Федеральное государственное автономное
образовательное учреждение высшего образования
“Национальный исследовательский университет
“Высшая школа экономики”
(НИУ ВШЭ)

на правах рукописи

Рамон Антонио Родригес Залепинос

Растровые (Тензорные) СУБД: теоретические основы, программное обеспечение и приложения

РЕЗЮМЕ ДИССЕРТАЦИИ

на соискание учёной степени
доктора компьютерных наук

Москва — 2024

Аннотация

Растровые (Тензорные) СУБД – ТСУБД – стремятся стать лучшими системами для управления, обработки и даже визуализации больших многомерных массивов (тензоров). ТСУБД представляют собой молодую, быстро развивающаяся и по своей сути междисциплинарную область: многие основные типы данных в различных областях естественным образом моделируются с помощью многомерных массивов (тензоров). В этой диссертации представлен фундаментальный теоретический, системный и практический вклад в область ТСУБД. А именно, мы представляем два новых направления исследований и разработок (R&D): моделирование физического мира и настраиваемые запросы в ТСУБД. Мы также предлагаем новую формальную модель данных для ТСУБД, новые алгоритмы, подходы, архитектурные и реализационные аспекты работы с данными в виде многомерных массивов (тензоров), которые могут превосходить по скорости современные подходы и технологии на несколько порядков. Значимость нашего вклада продемонстрирована на широком спектре практических приложений и реальных данных. Диссертация основана на результатах, которые апробированы на ведущих международных конференциях в области компьютерных наук: VLDB и SIGMOD.

Оглавление

Тема диссертации

Р.А. Родригес Залепинос является автором ТСУБД ChronosDB, которая представлена на VLDB 2018 [1] и SIGMOD 2019 [2], BitFun на VLDB 2020 [3], нового направления R&D на SIGMOD 2021 [4], а также туториала на VLDB 2021 [5], WebArrayDB & ArrayGIS [6] и SimDB [7] на VLDB 2022 и FastMosaic [8] на VLDB 2023.

Они основаны на множестве теоретических основ и программных механизмов, которые применимы к области ТСУБД в целом. Это связано с тем, что теоретический и практический вклад этой диссертации охватывает широкий спектр аспектов и приложений ТСУБД, которые берут своё начало из различных практически важных областей, включая хранение, управление, обработку, обмен и визуализацию больших тензоров.

Кроме того, в диссертации впервые были идентифицированы и рассмотрены новые направления R&D: настраиваемые запросы и моделирование физического мира; об этом прямо говорится в соответствующих публикациях. Наконец, публикации демонстрируют каким образом представленные теоретические основы и программные механизмы успешно решают многие важные задачи, включая учёт промышленного опыта, взаимодействие с пользователем, интероперабельность, а также открывают многие перспективные возможности для R&D.

Тема диссертации состоит из нескольких частей, которые относятся к ТСУБД и последовательно рассматриваются в диссертации, кратко обобщающей ключевые идеи, представленные в соответствующих публикациях, в удобной для чтения форме. Разумеется, в публикациях читатель может найти очень подробные материалы, а также качественные видео и домашние страницы проектов, которые обычно сопровождают публикации.

Остановимся подробнее на формулировке темы диссертации и ее отражении на структуре диссертации. Главы 1 и 5 – введение и заключение соответственно. Роли глав 2, 3 и 4 изложены ниже.

Теоретические основы

В этой главе **закладываются** новые теоретические основы в области ТСУБД. Мы начинаем с новой модели данных ТСУБД, которая служит основой для всех других результатов. Далее мы представляем новые направления исследований и разработок, которые мы идентифицировали и рассмотрели в данной диссертации: настраиваемые запросы и моделирование физического мира. Наконец, мы описываем основные идеи, лежащие в основе наших новых и эффективных распределенных тензорных алгоритмов, включая многомерный ретайлинг и многопутевое соединение массивов (тензоров), и масштабируемых подходов из области наук о данных: канонический корреляционный анализ (CCA), многомерное обнаружение изменений (MAD) и итеративно повторно-взвешенное MAD (IR-MAD).

Программное обеспечение

Глава посвящена архитектурным и реализационным **аспектам** управления и обработки многомерных массивов (тензоров) инновационных ТСУБД и их компонент (ChronosDB, BitFun, SimDB, ArrayGIS, WebArrayDB и FastMosaic), благодаря которым можно быть на порядки быстрее других систем, ускорить интерактивный анализ данных, запускать моделирование полностью внутри ТСУБД и выполнять операции связанные с тензорами полностью внутри Web-браузера.

Приложения

Эта глава **демонстрирует** значимость нашего вклада на широком спектре реальных данных и практических приложений. Глава также представляет дополнительные архитектурные и реализационные аспекты. Наши алгоритмы и подходы позволяют быстро управлять, обрабатывать и визуализировать данные климата и дистанционного зондирования Земли. Алгоритмы и подходы также предназначены для быстрого пересчета (обновления) тензоров для задач продовольственной безопасности и быстрого реагирования в чрезвычайных ситуациях. Кроме того, можно быстро строить мозаики массивов (тензоров). Впервые, используя ТСУБД, мы также демонстрируем моделирование дорожного движения с использованием управления массивами (тензорами) в стиле СУБД и интероперабельный обмен данными.

Массив (Тензор)

На сегодняшний день научно-исследовательское направление ТСУБД находится на стадии формирования своего терминологического словаря. Более того, это относительно молодая область R&D и не существует общепринятых стандартов для схемы массива (тензора), языков запросов, набора поддерживаемых операций (операторов) и многих других аспектов ТСУБД [9, 52, 66].

Мы указывали ранее, что ТСУБД работают с многомерными массивами (тензорами): формальное определение дано в разд. 2.1. Однако здесь мы дополнительно уточним название этого класса СУБД: почему мы используем словосочетание «Массив (Тензор)»?

История начинается с Titan [12] и Paradise [17], одних из первых систем баз данных, специально ориентированных на операции с массивами. Они были направлены на данные дистанционного зондирования Земли, поскольку недавно запущенные спутники поставили сложные задачи перед сообществом управления данными, генерируя огромные объемы данных, в основном массивы размерностью 2 и 3. На тот момент эти данные были новыми для СУБД и фундаментально отличались от других поддерживаемых типов данных.

Быстро стало понятно, что многие основные типы данных во многих других областях естественным образом моделируются многомерными массивами. Поскольку 2-мерные массивы были наиболее частыми, даже одна из самых ранних систем называлась RasDaMan, что означает “менеджер растровых данных”. Однако было ясно, что система управления массивами данных выходит далеко за рамки растров. Это отразилось в названиях последующих систем, например, “СУБД для многомерных массивов” [71] или “Язык запросов для многомерных массивов” [30].

Хотя слово “массив” не отражает чётко того, что система может работать с массивом размерностью более 2, термин “многомерный массив” становится слишком длинным. Хуже того, сложно перевести “Array DBMS” на другие языки. По меньшей мере, по этим двум веским причинам термин “Array DBMS” следует пересмотреть.

Сегодня мы считаем, что “Тензорная СУБД” лучше всего отражает суть системы баз данных, которая управляет многомерными массивами. Тенденция к использованию слова “тензор” активно поддерживается не только сообществом специалистов по управлению данными, но и в более широкой исследовательской среде [45, 66]. Например, “тензоры являются

естественными многомерными обобщениями матриц" и "под тензором мы подразумеваем только массив с d индексами" [45].

Однако мы переживаем промежуточный период постепенного перехода к названию "Тензорная СУБД". Следовательно, в этой диссертации мы по-прежнему используем словосочетание "массив (тензор)" для ясности относительно того, к каким объектам мы обращаемся и для содействия этому переходу.

Слово "тензор" все чаще используется не только для массива с более чем двумя измерениями, но даже для матриц ("2-d массивы" или "2-d тензоры") [1]. Технически и семантически для современной ТСУБД практически нет разницы каким образом работать с 1-мерным, 2-мерным или N-мерным массивом, где $N \in \mathbb{Z}$ и $N > 2$ [66]. Поэтому в нашей диссертации мы используем словосочетание "массив (тензор)" либо в редких случаях одно из этих двух слов.

Отметим, что в нашей модели данных тензор – это более чем массив с d индексами, поскольку он поддерживает моделирование разных типов данных, включая сетки различных видов, разд. 2.1.

Также стоит отметить, что некоторые исследователи используют термин "куб данных" [10]. Однако чаще всего под ним понимают объект, который можно получить, выполнив ряд запросов к ТСУБД [66].

Независимо от текущих и возможных будущих вариантов именования систем баз данных, которые управляют различными типами многомерных массивов и именованию этих массивов (растры, тензоры, кубы данных и прочее), слово "тензор" отлично отражает то, что массив может быть многомерным, является международным термином и широко используется в исследовательском сообществе непосредственно для обозначения многомерных массивов.

Диссертация в контексте современного развития ТСУБД

История начинается с Titan [12], Paradise [17], и RasDaMan [48], о чем мы уже упоминали. Однако R&D в этой области застопорились до появления лавины больших многомерных данных. В результате этого, глубокие исследования в области управления массивами (тензорами) начали появляться лишь недавно. Поэтому мы ранее отмечали, что ТСУБД все еще является молодой областью R&D [9, 52, 66].

Можно классифицировать массив-ориентированные системы на Растровые (Тензорные) СУБД, хранилища массивов (тензоров), движки, библиотеки, инструменты, национальные инициативы (которые имеют более широкие цели, но могут иметь внутри себя системы для работы с массивами) и другие классы [52]. Обзор таких систем представлен в [9]. Однако только ChronosDB [54, 55], SciDB [15] и RasDaMan [48] являются хорошо известными и полноценными ТСУБД [73]. Среди них ChronosDB – единственная основанная на файлах ТСУБД: работает на месте (in situ) и применяет подход делегирования, разд. 2.2, обеспечивая многочисленные преимущества управления данными, в том числе более быстрое усвоение данных и интероперабельность.

Среди ТСУБД [73], только модели данных ChronosDB и RasDaMan формализованы, причем модель данных ChronosDB имеет уникальную комбинацию свойств, разд. 2.1. Хотя существуют и другие алгоритмы работы на месте (in situ) [52, 54], наши новые эффективные алгоритмы и подходы, построенные на основе нашей новой модели данных, превосходят современные подходы на порядки, разд. 4.1.1.

Индексирование – важнейшая методика любой СУБД. Сейчас существует три типа индексов ТСУБД: для (1) выбора значений ячеек, (2) извлечения гиперсрезов и (3) вычислений [11, 53, 76, 77]. Первые два ускоряют выбор ячеек в заданных диапазонах значений и индексов соответственно. Последний ускоряет вычисления над массивами (тензора-

ми) [52]. Тип индекса для вычислений был впервые предложен в нашей работе [53] и ускоряет запросы до 8 раз, разд. 2.3.

ТСУБД выполняют хранение [26, 28, 46, 53, 66], управление [80, 81, 82], обработку [59], анализ [13, 14, 25], распространение [9, 55], визуализацию [7, 24, 55, 63] массивов (тензоров) и машинное обучение на их основе [44, 62, 72, 73]. Мы выявили и рассмотрели еще одно новое направление R&D в области ТСУБД: моделирование физического мира полностью внутри ТСУБД, что дает много преимуществ и перспективных направлений R&D [56, 61]. Это прямо отмечено в публикации [56].

Выразительный язык запросов имеет огромное значение, поскольку он является отправной точкой для пользователей СУБД любого типа. К эксплуатирующимся языкам запросов ТСУБД относятся AFL, AQL [15], rasQL [9], Command Line [54], GMQL [22] и первый нативный язык UDF (User Defined Function), который мы предложили [56].

ТСУБД в основном работают на настольных машинах, серверах или компьютерных кластерах [15, 48, 54]. Мы разработали WebArrayDB, первую ТСУБД, которая полностью работает в Web-браузере и может ускорять управление массивами (тензорами) в 2 раза и более по сравнению с запросами к облачному серверу. Чтобы продемонстрировать ее возможности, мы также разработали новую Web-ГИС (географическую информационную систему), основанную на WebArrayDB [63].

Некоторые разделы также содержат данные о современном состоянии дел в области ТСУБД, чтобы предоставить дополнительные обоснования новизны и влияния нашего вклада. Узнать больше о ТСУБД можно из статей [9, 52, 66]. Все наши публикации содержат ссылки на смежные работы.

Растровые (Тензорные) СУБД: Красота и Эффект

R&D в области ТСУБД можно разделить на два основных класса: качественные и количественные [9, 52, 66]. Качественные и количественные R&D взаимосвязаны и влияют друг на друга.

