введение в системы менеджмента качества / С. Пономарев, С. Мищенко, В.Я. Белобрагин. М.: Стандарты и качество, 2005.

101. Феллер В. Введение в теорию вероятностей и её приложения: в 2-х т. - М.: Мир,1967. - т.1, 498 с.

102. Феррари Д. Оценка производительности вычислительных систем. М.: Мир, 1981. - 576 С.

103. Французов Д. Оценка производительности вычислительных систем. Открытые системы, №2, 1996, С. 58-66.

104. Фролов А.В. Фролов Г.В. Локальные сети персональных компьютеров. М.: ДИАЛОГ-МИФИ, 1993. - 176 С.

105. Харари Ф. Теория графов. М.: Мир, 1973. - 300 С.

106. Харрингтон Дж. Проектирование реляционных баз данных. М: Лори-Пресс, 2000. - 230 С.

107. Харт Д.М. Системное программирование в среде Win32 - 2-е издание. М.: Вильямс, 2001. 464 С.

108. Хилайер С., Мизик Д. Программирование Active Server Pages. М.: Русская редакция. 2000. - 320 С.

109. Храмцов П.Б., Брик С.А. и др. Основы web-технологий. М.: ИНТУИТ.ру, 2003. – 512 С.

110. Цаленко М.Ш. Моделирование семантики в базах данных. М.: Наука, 1989. - 287 С.

111. Черняк Л. Intranet - объективная реальность, данная нам// Рынок ценных бумаг, №3, 1997.

112. Чжо З.Е., Чжо З.Л., Наинг Л.А. Разработка методики децентрализованной диагностики и диагностируемости с использованием модели тестирования// Вести высших учебных заведений Черноземья. 2015. № 3. С. 60-70.

113. Чистяков В.П. Курс теории вероятностей - 5-е изд. М.: Агар, 2000. - 255 С.

114. Шагурина Н. Web-службы: новая парадигма интеграции? / Сетевой журнал №2, М., 2003 – С. 14-17.

115. Шеннон Р. Имитационное моделирование систем. М.: Мир, 1978. - 418 С.

116. Amirijoo M., Hansson J., Son S.H. Specification and management of QoS in real-time databases supporting imprecise computations// IEEE Trans. Comput. 2006, 55, 304–319.

117. Ao W.C., Psounis K. Fast content delivery via distributed caching and small cell cooperation// IEEE Trans Mobile Comput. 2018; 17(5): 1048-1061.

118. Aranha R.F.M., Ganti V., Narayanan S., Muthukrishnan C.R., Prasad S.T.S., Ramamritham K. Implementation of a real-time database system// Inf. Syst. 1996, 21, 55–74.

119. Aranha R.F.M., Ganti V., Narayanan S., Muthukrishnan C.R., Prasad S.T.S., Ramamritham K. Implementation of a real-time database system// Inf. Syst. 1996, 21, 55–74.

120. Automatic Performance Diagnostics, Oracle, 2017. Режим доступа: https://docs.oracle.com/database/121/TGDBA/pfgrf_diag.htm#TGDBA026.

121. Automatic SQL tuning, Oracle, 2018. Режим доступа: https://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm

#CHDJDFGE.

122. B. L. Welch, “The Generalization of ‘Students’ problem when several different population varlances are involved’, / Biometrika, vol. 34, Issue 1-2, pp. 28–35, 1947.

123. Bernstein P.A., Hadzilacos V., Goodman N. Concurrency Control and Recovery in Database Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, 1987.

124. Brzeszczynski J., Ibrahim B.M. A stock market trading system based on foreign and domestic information// Expert Syst Appl. 2019; 118: 381-399.

125. Buttazzo G.C. Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications. Real-Time Systems Series. Springer, New York, 2004.

126. Cottet F., Delacroix J., Kaiser C., Mammeri Z. Ordonnancement Temps Rreel// Herm`es, 2000.

127. D. Dundjerski, S. Lazic, M. Tomasevic and D. Bojic, "Improving schema issue advisor in the Azure SQL database," / 25th Telecommunications Forum TELFOR, Belgrade, Serbia, 2017.

128. D. V. Aken, A. Pavlo, G. J. Gordon and B. Zhang, "Automatic Database Management System Tuning Through Large-scale Machine Learning," / 2017 ACM International Conference on Management of Data, Chicago, Illinois, USA, 2017.

129. Dar S., Franklin M.J., Jonsson B.T., Srivastava D., Tan M. Semantic Data Caching and Replacement// Proc. of the 22 VLDB Conf., Mumbai-Bombay, India, 1996, p. 330-341.

130. Data support system for controlling decentralised nuclear power industry facilities through uninterruptible condition monitoring / V. Povarov, A. Danilov, V. Burkovsky, K. Gusev // MATEC Web of Conferences. 13. "13th International Scientific-Technical Conference on Electromechanics and

Robotics "Zavalishin's Readings" - 2018". 2018. С. 2012.

131. de Assuncao M.D., da Silva V.A., Buyya R. Distributed data stream processing and МОИН computing: a survey on resource elasticity and future directions// J Netw Comput Appl. 2018; 103: 1-17.