Качественные R&D в основном сосредоточены на предоставлении конечным пользователям преимуществ, вытекающих из подхода к работе с массивами (тензорами) в стиле СУБД. Например, ТСУБД способствуют организации и оптимизации конвейеров, включающих управление большими массивами (тензорами), обработку, анализ, визуализацию, машинное обучение, моделирование и другие аспекты, предоставляя специальные языки запросов, интеграцию данных, автоматическое поддержание целостности данных, мощный ETL (Extract, Transform, Load) или усвоение данных, управление распределенными наборами данных в облаке, автоматическое распараллеливание, интеоперабельность и многое другое в одной системе.

Количественные R&D направлены на повышение производительности (ускорение конвейеров операций с массивами), уменьшение объемов хранения массивов (тензоров), снижение частоты ввода-вывода (например, запросов ввода-вывода в секунду в облаке или латентности сетевого ввода-вывода), снижение требований к памяти (оперативной, постоянной или любого другого типа), повышение масштабируемости (например, обрабатывать больше данных за единицу времени или на порядок меньшим временем выполнения, обслуживать больше пользователей при тех же ресурсах) и многие другие аспекты ТСУБД, критерии успеха которых обычно выражаются численно (например, скорость, объем, количество).

Существует множество различных подходов. Например, ускорить работу с массивом можно за счет большего объема памяти или, наоборот, обеспечить более компактное хранение массива за счет несколько

меньшей производительности. Количественные методы учитывают все уровни иерархии памяти, рис. 1a.

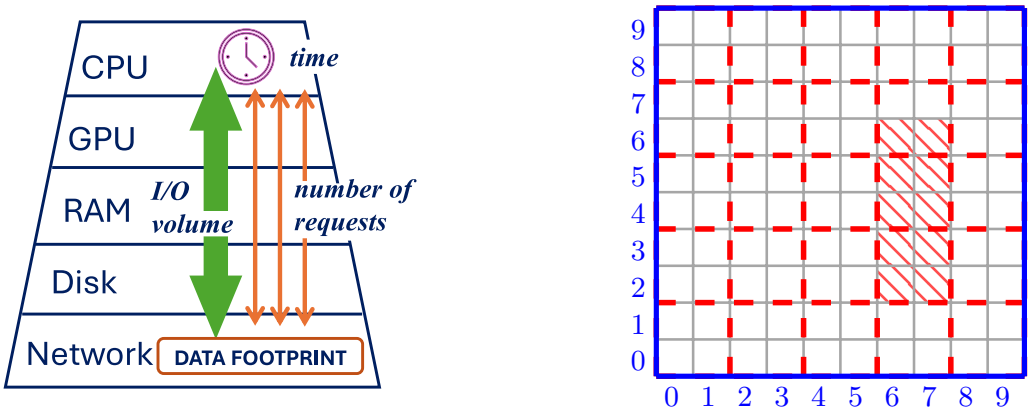
ТСУБД могут служить более удобными и бесшовными инструментами для ускорения управления массивами данных в различных исследовательских и практических областях. Пользователи могут абстрагироваться от проблем хранения, ввода-вывода, передачи, обмена и других сопутствующих проблем, быть уверенными в их эффективном решении и строить другие решения на основе ТСУБД.

Данная диссертация посвящена количественным R&D. Если количественный метод или подход работает внутри ТСУБД, пользователь не только получает преимущество от более быстрой работы с массивом или более компактного хранения, но и от всех других качественных преимуществ ТСУБД.

В чем красота количественных подходов ТСУБД? До сих пор мы перечисляли общее влияние ТСУБД на конечных пользователей. Однако красота этих подходов заключается в том, каким образом они реорганизуют хранение массивов (тензоров), ввод-вывод, порядок доступа и другие аспекты управления массивами (тензорами) для достижения вышеупомянутых целей производительности. Они предлагают новые алгоритмы, структуры данных, методы выполнения, индексирования, кэширования и сжатия данных, и это лишь некоторые из них.

Структуры данных и индексы могут быть статическими или динамическими, создаваться до выполнения запроса или адаптивно перестраиваться во время выполнения. Более того, многие аспекты ТСУБД полностью отличаются от других типов СУБД, например, соединения и индексы массивов совершенно не похожи на соединения и индексы реляционных таблиц, раздел. 2.2.2 и 2.3.2.

Рассмотрим упрощенный пример: матрица хранится на диске в построчном виде (строка за строкой). Легко полностью прочитать ее с помощью любого языка программирования в оперативную память, строку за строкой. Однако что делать, если матрица не помещается в память и мы читаем только те ее части, которые были заданы пользовательскими запросами? Мы также не знаем, какая часть будет запрошена следующей, а нам хотелось бы не только быстро отвечать на каждый запрос, но и снизить частоту и объем ввода-вывода. Если запрос запрашивает пунктирную область матрицы и используется та же схема расположения строк, мы можем прочитать 5 подстрок, каждая из которых содержит 2 ячейки, выполнив 5 запросов ввода-вывода, рис. 1b. Однако если мы будем разбивать (группировать) ячейки матрицы и рассматривать их в



(a) Влияние подходов (кэши не показаны)

(b) Чанкинг массива

Рис. 1: Иллюстрация количественных подходов ТСУБД

виде единиц ввода-вывода (чанки или группы разделены пунктирными красными линиями), то для ответа на тот же запрос нам потребуется всего 3 запроса ввода-вывода. Сокращение количества запросов ввода-вывода особенно важно в облаке, поскольку они тарифицируются отдельно, см. подробнее разд. 2.2.3.

Можно построить теоретически оптимальную (например, требующую наименьшего времени выполнения) структуру данных для некоторых конкретных (узких) случаев запросов. Однако проблема заключается в том, что зачастую априори не известно, какая компоновка массива (тензора) является оптимальной, поскольку она зависит от характеристик данных и рабочей нагрузки. Обычно порядок, скорость, масштаб и типы пользовательских запросов (или даже конвейеров, состоящих из серии запросов) не известны на 100% и меняются во время выполнения (особенно в многопользовательской среде). Многие методы учитывают определенные особенности массивов и рабочих нагрузок, чтобы разработать очень эффективные на практике, но редко теоретически оптимальные решения в такой динамичной области, какой является выполнение запросов в ТСУБД.

Глава 1

Введение

1.1 Актуальность темы диссертации

Многомерный массив (тензор) является основным типом данных в широком спектре областей, включая климатологию, экологию и дистанционное зондирование, а также в жизненно важных повседневных задачах, связанных с сельским и лесным хозяйством, городским управлением и чрезвычайными ситуациями [4, 52, 55, 60]. Все эти области испытывают колоссальный рост объемов тензорных данных, требующих эффективного управления, обработки, анализа и визуализации, и это лишь некоторые из большого множества областей.

Например, коммерческая компания Maxar (бывшая Digital Globe), ежедневно получает около 80 ТБ/день спутниковых изображений и уже накопила более 100 ПБ этих данных, т.е. массивов (тензоров), в облаке Amazon [34]. Sentinel – это семейство европейских спутников. Ежедневно распространяется около 203 ТиБ продуктов Sentinel, а общее количество загрузок составляет 80,5 ПиБ/год [67]. ECMWF (Европейский центр среднесрочных прогнозов погоды) имеет архив из 360 ПБ основных данных и 363 миллиона файлов. Он увеличивается на 287 ТБ в день и распространяет 215 ТБ ежедневно [19].

ТСУБД — молодое и быстро развивающееся направление. ТСУБД специально предназначены для выполнения эффективного управления и других релевантных операций с большими массивами (тензорами). Продвинутые исследования в области управления массивами (тензорами) только начинаются и многие возможности исследований и разработок все еще “лежат на поверхности” [52]. Таким образом, эта диссертация исследует различные обещающие возможности исследований и разработок и в то же время способствует разработке более эффективных и действенных решений вышеупомянутых важных практических задач.

Общая конечная цель или сверхцель R&D в области ТСУБД состоит в том, чтобы сделать их лучшими системами для управления большими многомерными массивами (тензорами). Р.А. Родригес Залепинос уже более 10 лет развивает ТСУБД в разных направлениях, предлагая для них новые направления R&D и приложения.

Степень разработанности темы диссертации. Область ТСУБД по праву можно считать молодой: общепринятых стандартов ещё не установлено, архитектуры и реализации всё ещё нуждаются в улучшении и доработке, а многие возможности R&D привлекательны и неисследованы [9, 52]. ТСУБД только расширяют свое присутствие в многочисленных областях исследований и разработок [58, 62, 65]. Таким образом, эта диссертация является своевременным вкладом в развитие R&D в области ТСУБД.

1.2 Цели и задачи диссертации

Повысить производительность (например, сократить время выполнения, объёмы ввода-вывода данных, требования к памяти) управления, обработки, анализа многомерных массивов (тензоров) и моделирования на основе Растровых (Тензорных) СУБД за счёт разработки новых подходов, алгоритмов и структур данных Растровых (Тензорных) СУБД выполнения запросов, индексирования, реорганизации хранилищ и других аспектов с целью предоставить эффективные операции с массивами (тензорами) в сочетании с преимуществами Растровых (Тензорных) СУБД (например, язык запросов, управление данными в стиле СУБД, интероперабельность) для исследовательских и практических областей, данные которых моделируются многомерными массивами (тензорами).

1.3 Основные результаты

Теоретическая и практическая значимость диссертации подтверждена публикациями в ведущих конференциях и изданиях по компьютерным наукам (разд. 1.4) и успешными применениями её результатов к реальным данным и практически важным областям (глав. 4).

В приведенном ниже списке указаны новизна и значимость наших результатов. **Обобщенно**, достигнуты следующие результаты:

- **Заложены** новые теоретические основы в области Растровых (Тензорных) СУБД ввиду получения следующих **новых результатов**:
  - модель данных для Растровых (Тензорных) СУБД, используемая нашими новыми эффективными алгоритмами, подходами, архитектурными и реализационными аспектами, которые в совокупности опережают по скорости существующие решения в десятки, сотни и тысячи раз, разд. 2.1
  - направления R&D, которые открывают широкий спектр новых возможностей для исследований, разработок и предоставляют многочисленные преимущества (например, управление данными в стиле СУБД, визуализацию, интероперабельность): настраиваемые запросы и моделирование полностью внутри Растровых (Тензорных) СУБД, раздел. 2.3 и 2.4.1
  - тип (класс, категория) индексов Растровых (Тензорных) СУБД: индекс вычислений, ускоряющий вычисления над массивами (тензорами); предыдущие работы ускоряют выполнение запросов по значениям и диапазонам, разд. 2.3.1
  - первый нативный язык UDF (User Defined Function) для Растровых (Тензорных) СУБД и новый оператор свертки, расширяющий их функциональность, например, впервые позволяющие проводить моделирование полностью внутри Растровых (Тензорных) СУБД, раздел. 3.3.1 и 3.3.2
  - эффективные распределенные алгоритмы для массивов (тензоров), которые значительно быстрее современных подходов: N-мерный ретайлинг, K-путевое соединение, агрегация, ресемплинг, решейпинг, чанкинг и другие, разд. 2.2
  - эффективные (с точки зрения времени выполнения) методы индексирования и быстрая иерархическая структура данных для быстрого пересчета тензоров, являющихся результатами настраиваемых математических функций, раздел. 2.3.2 и 2.3.3
  - масштабируемые (линейное время итерации, получая за итерацию канонические переменные CCA и коэффициенты трансформации IR-MAD) методы Науки о данных для Растровых (Тензорных) СУБД: CCA (канонический корреляционный анализ), многомерное обнаружение изменений (MAD) и итеративно повторно-взвешенное MAD (IR-MAD), разд. 2.5

- **Представлены** новые архитектурные и реализационные аспекты, которые в сочетании с нашими теоретическими результатами
  - превосходят современные системы на порядки (по времени выполнения), например SciDB, курируемую М. Стоунбрейкером, лауреатом премии А. Тьюринга ("Нобелевская премия по компьютерным наукам")
  - ускоряют интерактивную работу с данными в виде массивов (тензоров) за счёт ускорения соответствующих операций; например, разведочный анализ, вычисление настраиваемых функций, построение мозаики, разд. 2.3.2
  - впервые позволяют запускать моделирование полностью внутри Растровой (Тензорной) СУБД, используя наши новые компоненты: язык UDF, оператор свертки, планирование и другие, сохраняя при этом преимущества Растровых (Тензорных) СУБД (см. ниже), разд. 3.3
  - впервые обеспечивают выполнение Растровой (Тензорной) СУБД полностью внутри Web-браузера, чтобы сократить повышенное время отклика операций с массивами (тензорами), которое возникает из-за избыточных клиент-серверных коммуникаций в современных системах, разд. 3.4
- **Продемонстрирована** значимость наших результатов на широком спектре реальных данных и практических приложений:
  - управление, обработка и визуализация данных климата и дистанционного зондирования Земли с исключительной производительностью благодаря нашим новым подходам: **до 1024 раз быстрее** по сравнению с SciDB, передовой системой, курируемой М. Стоунбрейкером, лауреатом премии А. Тьюринга ("Нобелевская премия по компьютерным наукам"), разд. 4.1
  - пересчёт тензоров **до 8 раз быстрее** и работа в Web-браузерах **более 2 раз быстрее** в сценариях продовольственной безопасности и быстрого реагирования благодаря нашим новым и эффективным методам индексирования, структуре данных, архитектурным и реализационным аспектам, разд. 4.2
  - выполнение моделирования полностью в Растровой (Тензорной) СУБД (впервые): новая, сложная модель дорожного движения с использованием вышеупомянутых компонентов с **почти такой же производительностью**, что и у написанного вручную

кода в сочетании с **преимуществами стиля СУБД**, включая управление тензорами, их обработку и визуализацию, разд. 4.3

– создание высококачественных бесшовных мозаик из массивов (тензоров), которые уменьшают видимость швов, с помощью нашего нового масштабируемого подхода, который может работать **на порядок быстрее** для мозаики массивов (тензоров), чем популярная Python библиотека scikit-learn, разд. 4.4

Мы проанализировали исследовательские и промышленные модели данных и форматы хранения массивов (тензоров), массив-ориентированные системы, СУБД, языки запросов, схемы, алгоритмы и подходы к хранению, управлению, обработке, визуализации и интероперабельности, выявили их сильные стороны и ограничения, чтобы предложить новые и более эффективные подходы [51, 52, 54, 57, 64].

С целью демонстрации значимости нашего вклада, были также

- выявлены реальные приложения, которые в значительной степени зависят от массивов, например, R&D по климату и дистанционному зондированию Земли используют большие массивы ежедневно
- разработано вспомогательное программное обеспечение, облегчающее процесс ETL (Extract, Transform, Load) для усвоения данных в программные системы
- собраны и импортированы реальные данные массивов (тензоров) в ТСУБД, например, данные реанализа климата и дистанционного зондирования Земли
- разработаны программные компоненты для автоматического развертывания компьютерных кластеров в облаке и их масштабирования по мере необходимости
- проведены эксперименты на различных данных массивов (тензоров) реального мира и компьютерных кластерах разного размера в облаке, проведены анализ, сравнение и представление результатов
- разработаны специализированные интерактивные графические пользовательские интерфейсы для демонстрации применения ТСУБД и их компонентов к конкретным реальным приложениям и данным, раздел. 3.2.2, 3.4.3, 3.5.2 и 4.1.2

Методология, методы диссертационного исследования. Мы использовали теоретические и экспериментальные методы, включающие, помимо прочего: анализ, синтез, формализацию, моделирование, эксперимент и сравнение.

Авторский вклад составляет почти 100%, поскольку Р.А. Родригес Залепинос является единственным автором почти всех публикаций и проектов. Вклад включает, но не ограничивается: идеями, видением, моделью данных, подходами, алгоритмами, архитектурой и дизайном систем (программного обеспечения), выбором данных и приложений, экспериментами, текстом, рисунками, публикациями, презентацией работы, домашними страницами и видео. В некоторых незначительных случаях, вклад прямо указан в публикациях, разд. 1.4.

Основные положения, выносимые на защиту

- Новая модель данных для Растровых (Тензорных) СУБД со следующей **уникальной комбинацией свойств**: (1) представление массивов из нескольких файлов, произвольно распределенных по узлам кластера, в виде единого массива, (2) формализованный промышленный опыт для использования его в алгоритмах, (3) богатый набор типов данных (напр., гауссовы и нерегулярные сетки), (4) сопоставление подмассива с растровым файлом практически 1:1, но при этом без зависимости от формата файла, разд. 2.1

- Новые и эффективные распределенные алгоритмы работы с массивами (тензорами) на месте (in situ), включая многопутевое соединение, агрегацию, ресэмплинг, чанкинг и другие алгоритмы, производительность (скорость) которых значительно выше (**в десятки, сотни и тысячи раз**) на компьютерном кластере с реальными данными по сравнению с современными подходами, раздел. 2.2 и 4.1

- Новые и эффективные иерархическая структура данных и подходы индексирования настраиваемых функций, которые обеспечивают **ускорение до 8 раз** на реальных данных при выполнении настраиваемых запросов Растровых (Тензорных) СУБД, которые были впервые идентифицированы и рассмотрены в этой диссертации, раздел. 2.3, 4.2.1 и 4.2.2

- Новые архитектурные и реализационные аспекты, которые эффективно **организуют и демонстрируют** выполнение новых методов индексирования настраиваемых функций и иерархической структуры данных, которые **до 8 раз быстрее** на реальных данных для повторного вычисления результатов настраиваемых запросов Растровых (Тензорных) СУБД, раздел. 3.2 и 4.2

- Новый клеточный автомат дорожного движения, который ставит новые проблемы перед Растровыми (Тензорными) СУБД, решение которых **повысит эффективность моделирования** полностью внутри Растровых (Тензорных) СУБД; напр., несколько свойств у транспортных средств, локальные правила перехода работают с несколькими массивами на вход/выход, проверяют разные ограничения, необходимость в новом операторе свёртки, механизмах планирования, блокировки и управления версиями, раздел. 2.4.2 и 4.3.2

- Новый, масштабируемый способ выполнения канонического корреляционного анализа (CCA), популярного метода из Наук о данных, за линейное время, получая за итерацию по входным данным и канонические переменные CCA и коэффициенты трансформации IR-MAD (оператор MOSAIC), который может работать **на порядок быстрее**, чем популярная библиотека Python scikit-learn в контексте фазы усвоения данных в Растровых (Тензорных) СУБД, раздел. 2.5, 3.5 и 4.4

- Новые аспекты реализации, которые позволяют эффективно **организовывать и демонстрировать** выполнение масштабируемых методов Наук о данных на реальных данных: многомерное обнаружение изменений (MAD), итеративно повторно-взвешенное MAD (IR-MAD) и канонический корреляционный анализ (CCA), который обладает вышеуказанными свойствами (время работы, память, выходные данные), раздел. 3.5 и 4.4

- Новые архитектурные и реализационные аспекты ChronosDB, инновационной Растровой (Тензорной) СУБД, которые повышают эффективность управления, обработки и визуализации массивов (тензоров); ChronosDB превосходит SciDB **в среднем до 75 раз**. ChronosDB всегда быстрее и может превосходить SciDB **вплоть до 1024 раз**. SciDB курируется М. Стоунбрейкером, лауреатом Премии Тьюринга ("Нобелевская премия по компьютерным наукам"), раздел. 3.1.1 и 4.1.2

- Новые подходы управления массивами (тензорами) и методы распределенного выполнения запросов для Растровых (Тензорных) СУБД, которые работают с данными массивов (тензоров) на месте (in situ) и организуют эффективное выполнение предложенных в этой диссертации алгоритмов **до 1024 раз быстрее** на компьютерном кла-

стере и реальных данных по сравнению с современными подходами, раздел. 3.1.2, 3.1.3 и 4.1.1

- Новый оператор свертки для Растровых (Тензорных) СУБД и аспекты его реализации, который, в отличие от обычных операторов свертки, предоставляет ядру несколько окон ввода и позволяет ядру изменять произвольное количество ячеек в нескольких окнах вывода для поддержки **эффективного моделирования** в Растровых (Тензорных) СУБД, раздел. 3.3.1 и 4.3.1

- **Первый нативный** язык UDF (User Defined Function) для Растровых (Тензорных) СУБД, который в отличие от языков запросов к СУБД и языков программирования общего назначения, впервые позволяет выразить код логики моделирования для Растровых (Тензорных) СУБД, который предоставляет возможность явно задействовать эффективные средства выполнения нативных UDF в Растровых (Тензорных) СУБД, раздел. 3.3.2 и 4.3

- Новые механизмы планирования, управления версиями и блокировок для Растровых (Тензорных) СУБД, которые впервые позволяют эффективно запускать моделирование внутри Растровых (Тензорных) СУБД с **производительностью, сравнимой по скорости с написанным вручную кодом**, но со всеми преимуществами для пользователей, которые предоставляют Растровые (Тензорные) СУБД, раздел. 3.3.3 и 4.3

- Новые аспекты реализации SimDB и ChronosDB (например, рабочий процесс, реализация клеточного автомата дорожного движения, инициализация, исследование проактивных планов моделирования, интерактивная анимация), которые позволяют **организовать и продемонстрировать** эффективное моделирование впервые полностью внутри Растровой (Тензорной) СУБД, разд. 4.3

- Новые архитектурные и реализационные аспекты WebArrayDB, **первой** Растровой (Тензорной) СУБД, которая работает **полностью в Web-браузере**, и ArrayGIS, инновационной Web ГИС (географической информационной системы), которая основана на WebArrayDB; вместе они могут быть **более чем в 2 раза быстрее** на реальных данных по сравнению с запросами только к популярному облачному сервису для распространения и обработки массивов (тензоров), раздел. 3.4 и 4.2.3

1.4 Публикации и апробация работы

Эта диссертационная работа основана на следующих публикациях.

PVLDB – это **журнал квартиля Q1** со статьями в открытом доступе, см. Scimago: **топ 15** в области компьютерных наук по состоянию на май 2023 г. Каждой принятой статье предоставляется место для презентации на следующей доступной конференции VLDB: vldb.org/2021

VLDB – это ведущая международная конференция для исследователей управления данными и базами данных, поставщиков, практиков, разработчиков приложений и пользователей; ранг **CORE A*** (высший) согласно CORE rankings

SIGMOD – это ведущая международная конференция по управлению данными, системам баз данных и структурам данных (**CORE A***); например, B-деревья, R-деревья и RAID массивы были представлены на SIGMOD: https://2021.sigmod.org

Согласно требованиям диссертационного совета по компьютерным наукам (НИУ ВШЭ, 06/2022), не менее 10 статей, все индексируемые WOS (Web of Science), Scopus, приведены ниже. Статьи опубликованы в:

- журнале PVLDB (Q1, WOS, Scopus): [1], [3], [5], [6], [7] и [8]
- трудах конференции SIGMOD (CORE A*, WOS, Scopus): [2] и [4]
- издании Lecture Notes in Computer Science, LNCS (Q2, WOS, Scopus): [9], [11], [12] и [13]
- издании Communications in Computer and Information Science, CCIS (Q3, WOS, Scopus): [14] и [16]
- трудах конференций (WOS, Scopus): [10] и [15]

Защита должна осуществляться на основе не менее 7 из этих статей, в данном случае: [1], [2], [3], [4], [6], [7], [8] (публикации повышенного уровня) и [11], [12], [13], [14], [15] (публикации стандартного уровня).

Журнал PVLDB (PROCEEDINGS OF THE VLDB ENDOWMENT) включен в Список рекомендованных журналов НИУ ВШЭ «Список А» (ISSN: 2150-8097). Конференции VLDB & SIGMOD включены в Список рекомендованных конференций НИУ ВШЭ «Список A^{CONF}» (Перечень ведущих конференций в области компьютерных наук).

Авторский вклад. Р.А. Родригес Залепинос — единственный автор почти всех публикаций. Авторский вклад указан явно в публикациях в соавторстве.

Согласно требованиям диссертационного совета по компьютерным наукам НИУ ВШЭ, не менее 4 статей должны быть без соавторов, или соискатель должен быть главным соавтором. В этой диссертации, все публикации первого уровня, кроме одной (7 из 8), без соавторов и Р.А. Родригес Залепинос является основным соавтором одной статьи.

Все перечисленные публикации посвящены ТСУБД. Кроме того, все журнальные статьи и материалы конференций, являющиеся частью этой диссертации, были опубликованы после присуждения Р.А. Родригесу Залепиносу ученой степени кандидата технических наук в 2013 году.

Публикации повышенного уровня (все индексируются **обеими** системами, WOS и Scopus):

1. **R.A. Rodriges Zalipynis**. ChronosDB: Distributed, File Based, Geospatial Array DBMS. ***PVLDB***, 11(10): 1247–1261, 2018. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

   ChronosDB быстрее SciDB в **75×** в среднем. **ChronosDB** всегда быстрее SciDB, вплоть до **1034×** **ChronosDB** – нативная облачная СУБД. SciDB разрабатывается Paradigm4 и М. Стоунбрейкером, лауреатом Премии А. Тьюринга (ACM) ("Нобелевская премия по компьютерным наукам")

   Web-страница: http://chronosdb.gis.land

2. **R.A. Rodriges Zalipynis**. ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud. ***SIGMOD*** 2019, P. 1985–1988. DOI · **CORE A***, WOS & Scopus

   Представлен **новый распределенный WMTS сервер** непосредственно в ChronosDB и **новые** **ChronosDB** компоненты, позволяя пользователям взаимодействовать с **ChronosDB** и оценить её преимущества: (1) интерактивный графический Web-интерфейс, (2) толкователь планов выполнения (исследование сгенерированного DAG), и (3) визуализатор набора данных (отображение наборов данных **ChronosDB** на интерактивной Web-карте).