132. DELL software / "Spotlight on SQL Server Enterprise Edition," 2016.

133. Hamdi S., Bouazizi E., Faiz S. QoS management in real-time spatial big data using feedback control scheduling// ISPRS Ann. Photogrammetry Remote Sens. Spat. Inf. Sci. 2015, II–3/W5, 243–248.

134. Hamdi S., Bouazizi E., Faiz S. QoS management in real-time spatial big data using feedback control scheduling// ISPRS Ann. Photogrammetry Remote Sens. Spat. Inf. Sci. 2015, II–3/W5, 243–248.

135. Hayes B. Cloud computing// Communication ACM, vol. 51, no. 7, p. 9, 2008.

136. Idera, "Idera SQL diagnostic manager," Idera, 2016. Режим доступа: https://www.idera.com/productssolutions/sqlserver/sqldiagnosticmanager/resources.

137. Intellectual decision-making system in the context of potentially dangerous nuclear power facilities / A. Danilov, V. Burkovsky, S. Podvalny, K. Gusev, V. Povarov // MATEC Web of Conferences. 13. "13th International Scientific-Technical Conference on Electromechanics and Robotics "Zavalishin's Readings" - 2018". 2018. С. 2009.

138. J. Kemnetz and C. Watson, "Overview of Azure Diagnostic Logs," Microsoft, 21 August 2017. Режим доступа: https://docs.microsoft.com/en-us/azure/monitoring-and- diagnostics/monitoring-overview-of-diagnostic-logs.

139. Jaziri W., Gargouri F. Ontology theory, management and design: an overview and future directions// Ontology Theory, Management and Design: Advanced Tools and Models, 2010, pp. 27–77.

140. Jaziri W., Sassi N., Damak D. Using temporal versioning and

integrity constraints for updating geographic databases and maintaining their consistency// J. Database Manag. 2015, 26, 30–59.

141. Jeyakumar V., Madani O., Parandehgheibi A., Kulshreshtha A., Zeng W., Yadav N. ExplainIt! - A declarative root-cause analysis engine for time series data. SIGMOD'19 2019 International Conference on Management of Data, Amsterdam, Holland. 2019;333-348.

142. Kang K.-D., Son S.H., Stankovic J.A., Abdelzaher T.F. A QoS-sensitive approach for timeliness and freshness guarantees in real-time databases// 14th Euromicro Conf. on Real-Time Systems. IEEE Computer Society, Vienna, Austria, 2002, pp. 203–212.

143. Kang W., Son S.H., Stankovic J.A. DRACON: QoS management for large-scale distributed real-time databases// JSW 2009, 4, 747–757.

144. Katet M., Bouazizi E., Sadeg B. Prise en Compte des Donnŕees Dŕerivŕees Temps Rŕeel dans Une Architecture de Contrˆole par Rŕetroaction// Revue rElectronique des Technologies de lInformation (e-TI), 2007, vol. 11.

145. Liu C.L., Layland J.W. Scheduling algorithms for multiprogramming in a hard real-time environment// J. ACM 1973, 20, 46–61.

146. Lu C., Stankovic J.A., Son S.H., Tao G. Feedback control real-time scheduling: framework, modeling, and algorithms// Real-Time Syst. 2002, 23, 85–126.

147. M. S. Hossain, S. Rahaman, A.-L. Kor, K. Andersson and C. Pattinson, "A Belief Rule Based Expert System for Datacenter PUE Prediction under Uncertainty," / IEEE Transactions on sustainable computing, vol. 2, no. 2, pp. 140-151, 2017.

148. Mach P., Becvar Z. Cloud-aware power control for real-time application offloading in mobile МОИН computing// Trans Emerg Telecommun Technol. 2016; 27(5): 648-661.

149. Microsoft, "Azure SQL Database: The world's first intelligent cloud database service," Microsoft, 22 August 2017. Режим доступа:

https://techcommunity.microsoft.com/t5/Microsoft-Ignite-Content-2017/Azure-SQL-Database-The-world-s-first-intelligent-cloud- database/td-p/98549.

150. Minas L., Ellison B Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers// Intel Press, 2009.

151. Mustafayev V.A., Zeynalabdiyeva I.S., Kravets O.Ja. Control model of parallel functioning production modules as fuzzy Petri nets. Journal of Physics: Conference Series. 2021;2094: 022003. https://doi.org/10.1088/1742-6596/2094/2/022003.

152. O.Brent, "sp_AskBrent," 2016. Режим доступа: https://www.brentozar.com/askbrent/.

153. Oleinikova S.A., Kravets O.Ja., Aksenov I.A., Frantsisko O.Yu., Rahman P.A., Atlasov I.V. The general scheme of the genetic algorithm for solving the task scheduling problem for a multistage system and assigning time for jobs. International Journal on Information Technologies and Security. 2021;13(4):47-58.

154. Overview of Oracle Enterprise Manager Cloud Control 13c, Oracle, 2016. Режим доступа: https://docs.oracle.com/cd/E63000_01/EMCON/overview.htm#EM CON109.