   Web-страница: http://chronosdb.gis.land

3. **R.A. Rodriges Zalipynis**. BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. ***PVLDB***, 13(12):

2909–2912, 2020. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

Новый класс запросов Тензорных СУБД идентифицирован и рассмотрен: настраиваемые запросы. **BitFun** предоставляет новые стратегии для переиндексирования тензоров, чтобы эффективно отвечать на запросы с похожими математическими функциями. **BitFun** быстрее до **8×** по сравнению с вычислением результатов с нуля.

Web-страница: http://bitfun.gis.land
Видео: https://youtu.be/uxGuZU8yEvE (7 мин.)

4. **R.A. Rodriges Zalipynis**. Convergence of Array DBMS and Cellular Automata: A Road Traffic Simulation Case. ***SIGMOD*** 2021, P. 2399–2403 · **open access** · DOI · **CORE A***, WOS & Scopus

   Представлено **новое направление R&D** в области Растровых (Тензорных) СУБД. Впервые выполнено моделирование непосредственно с помощью Растровой (Тензорной) СУБД. Среди многих преимуществ, предложенный подход обеспечивает эффективные средства распараллеливания, слияния данных, обработки тензоров и интероперабельность. ChronosDB, например, моделирует сложную сеть дорожного движения с несколькими полосами, перекрестками и светофорами.

   Web-страница: http://sigmod2021.gis.gg/
   Видео: https://youtu.be/3g1m1fNL6P4 (8 мин.)
   Видио 20 мин.: dl.acm.org/doi/10.1145/3448016.3458457

5. **R.A. Rodriges Zalipynis**. Array DBMS: Past, Present, and (Near) Future. ***PVLDB***, 14(12): 3186–3189, 2021. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

   Первый разносторонний туториал по научным направлениям и разработкам в области Растровых (Тензорных) СУБД. Представляет многочисленные перспективные возможности для исследований и разработок.

Длительность: 90 минут (1.5 часа), включен в основную программу конференции, основной день конференции. Всего принято только 8 туториалов на VLDB 2021: список туториалов VLDB 2021

Web-страница: http://vldb2021.gis.gg/

Домашняя страница представляет высококачественные видео (1.5 часа)

6. **R.A. Rodriges Zalipynis**, N. Terlych. WebArrayDB: A Geospatial Array DBMS in Your Web Browser. ***PVLDB***, 15(12): 3622–3625, 2022. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

   Первая Растровая (Тензорная) СУБД, которая работает полностью внутри Web-браузера: **WebArrayDB**. Статья также представляет **ArrayGIS**, новую Web ГИС на основе **WebArrayDB**. Системы могут быть в **2×** и более раз быстрее по сравнению с работой только с Sentinel-Hub, популярным облачным сервисом для распространения данных Sentinel (недавно был приобретён компанией Planet).

   Вклад Р.А. Родригеса Залепиноса (указан на странице №3625 статьи): идеи, подходы, архитектура программного обеспечения и дизайн, выбор библиотек, статья.

   Web-страница: https://wikience.github.io/webdb2022
   Видео: https://youtu.be/NnpNR8GArj0 (5 мин.)

Системы находятся в свободном доступе: http://webdb.gis.gg

7. **R.A. Rodriges Zalipynis**. SimDB in Action: Road Traffic Simulations Completely Inside Array DBMS. ***PVLDB***, 15(12): 3742–3745, 2022. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

   Первая Растровая (Тензорная) СУБД, которая выполянет полный производственный цикл моделирования полностью внутри себя: от подготовки данных до моделирования и вычисления статистики.

   ТСУБД могут принести многочисленные преимущества при моделировании благодаря “СУБД подходу”, напр., мощному распараллеливанию и интероперабельности. В то же вре-

мя, моделирование расщиряет горизонт Растровых (Тензорных) СУБД и открывает широкий спектр возможнстей для новых R&D.

Web-страница: https://wikience.github.io/simdb2022
Видео: https://youtu.be/NnpNR8GArj0 (5 мин.)

8. **R.A. Rodriges Zalipynis**. FastMosaic in Action: A New Mosaic Operator for Array DBMSs. ***PVLDB***, 16(12): 3938–3941, 2023. DOI · PDF · **Article, Q1 Journal**, indexed by WOS & Scopus

   FastMosaic является новым оператором мозаики для Растровых (Тензорных) СУБД, который оснащен нашим новым, масштабируемым способом выполнения Канонического корреляционного анализа (CCA) за линейное время, получая канонические переменные CCA вместе с коэффициентами трансформации мозаики за тот же проход по входным данным.

   Новый алгоритм CCA может быть **на порядки быстрее** по сравнению с популярной библиотекой Python scikit-learn для целей создания мозаик массивов (тензоров).

   CCA является популярным инструментом для поиска корреляций в многомерных наборах данных. CCA широко используется в науках о данных для уменьшения размерности и обнаружения скрытых переменных.

   Web-страница: https://wikience.github.io/fastmosaic2023
   Видео: https://youtu.be/DXC4r5DCd6k (15 мин.)

Публикации стандартного уровня (все индексируются **обеими** системами, WOS и Scopus):

9. R. A. Rodriges Zalipynis (2021) **Towards Machine Learning in Distributed Array DBMS: Networking Considerations**, *Machine Learning for Networking: Third International Conference*, MLN 2020, Paris, France, November 24–26, 2020, Revised Selected Papers, **Lecture Notes in Computer Science (LNCS)**, Vol. 12629, P. 284–304, Springer, 2021. DOI – **WOS, Scopus, Q2**

10. R. A. Rodriges Zalipynis (2019) **Evaluating Array DBMS Compression Techniques for Big Environmental Datasets**, *Proceed-*

ings of the 2019 IEEE 10th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), P. 859–863. ***IEEE***, 2019. DOI – **WoS, Scopus**

11. R. A. Rodriges Zalipynis (2018) **Generic Distributed In Situ Aggregation for Earth Remote Sensing Imagery**, *Proceedings of Analysis of Images, Social Networks and Texts – 7th International Conference*, AIST 2018, July 5–7, 2018, Revised Selected Papers. **Lecture Notes in Computer Science (LNCS)**, Vol. 11179, P. 331–342, Berlin: Springer, 2018. DOI – **WOS, Scopus, Q2**

12. R. A. Rodriges Zalipynis (2018) **Distributed In Situ Processing of Big Raster Data in the Cloud**, *Perspectives of System Informatics – 11th International Andrei P. Ershov Informatics Conference*, PSI 2017, June 27–29, 2017, Revised Selected Papers, **Lecture Notes in Computer Science (LNCS)**, Vol. 10742, P. 337–351, Springer, 2018. DOI – **WOS, Scopus, Q2**

13. R.A. Rodriges Zalipynis, E. Pozdeev, A. Bryukhov **Array DBMS and Satellite Imagery: Towards Big Raster Data in the Cloud**, *International Conference on Analysis of Images, Social Networks and Texts* (AIST), Revised Selected Papers. **Lecture Notes in Computer Science (LNCS)**, Vol. 10716, P. 267–279, Springer, 2017. DOI – **WOS, Scopus, Q2**

    Best talk award, certificate: PDF

    > R.A. Rodriges Zalipynis contributions (stated on page №277 and on the Springer Web Portal):
    >
    > all text, figures, design and implementation of algorithms and ChronosServer, ChronosServer data model, Azure management code, SciDB import code, experimental setup, experiments.

14. R.A. Rodriges Zalipynis, A. Bryukhov, E. Pozdeev (2017) **Retrospective Satellite Data in the Cloud: An Array DBMS Approach**, Supercomputing. RuSCDays 2017. **Communications in Computer and Information Science (CCIS)**. Revised Selected Papers. Vol. 793, P. 351–362. Springer. DOI – **WOS, Scopus, Q3**

    > R.A. Rodriges Zalipynis contributions (stated on page №361 and on the Springer Web Portal):

all text, figures, design and implementation of algorithms and ChronosServer, ChronosServer data model, Azure management code, SciDB import code, experimental setup, experiments.

15. R. A. Rodriges Zalipynis (2017) **Array DBMS in Environmental Science: Sea Surface Height Data in the Cloud**, *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, P. 1062–1065, ***IEEE***, 2017. DOI – **WOS, Scopus**

16. R. A. Rodriges Zalipynis (2016) **ChronosServer: Fast In Situ Processing of Large Multidimensional Arrays with Command Line Tools**, *Supercomputing. RuSCDays 2016.* Revised Selected Papers. **Communications in Computer and Information Science (CCIS)**. Vol. 687, P. 27–40, Springer, 2016. DOI – **WOS, Scopus, Q3**

Доклады на конференциях:
Названия конференций, город, тема доклада, год их проведения:

- The 49th International Conference on Very Large Data Bases (Vancouver, Canada). FastMosaic in Action: A New Mosaic Operator for Array DBMSs, 2023
- The 48th International Conference on Very Large Data Bases (Sydney, Australia). WebArrayDB: A Geospatial Array DBMS in Your Web Browser, 2022
- The 48th International Conference on Very Large Data Bases (Sydney, Australia). SimDB in Action: Road Traffic Simulations Completely Inside Array DBMS, 2022
- Двадцатая международная конференция «Современные Проблемы Дистанционного Зондирования Земли из Космоса (Физические основы, методы и технологии мониторинга окружающей среды, потенциально опасных явлений и объектов)» (Москва). Доклад:
  ChronosDB: высокопроизводительная обработка данных дистанционного зондирования Земли, 2022
- ACM SIGMOD/PODS International Conference on Management of Data (Xi'an, Shaanxi, China). Convergence of Array DBMS and Cellular Automata: A Road Traffic Simulation Case, 2021
- The 47th International Conference on Very Large Data Bases (Copenhagen, Denmark). Array DBMS: Past, Present, and (Near) Future, 2021
- The 46th International Conference on Very Large Data Bases (Tokyo, Japan). BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS, 2022

- 3rd International Conference on Machine Learning for Networking (MLN'2020) (Paris, France). Towards Machine Learning in Distributed Array DBMS: Networking Considerations, 2020
- ACM SIGMOD/PODS International Conference on Management of Data (Amsterdam, Netherlands). ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud, 2019
- IEEE 10th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Metz, France). Evaluating Array DBMS Compression Techniques for Big Environmental Datasets, 2019
- The 44th International Conference on Very Large Data Bases (Rio de Janeiro, Brasil). ChronosDB: Distributed, File Based, Geospatial Array DBMS, 2018
- The 7th International Conference on Analysis of Images, Social Networks, and Texts (AIST'2018) (Moscow, Russia). Generic Distributed In Situ Aggregation for Earth Remote Sensing Imagery, 2018
- Perspectives of System Informatics - 11th International Andrei Ershov Informatics Conference, PSI 2017 (Moscow, Russia). Distributed In Situ Processing of Big Raster Data in the Cloud, 2017
- Russian Supercomputing Days (Moscow, Russia). Retrospective Satellite Data in the Cloud: An Array DBMS Approach, 2017
- 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2017) (Bucharest, Romania). Array DBMS in Environmental Science: Sea Surface Height Data in the Cloud, 2017
- Analysis of Images, Social Networks and Texts. 6th International Conference, AIST 2017 (Moscow, Russia). Satellite Imagery and Array DBMS: Towards Big Raster Data in the Cloud, 2017
- Russian Supercomputing Days (Moscow, Russia). In-situ processing of big raster data with command line tools, 2016

1.5 Исходный код программного обеспечения

Архитектурные и реализационные аспекты программного обеспечения являются одними из результатов данной диссертации. Однако перечисленное далее программное обеспечение либо его исходный код не являются частью либо результатами данной диссертации: ChronosDB [54, 55], BitFun [53], SimDB [56, 61], WebArrayDB & ArrayGIS [63] и FastMosaic [59].

Глава 2

Теоретические основы

2.1 Новая формальная модель данных ТСУБД

Мы совершенствовали модель в течение нескольких лет [51, 57, 64]. В настоящее время она является устоявшейся [52, 54] и служит основой для результатов, описанных в этой работе, включая новые направления R&D, которые появляются поверх модели без ее модификаций благодаря ее основательности. Эта формальная модель позволяет поставить ТСУБД на прочную теоретическую основу и использовать промышленный опыт. Например, модель позволяет дать строгие формальные определения тензорных наборов данных и операций, а также работать с массивами на месте (in situ), в их исходных форматах файлов. В свою очередь, это позволяет строить другие теоретические основы на базе этой модели. Например, на основе строгих и формальных определений появляется возможность реализовать различные механизмы, позволяющие проводить эффективное моделирование полностью внутри ТСУБД, раздел. 3.3 и 4.3.

2.1.1 Мотивация создания новой модели данных

Модели данных ТСУБД пытаются учесть и обобщить разнообразие больших массивов: 1-мерные временные ряды, 2-мерные геопространственные сетки, пути, полосы или другие массивы, в которых одно из измерений представляет собой координаты (напр., время, высота, № модели), 3-мерные пространственно-временные кубы, например, астрономические наблюдения или биомедицинские данные, неструктурированные сетки и вообще любые данные, которые могут быть смоделированы многомерными массивами (тензорами).

Однако массивы традиционно хранятся в файлах, а не в базах данных. Такие файлы имеют сложные именования, различные системы координат, форматы и поддерживаемые типы данных: это лишь некоторые особенности. Важно также, что набор данных почти всегда разбит на много файлов. Более того, резкое увеличение объемов данных в виде массивов стимулирует использование компьютерных кластеров для распределенного управления, обработки, анализа и визуализации массивов.

Промышленные модели массивов (тензоров) предоставляют унифицированный доступ к массиву (тензору), который может быть представлен в различных форматах хранения или сервисных API. Наиболее широко используемыми и устоявшимися моделями массивов являются CDM, GDAL и ISO, в той или иной степени отображающиеся друг на друга [39]. Эти модели являются результатом десятилетий

значительного практического опыта, но их объединяет ключевой недостаток: они работают только с одним файлом, а не с набором файлов в виде единого массива. Наиболее известными исследовательскими моделями и алгебрами для плотных многомерных массивов общего назначения являются AML [33], AQL [30] и RAM [71]. Они отображаемы на алгебру массивов [8].

Создание новой формальной модели данных было обусловлено несколькими характеристиками, которые одновременно не присутствуют в существующих моделях данных [54]: (1) представление массивов из нескольких файлов, произвольно распределенных по узлам кластера, в виде единого массива, (2) формализованный промышленный опыт для использования его в алгоритмах, (3) богатый набор типов данных (напр., гауссовы и нерегулярные сетки), (4) сопоставление подмассива с растровым файлом практически 1:1, но при этом без зависимости от формата файла.

Наша модель имеет несколько уровней. Пользователь воспринимает большой многомерный массив на логическом уровне в виде одного объекта, разд. 2.1.2. Набор массивов (тензоров) системного уровня (подмассивов) распределяется между узлами кластера и хранится в виде обычных файлов в различных форматах на втором уровне модели, разд. 2.1.3. Операция с логическим массивом (тензором) пользовательского уровня отображается на последовательность операций с соответствующими подмассивами системного уровня в наших алгоритмах, разд. 2.2. Более подробная информация о преимуществах и свойствах нашей новой модели данных содержится во введении, разделах 2 и 4 [54].

Раздел 2.2 демонстрирует, что наши новые и эффективные алгоритмы могут использовать промышленный опыт путём делегирования части работы на одном узле кластера стороннему программному обеспечению. Глава 4 демонстрирует, каким образом операции и алгоритмы, основанные на нашей новой модели данных, служат для эффективного управления, обработки, визуализации и анализа разнообразных реальных данных в виде массивов (тензоров) в различных реальных приложениях.

2.1.2 Тензоры или многомерные массивы

Представим логический уровень нашей модели данных [52, 54], рис. 2.1.

N-мерным массивом (тензором) является отображение $A : D_1 \times D_2 \times \cdots \times D_N \mapsto \mathbb{T}$, где $N > 0$, $D_i = [0, l_i) \subset \mathbb{Z}$, $0 < l_i$ есть конечное целое число, а $\mathbb{T}$ является числовым типом[1]. Назовём l_i *размером* либо *длиной* измерения i[2]. Обозначим N-мерный массив (тензор) следующим образом:

$$A\langle l_1, l_2, \ldots, l_N \rangle : \mathbb{T} \tag{2.1}$$

Форма A обозначается посредством $l_1 \times l_2 \times \cdots \times l_N$, а *размер* A обозначается при помощи $|A|$; при этом $|A| = \prod_i l_i$. Мы ссылаемся на значение *ячейки* либо *элемента* A с целочисленными индексами $(x_1, x_2, \ldots, x_N)$ посредством $A[x_1, x_2, \ldots, x_N]$, где $x_i \in D_i$. Значение каждой ячейки A имеет тип $\mathbb{T}$. Символы `NA` обозначают отсутствие значения.

Массив (тензор) может быть инициализирован после его определения посредством перечисления значений его элементов. Например, следующая запись определяет и инициализирует 2-мерный массив (матрицу) целых чисел: $A\langle 2, 2 \rangle$: int =

[1] Тип C++ в соответствии с ISO/IEC 14882 может рассматриваться для конкретики в отношении диапазонов значений, размера в байтах и других свойств

[2] Здесь и далее $i \in [1, N] \subset \mathbb{Z}$

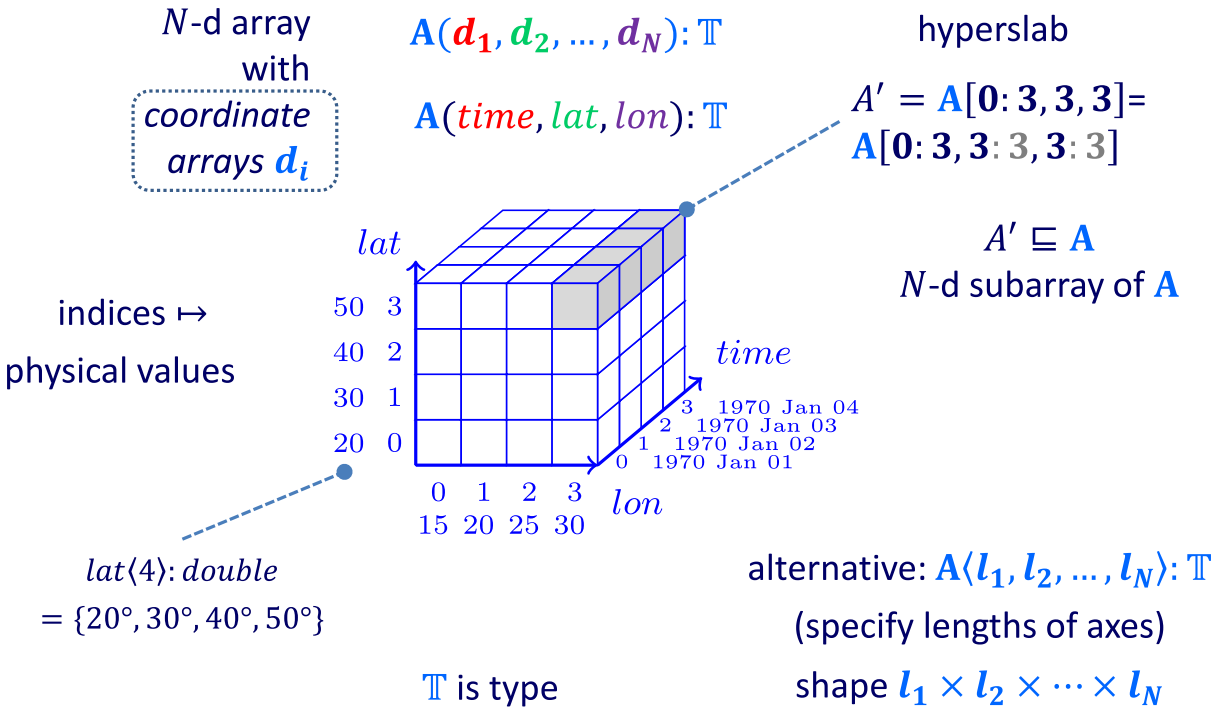


Рис. 2.1: Иллюстрация массива (тензора) пользовательского уровня

$\{\{1, 2\}, \{\mathtt{NA}, 4\}\}$. В этом примере, $A[0, 0] = 1$, $A[1, 0] = \mathtt{NA}$, $|A| = 4$, при этом A имеет форму 2×2.

Индексы x_i опционально отображаются на конкретные значения i-го измерения с помощью *координатных* массивов $A.d_i\langle l_i\rangle : \mathbb{T}_i$, где $\mathbb{T}_i$ есть полностью упорядоченное множество, $d_i[j] < d_i[j + 1]$, а $d_i[j] \neq \mathtt{NA}$ для $\forall j \in D_i$. В этом случае, массив A из ф-л. (2.1) может быть также определён следующим образом:

$$A(d_1, d_2, \ldots, d_N) : \mathbb{T} \tag{2.2}$$

Гиперсрезом $A' \sqsubseteq A$ является N-мерный подмассив A. Гиперсрез A' определяется следующей нотацией

$$A[b_1 : e_1, \ldots, b_N : e_N] = A'(d'_1, \ldots, d'_N) \tag{2.3}$$

где $b_i, e_i \in \mathbb{Z}$, $0 \leqslant b_i \leqslant e_i < l_i$, $d'_i = d_i[b_i : e_i]$, $|d'_i| = e_i - b_i + 1$ и для всех $y_i \in [0, e_i - b_i]$ имеет место следующее:

$$A'[y_1, \ldots, y_N] = A[y_1 + b_1, \ldots, y_N + b_N] \tag{2.4a}$$
$$d'_i[y_i] = d_i[y_i + b_i]. \tag{2.4b}$$

Уравнения (2.4a) и (2.4b) гласят, что A и A' имеют общее координатное подпространство, в пределах которого значения ячеек A и A' совпадают. Размерности A и A' являются одинаковыми. В определениях гиперсрезов, ф-л. (2.3), мы будем опускать "$: e_i$" если $b_i = e_i$ либо полностью "$b_i : e_i$" если $b_i = 0$ и $e_i = |d_i| - 1$.

Массивы (тензоры) X и Y перекрываются iff $\exists Q : Q \sqsubseteq X \wedge Q \sqsubseteq Y$. Массив (тензор) Q называется *наибольшим общим гиперсрезом* X и Y, который обозначается $gch(X, Y)$ iff $\nexists W : (W \sqsubseteq X) \wedge (W \sqsubseteq Y) \wedge (Q \sqsubseteq W) \wedge (Q \neq W)$. Массив (тензор) X покрывает массив (тензор) Y iff $Y \sqsubseteq X$.

Мы называем массив (тензор) на рис. 2.1, массивом (тензором) пользовательского уровня: это логическое представление возможно очень большого N-мерного массива (тензора), который может не поместиться в пределах одной машины, будь то оперативная память, постоянное либо любое другое хранилище. Массив (тензор) на рис. 2.1 имеет три измерения и три координатных массива, которые отображают индексы измерений во временные и пространственные координаты. Гиперсрез на рис. 2.1 представляет собой временной ряд для точки с координатами $(50°, 30°)$.

Обычно СУБД может предоставить схему для своего объекта (объектов). Например, реляционные СУБД могут выдать схему для таблиц. Поскольку наша модель предназначена для ТСУБД, массив (тензор) пользовательского уровня также может иметь схему, которая описана в разд. 3.1.2 и [52, 54]. Модель реализована в ChronosDB, чьи пользователи избегают изучения новой нотации схемы и могут просматривать массивы (тензоры) привычным для них способом.

2.1.3 Многоуровневые, распределенные наборы данных

Второй уровень нашей модели состоит из набора тензоров (подмассивов) системного уровня, которые распределены между узлами кластера и хранятся в виде обычных файлов в различных форматах [52, 54], рис. 2.2. Массив (тензор) пользовательского уровня никогда не материализуется и не хранится в явном виде: операция с объектом пользовательского уровня отображается на последовательность операций с соответствующими подмассивами.

Наборы данных могут быть сырыми и регулярными. Ограничения по форме не накладываются на сырые массивы (тензоры) системного уровня, что позволяет представлять очень сложные наборы реальных данных, например разрозненные, перекрывающиеся спутниковые сцены. Для регулярных наборов данных, помимо прочих характеристик, мы требуем, чтобы их подмассивы соответствовали определенным критериям, которые могут быть использованы для разработки эффективных алгоритмов. Например, подмассивы могут накладываться друг на друга, но в соответствии с определенным шаблоном. Сырые наборы данных могут быть усвоены (преобразованы) в регулярные наборы данных.

Формально, *сырой набор данных* $\mathbb{D}^{raw} = (A, P^{raw})$ содержит массив (тензор) *пользовательского уровня* $A(d_1, d_2, \ldots, d_N) : \mathbb{T}$ и набор массивов (подмассивов) *системного уровня* $P^{raw} = \{(A', B, E, wid)\}$, где $A' \sqsubseteq A$, $B\langle N\rangle : \text{int} = \{b_1, b_2, \ldots, b_N\}$, $E\langle N\rangle : \text{int} = \{e_1, e_2, \ldots, e_N\}$ так, что $A' = A[b_1 : e_1, b_2 : e_2, \ldots, b_N : e_N]$, а wid является идентификатором узла кластера, хранящего A'.