155. P. H. Callahan, "Expert systems for AT&T switched network maintenance," / AT&T Technical Journal, vol. 67, no. 1, pp. 93-103, 1988.

156. Palanisamy B., Singh A., Liu L. Cost-effective Resource Provisioning for MapReduce in a Cloud// IEEE Transactions on Parallel and Distributed Systems, 2014.

157. Raeder T., Dalessandro B., Provost F. Design principles of massive, robust prediction systems. 18th ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China. 2012;1357-1365.

158. S. Steve, "SQL Database Advisor," Microsoft, 2016. Режим доступа: https://azure.microsoft.com/en-us/documentation/articles/sql-database-

index-advisor/.

159. Sassi N., Brahmia B., Jaziri W., Bouaziz R. From temporal databases to ontology versioning: an approach for ontology evolution// Ontology Theory, Management and Design: Advanced Tools and Models, 2010, pp. 225–246.

160. Semghouni S., Amanton L., Sadeg B., Berred A. On new scheduling policy for the improvement of firm RTDBSs performances// Data Knowl. Eng. 2007, 63, 414–432.

161. Sinyukov D.S. Problems of troubleshooting in databases. Modern informatization problems in simulation and social technologies (MIP-2022'SCT): Proceedings of the XXVII-th International Open Science Conference. Yelm, WA, USA. 2022;162-174.

162. Sinyukov D.S., Danilov A.D., Aksenov I.A. Comparative analysis of software systems for intelligent operator support at the upper management level// APITECH

163. Song H., Kam-Yiu L., Jiantao W., Sang H.S., Aloysius K.M. Adaptive coscheduling for periodic application and update transactions in real-time database systems// J. Syst. Softw. 2012, 85, 1729–1743.

164. SQL Sentry, "Sql Sentry preformance advisor," SQL Sentry, 2016. Режим доступа: http://www.sqlsentry.com/products/performance-advisor/sql-server-performance.

165. T. Raeder, B. Dalessandro, F. Provost, "Design principles of massive, robust prediction systems" / 18th ACM SIGKDD international conference on Knowledge discovery and data mining, Beijing, China, 2012.

166. Tiwari R.G., Navathe SB., Kulkarni G.J. Towards Transactional Data Management over the Cloud// 2010 Second Int. Symp. Data, Privacy, ECommerce, 2010, pp. 100–106.

167. V. Jeyakumar, O. Madani, A. Parandehgheibi, A. Kulshreshtha, W.Zeng, N. Yadav, "ExplainIt! -- A declarative root-cause analysis engine for

time series data," / SIGMOD'19 2019 International Conference on Management of Data, Amsterdam, Holland, 2019.

168. V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellamanickam, S. Keerthi, S. Herbert and S. Dhulipalla, "Learning a Hierarchical Monitoring System for Detecting and Diagnosing Service Issues," / 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 2015.

169. Woochul K., Jaeyong C. QoS management for embedded databases in multicorebased embedded systems// Mob. Inf. Syst. 2015, 14.

170. Xiaolin L. Real time regression analysis in internet of stock market cycles// Cognit Syst Res. 2018; 52: 371-379.

171. Xiong M., Ramamritham K. Deriving deadlines and periods for real-time update transactions// IEEE Trans. Comput. 2004, 53, 567–583.

172. Zeddini B., Duvallet C., Sadeg B. Une Approche Qualitre de Service dans Les Syst`emes Multimredias Distribures// Revue rElectronique des Technologies de l'Information (e-TI), 2007, vol. 4.

173. Zhang Q., Zhani M.F., Boutaba R., Hellerstein J.L. Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud// IEEE transactions on cloud computing, vol. 2, no. 1, 2014.

174. Zhao Z., Min G., Gao W., Wu Y., Duan H., Ni Q. Deploying computing nodes for large-scale IoT: a diversity aware approach// IEEE Internet Things J. 2018; 5(5): 3606-3614.

175. Zipf G.K. Human Behavior and the Principle of Least Effort. - Addison-Wesley Press, 1949. 573 c.

Научное издание

СИНЮКОВ Денис Сергеевич
ДАНИЛОВ Александр Дмитриевич
КРАВЕЦ Олег Яковлевич

УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ С ОПЕРАТИВНЫМ КОНТЕНТОМ НА ОСНОВЕ РАСПРЕДЕЛЕННОГО КЭШИРОВАНИЯ

Монография

Издание публикуется в авторской редакции

Дизайн обложки С.А. Кравец

Подписано в печать 18.09.2023. Формат 60x84 1/16
Усл. печ. л. 9,5. Заказ 000. Тираж 500 экз.

ООО Издательство «Научная книга»
394077, Россия, г. Воронеж, ул. 60-й Армии, 25-120
http://www.sbook.ru/

Отпечатано с готового оригинал-макета
в ООО «Цифровая полиграфия»
394036, Россия, г. Воронеж, ул. Куколкина, 6
Тел. (473) 261-03-61