Регулярный набор данных $\mathbb{D} = (A, P, S, \rho, r^0)$ содержит массив (тензор) *пользовательского уровня* $A(d_1, d_2, \ldots, d_N) : \mathbb{T}$, множество массивов системного уровня (подмассивов) $P = \{(A', B, E, wid, key)\}$, где A', B, E, wid означают то же самое, что и для P^{raw}, $key\langle N\rangle : \text{int} = \{k_1, k_2, \ldots, k_N\}$, $k_i \in \mathbb{Z}$, $A' \sqsubseteq A[h_1^b : h_1^e, \ldots, h_N^b : h_N^e]$, где

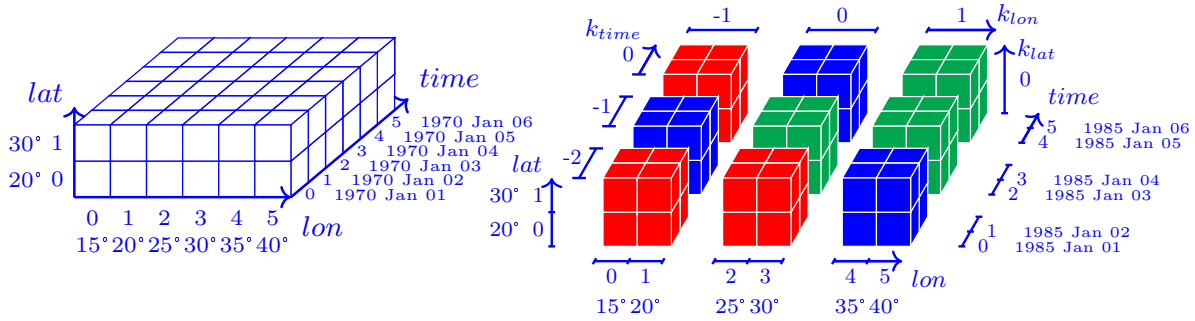
$$h_i^b = max(r_i^0 + k_i \times s_i - \rho_i, 0), \tag{2.5a}$$

$$h_i^e = min(r_i^0 + (k_i + 1) \times s_i - 1 + \rho_i, |A.d_i| - 1), \tag{2.5b}$$

$S = (s_1, s_2, \ldots, s_N)$ является наибольшей возможной формой для $\forall p \in P$, $\rho = (\rho_1, \rho_2, \ldots, \rho_N)$ является величиной перекрытия подмассивов, а $r^0 = (r_1^0, r_2^0, \ldots, r_N^0)$ референсными индексами, где $s_i, r_i^0 \in \mathbb{Z}$, $s_i > 0$, $\rho_i \in [0, s_i \text{ div } 2) \subset \mathbb{Z}$ и $\nexists p, q \in P : p.key = q.key \wedge p = q$. Отметим, что A является общим для $\mathbb{D}^{raw}$ и $\mathbb{D}$. Будем ссылаться на подмассив A' по ключу посредством $\mathbb{D}\langle\langle key\rangle\rangle$ либо $\mathbb{D}\langle\langle k_1, k_2, \ldots, k_N\rangle\rangle$.

Регулярный набор данных показан на рис. 2.2. Массив (тензор) $A(time, lat, lon)$ с формой $6 \times 2 \times 6$ делится 2-мерными плоскостями на 9 подмассивов (это очень часто встречается на практике), где $S = (2, 2, 2)$, $r^0 = (4, 0, 2)$, перекрытие не показано. Подмассивы с одинаковым цветом находятся на одном узле кластера. Значения координатных массивов показаны рядом с каждым индексом. Ключ $\langle\langle -1, 0, 1 \rangle\rangle$ ссылается на подмассив $A[2{:}3, 0{:}1, 4{:}5]$ для Jan 03-04, $lat = 20° \ldots 30°$, $lon = 35° \ldots 40°$.

Это означает, что массив (тензор) A разделен $(N-1)$-мерными гиперплоскостями на N-мерные подпространства. Массив системного уровня может не полностью покрывать подпространство, в котором он находится. Обратите внимание, что не все подпространства должны содержать объект. Подмассивы могут перекрываться, но только перекрывающиеся ячейки могут находиться в одном и том же подпространстве. Все ключи подмассивов уникальны. Мы рассматриваем пустое подпространство в виде подмассива, все ячейки которого равны `NA`. Это один из способов, с помощью которого наша модель поддерживает большие разреженные массивы.



(a) Пользовательский массив (b) Подмассивы (системный уровень) [54]

Рис. 2.2: Массивы уровня пользователя и системного уровня

В модели есть и третий уровень: чанки. Это позволяет системам, основанным на данной модели, стать еще более гибкими в плане адаптации к динамическим рабочим нагрузкам благодаря эффективному выполнению операции под названием “чанкинг”, раздел. 2.2.3 и 4.1.1.

Обратите внимание, что мы опускаем части модели, которые не используем в последующих разделах диссертации. Полное формальное определение нашей модели приведено в [54].

2.2 Новые распределенные тензорные алгоритмы

Модель, представленная в разд. 2.1 позволяет формально определить тензорные операции (операторы) и разработать распределенные алгоритмы и эффективные подходы их выполнения на компьютерных кластерах, возможно, в облаке.

Мы сформировали набор операций с массивами (тензорами) для реализации, проанализировав их важность с точки зрения обеспечения основных и расширенных возможностей по управлению и обработке массивов (тензоров). Мы приводим обоснование каждой операции. Другими словами, выбранные операции являются строительными блоками, которые могут быть использованы для конструирования сложных аналитических конвейеров, см. пример конвейера SAVI (разд. 3.1.3) с его планом выполнения (раздел 3.4 из [54]), а также другие приложения, основанные на этих операциях, глав. 4.

Одно из основных отличий наших новых алгоритмов от остальных алгоритмов состоит в том, что они способны управлять тензорами и обрабатывать их на месте (in situ), непосредственно в разнообразных форматах файлов: подмассивы могут храниться в виде обычных файлов на узлах кластера без импорта во внутренний формат СУБД. Это дает множество преимуществ, в том числе возможность частично делегировать операции с массивами (тензорами) существующему проработанному и качественному программному обеспечению (см. Введение из [54]), используя таким образом промышленный опыт. Однако разработка эффективной системы, работающей с данными на месте (in situ) и использующей подход делегирования "выходит далеко за рамки вызова отдельных экземпляров одной и той же функции на каждом узле", что говорится о ChronosDB в [66].

Псевдокоды алгоритмов достаточно велики [51, 54, 57, 60, 64], поэтому мы приводим здесь формальные определения операций и ключевые идеи, лежащие в основе алгоритмов. Псевдокоды основаны на нашей новой формальной модели данных (разд. 2.1) и содержат строки, выделенные светло-серым цветом, чтобы указать на использование промышленного опыта, помимо прочих целей. Алгоритмы могут выполнять широкий спектр распределенных операций очень эффективно: до 1034 раз быстрее SciDB (разработана компанией Paradigm4 и М. Стоунбрейкером, лауреатом премии А. Тьюринга), разд. 4.1.1.

2.2.1 Распределенный N-мерный ретайлинг

Распределенный N-мерный ретайлинг является одной из наиболее важных тензорных операций, которая лежит в основе многих других жизненно важных функций, являющихся ключевыми для ТСУБД. Например, ретайлинг может быть использован для "приготовления" регулярного набора данных из сырого набора данных, для трансформации регулярного набора данных, а также для K-путевых соединений, что ставит его в ряд фундаментальных операций для массивов (тензоров) [54].

В качестве входных данных N-мерный ретайлинг принимает сырой или регулярный набор данных $\mathbb{D} = (A, P)$ и параметры ретайлинга $S' = (s'_1, s'_2, \ldots, s'_N)$, $\rho' = (\rho'_1, \rho'_2, \ldots, \rho'_N)$, $\bar{r}^0 = (\bar{r}^0_1, \bar{r}^0_2, \ldots, \bar{r}^0_N)$: целевую форму, размер перекрытия и референсные индексы соответственно. В качестве выходных данных ретайлинг генерирует новый регулярный набор данных $\mathbb{D}' = (A, P', S', \rho', \bar{r}^0)$. Заметим, что A является общим для $\mathbb{D}$ и $\mathbb{D}'$.

Ключевая идея предлагаемого алгоритма заключается в следующем. Мы разрезаем каждый подмассив системного уровня $p \in P$ на более мелкие фрагменты $P' = \{p' : p' \sqsubseteq p\}$ (под-под-массивы) и присваиваем каждой части новый ключ. Затем мы объединяем все части с одинаковым ключом в один, новый тензор системного уровня (новый подмассив в новом, выходном наборе данных). Каждый фрагмент, используя свой ключ, ассоциируется с узлом кластера. Таким образом, все фрагменты с одинаковым ключом собираются и объединяются на одном узле в нашем распределенном алгоритме, который построен таким образом, что может делегировать вырезку фрагмента и слияние (сборку множества фрагментов в подмассив) внешнему программному обеспечению [54].

Рисунок 2.3 иллюстрирует алгоритм ретайлинга на 2-мерном случае. Рассмотрим регулярный набор данных $\mathbb{D} = (A, P, S, \rho, r^0)$, где $A(lat, lon)$ есть 2-мерный массив (матрица), $S = (5, 6)$, $\rho = (0, 0)$, $r^0 = (5, 6)$. A имеет форму 10×18. Кроме того, A состоит из 6 подмассивов тела которых разделены толстыми синими линия-

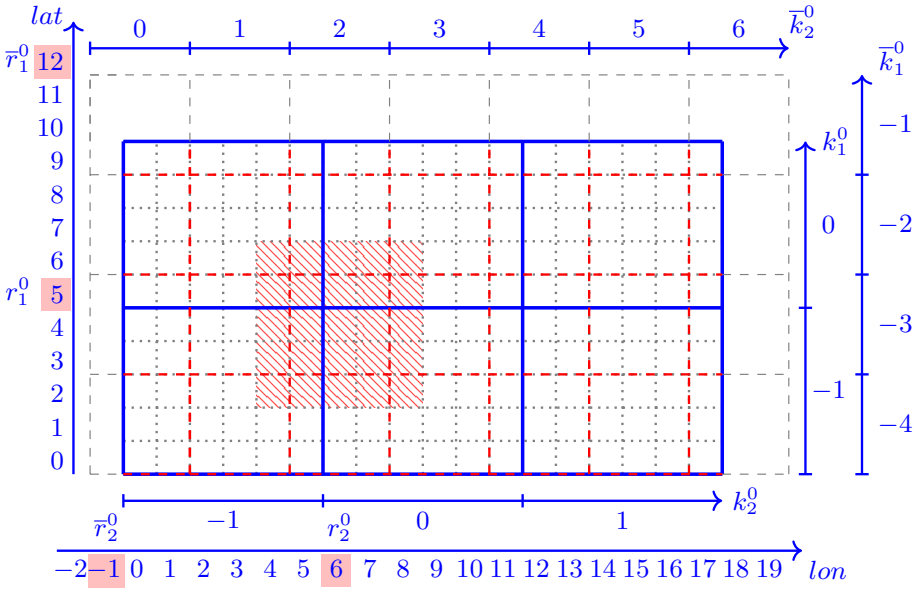


Рис. 2.3: Пример ретайлинга [54]

ми. Подмассивы могут располагаться на разных узлах кластера. Серые пунктирные линии разделяют ячейки матрицы. Диапазон ключей для подмассивов составляет $k_1^0 \in [-1, 0]$, $k_2^0 \in [-1, 1]$.

Мы выполняем ретайлинг набора данных $\mathbb{D}$ в $\mathbb{D}' = (A, P', S', \rho', \bar{r}^0)$, где $S' = (3, 3)$, $\rho' = (1, 1)$, $\bar{r}^0 = (12, -1)$. Пунктирные красные линии разделяют тела новых подмассивов. В результате работы алгоритма будет получено 28 новых подмассивов с диапазоном ключей $\bar{k}_1^0 \in [-4, -1]$, $\bar{k}_2^0 \in [0, 6]$. Например, заштрихованная область обозначает подмассив $\mathbb{D}'\langle\langle -3, 2\rangle\rangle = A[2:6, 4:8]$. Отметим, что не все новые массивы системного уровня будут иметь форму 5×5 ($5 = s'_1 + \rho'_1 \times 2$). Следовательно, некоторые подмассивы не будут полностью покрывать соответствующие им подпространства, например $\mathbb{D}'\langle\langle -1, 6\rangle\rangle = A[8:9, 16:17]$. Референсные координаты r_i^0 и $\bar{r}_i^0$ выделены розовым цветом на осях *lat* и *lon*. Мы изобразили входные и выходные ключи подмассивов на осях k_i^0 и $\bar{k}_i^0$ соответственно.

2.2.2 Распределенное K-путевое соединение

Для $K > 1$, многие K-арные тензорные операции требуют K-путевого соединения [51, 54, 57, 60, 64]. Например, алгебра массивов (карт) широко используется в промышленности и составляет большую часть рабочих нагрузок ТСУБД [52, 66]. Она может потребовать K-путевого соединения массивов (тензоров) для двух и более входных массивов (тензоров). Распределенное K-путевое соединение массивов (тензоров) основано на распределенном N-мерном ретайлинге, разд. 2.2.1.

Следующее формальное определение K-путевого соединения массивов (тензоров) приводится из [54].

K-путевое соединение $\bowtie$: $\odot, \kappa, A_1, A_2, \ldots, A_K \mapsto A_\bowtie$ принимает на вход тип соединения $\odot \in \{\text{INNER}, \text{OUTER}\}$, отображение $\kappa : \mathbb{T}_1, \mathbb{T}_2, \ldots, \mathbb{T}_K \mapsto \mathbb{T}_\bowtie$, и N-мерные массивы $A_j(d_1^j, d_2^j, \ldots, d_N^j) : \mathbb{T}_j$, $j \in [1, K]$ такие, что $\exists d_i^\bowtie : d_i^j \sqsubseteq d_i^\bowtie$ для $\forall i, j$.

K-путевое соединение производит N-мерный массив $A_\bowtie(d_1^\bowtie, d_2^\bowtie, \ldots, d_N^\bowtie) : \mathbb{T}_\bowtie$ такой, что $A_\bowtie[x_1, x_2, \ldots, x_N] = \kappa'(a_1, a_2, \ldots, a_K)$, где $x_i \in [0, |d_i^\bowtie|)$, $a_j = A_j[y_1^j, y_2^j, \ldots, y_N^j]$, если $\exists y_i^j : d_i^j[y_i^j] = d_i^\bowtie[x_i]$; $a_j = \texttt{NA}$ иначе.

Операция κ' формально определяет разницу между двумя типами соединения. Если $a_j = \texttt{NA}$ для $\forall j$, то κ' возвращает NA независимо от значения $\odot$. Если $\odot = \text{INNER}$ и $\exists j : a_j = \texttt{NA}$, κ' возвращает NA. В противном случае κ' возвращает $\kappa(a_1, a_2, \ldots, a_K)$. Для алгебраических вычислений, где любая операция над набором ячеек должна возвращать NA, если хотя бы одно из значений ячейки отсутствует, полезно INNER-соединение. Когда хотя бы одно не отсутствующее значение вносит вклад в результирующую ячейку, например, K-дневное безоблачное композитное спутниковое изображение, полезно использовать OUTER-соединение [54].

Распределенное 2-путевое соединение массивов (тензоров) иллюстрирует рисунок 2(e) [54]. Он показывает подробную информацию об этапах 2-мерного ретайлинга и 2-мерного соединения, а также значения ключей, назначенных каждой сущности, участвующей в плане выполнения (кортежи содержат ключи подмассивов).

Среди прочего, такое соединение необходимо в составе сложного аналитического конвейера, который использует алгебру массивов (карт), разд. 3.1.3. Распределенное 2-путевое соединение массивов (тензоров) выполняется на массивах Band4 и Band5 с различными формами подмассивов. Band5 требует распределенного 2-мерного ретайлинга во время соединения (в этом случае $K = N$). Алгоритм построен таким образом, что может делегировать вычисление κ на подмассивах с одним и тем же ключом внешнему оптимизированному программному обеспечению [54].

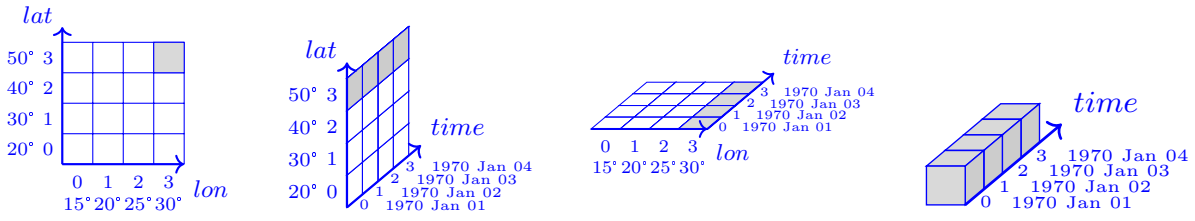
2.2.3 Агрегация, чанкинг и другие операции

Агрегация

Агрегация часто необходима при анализе данных реальных массивов. Например, для 4-мерного массива $A(time, elevation, lat, lon)$, агрегированные массивы $A'(time, lat, lon)$ и $A''(time, elevation, lat)$ могут представлять собой средние значения по всем доступным вертикальным уровням и зональные средние значения соответственно.

Агрегация массивов может происходить по произвольному подмножеству осей массива [60]. Агрегация N-мерного массива $A(d_1, d_2, \ldots, d_N) : \mathbb{T}$ по множеству осей $d_{aggr} \subset \{d_1, d_2, \ldots, d_N\}$ является таким K-мерным массивом $A_{aggr}(d_{y_1}, \ldots, d_{y_K}) : \mathbb{T}$, что $K = N - |d_{aggr}|$, $y_k < y_{k+1}$, $y_k \in [1, N]$, $d_{y_k} \notin d_{aggr}$, $A_{aggr}[x_{y_1}, \ldots, x_{y_K}] = f_{aggr}(cells(A[g(1), g(2), \ldots, g(N)]))$, где $\forall x_{y_k} \in [0, |d_{y_k}|)$, $g(i) = 0 : |d_i| - 1$, если $d_i \in d_{aggr}$, иначе $g(i) = x_i$, $cells : A' \mapsto T$ есть мультимножество всех значений ячеек массива $A' \sqsubseteq A$, $f_{aggr} : T \mapsto w \in \mathbb{T}$ является агрегирующей функцией. Заметим, что условие $K > 0$ всегда выполняется, иначе результатом будет не массив, а скаляр.

Например, мы можем агрегировать массив (тензор) на рис. 2.1 по оси $time$, оси lon, оси lat или осям lat и lon одновременно, рис. 2.4.



(a) $A_{aggr}(lat, lon)$ (b) $A_{aggr}(time, lat)$ (c) $A_{aggr}(time, lon)$ (d) $A_{aggr}(time)$

Рис. 2.4: Примеры результата агрегации для массива (тензора) на рис. 2.1 [60]

Выполнение агрегации (и других операций) в распределенном режиме сопряжено с трудностями, особенно когда подмассивы находятся в сложных форматах файлов массивов (тензоров), включая NetCDF, GeoTIFF, HDF и других форматах [60].

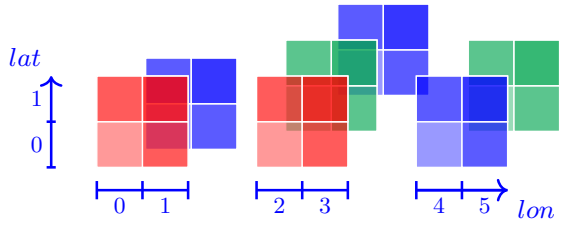


Рис. 2.5: Промежуточные агрегаты подмассивов (по оси *time*) из рис. 2.2b [60]

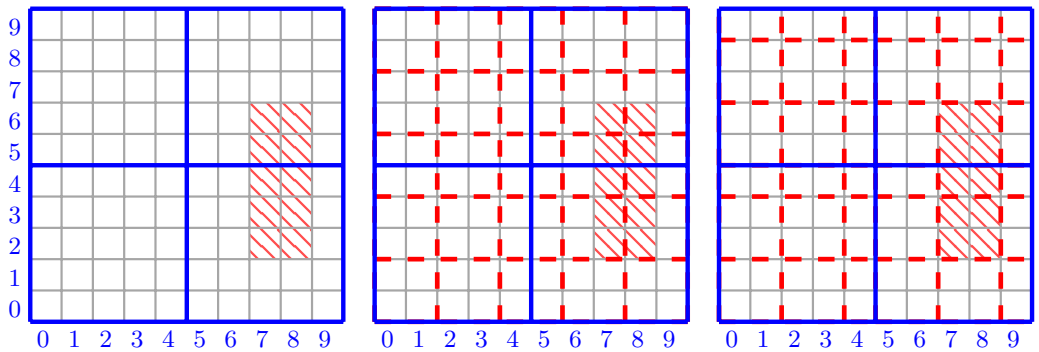
Напомним, что массивы (тензоры) системного уровня с одинаковым цветом расположены на одном узле кластера, рис. 2.2b. Алгоритм агрегации состоит из двух этапов. Вначале подмассивы параллельно агрегируются на узлах кластера в локальные промежуточные подмассивы, рис. 2.5. Вдобавок для промежуточных подмассивов генерируются промежуточные ключи. Далее промежуточные подмассивы распределяются между узлами кластера с помощью политики отображения и вычисляется конечный результат (не один массив, а набор подмассивов, распределенных по узлам кластера). Наши алгоритмы агрегации построены таким образом, что способны делегировать вычисление f_{aggr} внешнему программному обеспечению [54, 60].

Чанкинг

Чанкинг – одна из самых важных операций, поскольку она позволяет ТСУБД адаптироваться к динамическим рабочим нагрузкам. Производительность операции может различаться на порядки в зависимости от конфигурации чанков, разд. 4.1.1. Массивы (тензоры) системного уровня состоят из чанков, единиц ввода-вывода (третий уровень модели). Чанкинг изменяет компоновку хранения чанков и их форму.

Точный чанкинг реорганизует ячейки массива таким образом, что ячейки с индексами $(x_1, \ldots, x_N)$ и $(y_1, \ldots, y_N)$ принадлежат одному и тому же чанку, если x_i div $c_i = y_i$ div c_i для $\forall i$, рис. 2.6(b). Неточный чанкинг выполняет точный чанкинг массивов системного уровня [54].

Давайте прочитаем фрагмент размером 5×2 из 2-мерного массива (подмассивы разделены толстыми синими линиями), рис. 2.6. Для построчной схемы хранения,



(a) No chunking (b) Exact 2×2 chunking (c) Inexact 2×2 chunking

Рис. 2.6: Чанкинг массива (тензора) [54]

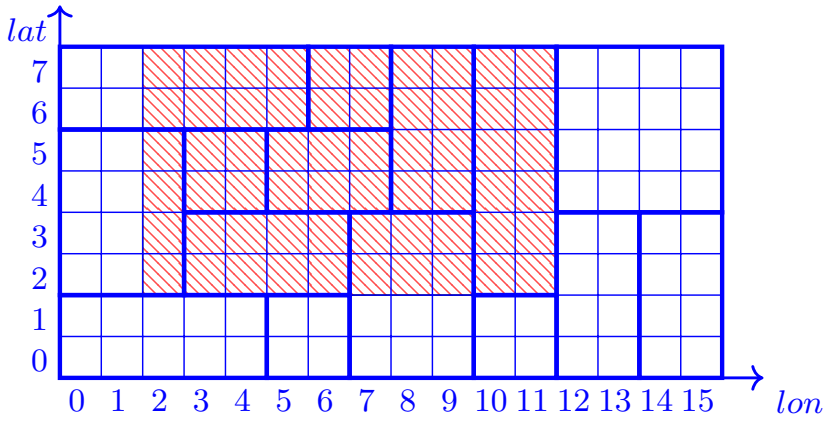


Рис. 2.7: Извлечение гиперсреза [57]

мы можем потратить 5 запросов ввода-вывода, чтобы прочитать 5 полос размером 1×2, рис. 2.6(a). Однако из массива, над которым был выполнен чанкинг считываются только чанки с нужными данными. Точный чанкинг работает внутри логического массива (тензора) пользовательского уровня, в то же время неточный чанкинг полагается на индексы системного уровня, рис. 2.6(b,c).

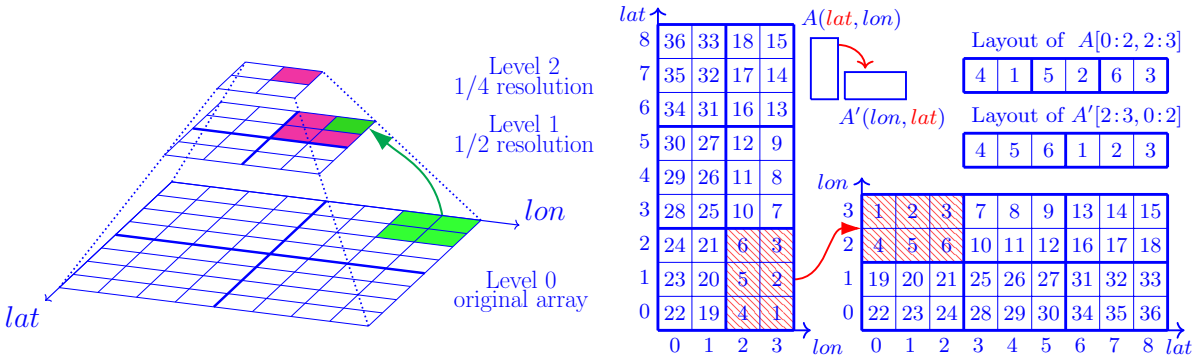
Ни одна форма чанка не является оптимальной для всех сценариев доступа. Обычно невозможно "угадать" априори хорошую форму чанка для произвольного случая: формы чанков часто подбираются экспериментально. ТСУБД должны быть способны быстро выполнять чанкинг тензоров, чтобы поддерживать такой подбор и адаптироваться к динамическим рабочим нагрузкам.

Извлечение гиперсреза

Эффективно извлечь гиперсрез далеко не просто: производительность может различаться на порядки в зависимости от формы подмассивов и компоновки их ячеек, что обычно подбирается экспериментально, методом проб и ошибок, разд. 4.1.1.

Заштрихованная область обозначает гиперсрез $A' = A[2:7, 2:11]$, рис. 2.7. A является логическим 2-мерным массивом $A(lat, lon)$ из 15 подмассивов, разделяемых толстыми линиями. Подмассивы могут располагаться на разных узлах кластера и могут принадлежать сырому или регулярному набору данных. Мы сводим извлечение гиперсреза логического массива (тензора) к извлечению гиперсрезов из соответствующих массивов (тензоров) системного уровня. Сначала мы фильтруем подмассивы, которые не пересекаются с A', например $A[0:1, 0:4]$. В новый набор данных переносятся подмассивы полностью находящиеся внутри A', например $A[4:5, 5:7]$. Для завершения операции из оставшихся подмассивов извлекаются гиперсрезы.

Алгоритм способен делегировать извлечение гиперсреза многофункциональным и высоко оптимизированным внешним программным инструментам. Подробные псевдокоды (различные версии) находятся в [54, 57]. Чанкинг и извлечение гиперсреза работают чрезвычайно эффективно в ChronosDB, разд. 4.1.1. Они могут быть на порядки быстрее, чем SciDB, даже когда SciDB соответствующим образом настроена путем предварительного использования чанкинга [54].



(a) Многоуровневая пирамида, 3 уровня [51]

(b) Решейпинг массива (тензора) [54, 57]

Рис. 2.8: Иллюстрация даунсемплинга и решейпинга в ТСУБД

Ресемплинг

Ресемплинг является основной операцией для множества приложений [50]. Даунсемплинг (укрупнение), апсемплинг и интерполяция являются типичными представителями ресемплинга. В свою очередь, многоуровневая пирамида иллюстрирует даунсемплинг. Рисунок 2.8a изображает 3 уровня такой пирамиды.

Обычно большие массивы (тензоры) визуализируются интерактивно с помощью многоуровневых пирамид. Задается ряд уровней масштабирования, например, $Z = \{0, 1, \ldots, 16\}$. Визуализатор переключает текущий уровень масштабирования, когда пользователь увеличивает/уменьшает масштаб. При уровне масштабирования $z \in Z$, тензор отображается с $2^z\times$ меньшим разрешением, чем у оригинала. Многоуровневая пирамида – это стек массивов (тензоров) для всех уровней масштабирования, рис. 2.8a. Эта техника значительно снижает сетевой трафик и нагрузку на систему, что ставит эту функциональность в разряд важнейших для ТСУБД.

Формально, если задан 2-мерный массив $A(d_1, d_2) : \mathbb{T}$, то его укрупнённой версией является массив $A'(d'_1, d'_2) : \mathbb{T}$ такой, что $d'_i\langle\lceil l_i\rceil\rangle = \{d_i[0], d_i[2], \ldots, d_i[\lfloor l_i\rfloor \times 2]\}$, где $l_i = |d_i|/2$ и $A'[x'_1, x'_2] = avg(\{A[x_1, x_2] : x_i \text{ div } 2 = x'_i \wedge A[x_1, x_2] \neq \texttt{NA}\})$ для $\forall x_i$, avg это среднее и $avg(\varnothing) = \texttt{NA}$. Мы полагаем, что в d_i хранятся координаты наименьшей граничной ячейки. В то же время, интерполяция оценивает значение в координате $(y_1, \ldots, y_N)$ для массива $A(d_1, \ldots, d_N)$, где $y_i \in (d_i[j], d_i[j + 1]) \subset \mathbb{T}_i$, и $j \in [0, |d_i| - 1) \subset \mathbb{Z}$. Например, разрешение массива может быть удвоено интерполяцией, когда $y_i = (d_i[j] + d_i[j + 1])/2$ [54].

Благодаря нашей новой модели данных и подходам работы на месте (in situ), мы поддерживаем множество методов ресемплинга [50, 54]. Однако SciDB обеспечивает только усреднение близлежащих значений массива (тензора).

Решейпинг

Операция решейпинга изменяет порядок осей тензора, рис. 2.8b. Эта операция влияет не только на индексацию ячеек тензора, она имеет гораздо более глубокие последствия с точки зрения ТСУБД. Решейпинг изменяет компоновку тензорного хранилища в соответствии с заданным порядком осей.

Например, решейпинг помогает достичь наиболее быстрого извлечения гиперсреза по заданному измерению. Например, чтение временного ряда $A[x_1, x_2, 0{:}|time| - 1]$ из 3-мерного массива $A(lat, lon, time)$ по сравнению с $A(time, lat, lon)$. Это связано

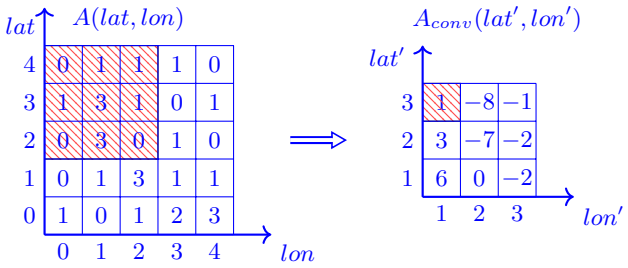


Рис. 2.9: Иллюстрация свёртки с ядром K_2

с тем, что обычно последняя размерность N-мерного массива изменяется быстрее всего: ячейки вдоль N-го измерения хранятся в памяти последовательно: см. схемы хранения A и A' на рис. 2.8b. Для построчного расположения мы можем расходовать 3 операции ввода-вывода, чтобы прочитать $A[0:2,2]$ в отличие от всего одной операции ввода-вывода для чтения тех же данных из $A'[2,0:2]$.

Формально, операция *решейпинга* $\Psi : A, \pi \mapsto A'$ принимает на вход N-мерный массив $A(d_1, \dots, d_N) : \mathbb{T}$ и перестановочное отображение $\pi : i \mapsto j$, где $i, j \in [1, N] \subset \mathbb{Z}$, $\pi(i) \neq \pi(j)$ для $i \neq j$ и $\bigcup_i \{\pi(i)\} = [1, N]$. Операция решейпинга выдает N-мерный массив $A'(d_{\pi(1)}, \dots, d_{\pi(N)}) : \mathbb{T}$ такой, что $A[x_1, \dots, x_N] = A'[x_{\pi(1)}, \dots, x_{\pi(N)}]$, где $x_i \in [0, |d_i|) \subset \mathbb{Z}$ для всех i.

Решейпинг массива пользовательского уровня может быт сведен к решейпингу массивов системного уровня (подмассивов) по отдельности путём делегирования решейпинга подмассивов внешним инструментам.

Свёртка

Операция свертки $\Xi : A, K \mapsto A_{conv}$ для 2-мерного массива $A(d_1, d_2) : \mathbb{T}$ и ядра $K\langle k_1, k_2\rangle : \mathbb{T}$, где $k_i \leqslant |d_i|$, и $k_i \bmod 2 = 1$ для всех i, порождает массив 2-d $A_{conv}(d'_1, d'_2) : \mathbb{T}$ такой, что $d'_i = d_i[k_i \operatorname{div} 2 : |d_i| - k_i \operatorname{div} 2 - 1]$ и

$$A_{conv}[x_1, x_2] = \sum_{\substack{\forall x'_1 \in [0, k_1) \\ \forall x'_2 \in [0, k_2)}} K[x'_1, x'_2] \times A'[x'_1, x'_2] \tag{2.6}$$

где $A' = A[x_1 - k_1/2 : x_1 + k_1/2, x_2 - k_2/2 : x_2 + k_2/2]$ и $k_i/2 \leqslant x_i < |A.d_i| - k_i/2$ для всех i (деление "/" является целочисленным) [64].

Свертка часто используется для обработки массивов [50]. Например, обнаружение границ с помощью оператора Собеля $K_1\langle 3, 3\rangle = \{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, 2, 1\}\}$ и $K_2\langle 3, 3\rangle = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, 0, 1\}\}$ происходит следующим образом. Сначала вычисляются локальные градиенты в каждой ячейке массива вдоль осей d_1 и d_2: $A_{d1} = \Xi(A, K_1)$ и $A_{d2} = \Xi(A, K_2)$, рис. 2.9. Затем, массив $A_{edges} = \sqrt{A_{d1}^2 + A_{d2}^2}$ будет содержать более высокие значения для ячеек, классифицированных границами и меньшие значения для других типов ячеек [64].

Можно выполнить свертку для каждого 2-мерного $p \in P$ используя внешнее программное обеспечение, при условии, что ретайлинг был применен к входному набору данных с $\rho_i \geqslant 1$ для $\forall i$. Таким образом, свертка может быть эффективно применена к каждому массиву (подмассиву) системного уровня параллельно без обмена данными между узлами кластера, поскольку подмассивы перекрываются.

Мы также представляем еще одну новую операцию (оператор) свертки в разд. 3.3.1.

Мозаика массивов (тензоров)

Пожалуйста, обратитесь к разделам 2.5, 3.5 и 4.4 для определения операции мозаики массивов (тензоров) и описания соответствующих подходов.

2.3 Настраиваемые запросы

2.3.1 Новое направление R&D: настраиваемые запросы

Десятки параметризованных математических функций ежедневно применяются к тензорам для решения жизненно важных практических задач, включая городское планирование, мониторинг сельского хозяйства, контроль лесного хозяйства и быстрого реагирования при ликвидации последствий стихийных бедствий [4, 78], рис. 2.10.

$$\text{SAVI} = \frac{\text{NIR} - \text{R}}{\text{NIR} + \text{R} + \boxed{L}} \times (1 + \boxed{L}) \quad \text{ARVI} = \frac{\text{NIR} - (\text{R} - \boxed{\gamma}(\text{B} - \text{R}))}{\text{NIR} + (\text{R} - \boxed{\gamma}(\text{B} - \text{R}))} \quad \text{PVI} = \begin{matrix} cos(\boxed{@}) \times \text{NIR} \\ -sin(\boxed{@}) \times \text{R} \end{matrix}$$

$$\text{GARI}^{\dagger} = \frac{\text{NIR} - [\text{G} - \boxed{\gamma}(\text{B} - \text{R})]}{\text{NIR} + [\text{G} - \boxed{\gamma}(\text{B} - \text{R})]} \quad \text{TSAVI} = \frac{\boxed{@}(\text{NIR} - \boxed{@}\text{R} - \text{B})}{\text{R} + \boxed{@}\text{NIR} - \boxed{@}\boxed{b}} \quad \text{TVI} = \sqrt{\frac{\text{NIR} - \text{R}}{\text{NIR} + \text{R}} + 0.5\boxed{L}}$$

$$\text{AVI} = tan^{-1}\left[\frac{\boxed{\lambda_3} - \boxed{\lambda_2}}{\boxed{\lambda_2}} \frac{1}{(\text{NIR} - \text{R})}\right] + tan^{-1}\left[\frac{\boxed{\lambda_2} - \boxed{\lambda_1}}{\boxed{\lambda_2}} \frac{1}{(\text{G} - \text{R})}\right] \quad \text{WDRVI} = \frac{\boxed{@}\text{NIR} - \text{R}}{\boxed{@}\text{NIR} + \text{R}}$$

Рис. 2.10: Примеры типичных функций с **настраиваемыми параметрами** [53]

Рассмотрим типичный пример – индекс растительности с поправкой на почву (SAVI, рис. 2.10), направленный на минимизацию влияния яркости почвы. NIR и R – это 2-мерные массивы с интенсивностью отраженного солнечного излучения в ближнем инфракрасном и видимом красном спектрах соответственно. L – коэффициент поправки на почву, $L \in [0, 1]$ (зависит от почвы) [78]. Пользователь может менять L много раз чтобы найти подходящие значения SAVI для конкретной области интереса.

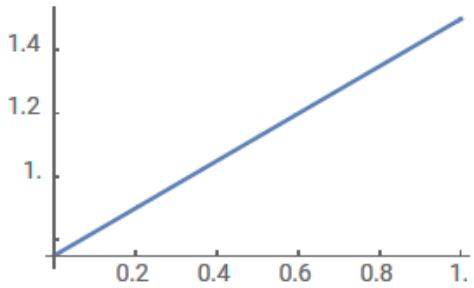
BitFun работает с важным классом запросов, который ранее явно не рассматривался в контексте ТСУБД: настраиваемые запросы. BitFun явно фокусируется на факте настраиваемости [53]. К моменту создания BitFun, современные ТСУБД не были оснащены механизмами индексирования ячеек тензоров: SciDB [15], TileDB [46], RasDaMan [48], PostGIS [47], и Oracle Spatial [43]. Подходы к индексированию массивов не рассматривали в явном виде настраиваемые сценарии [11, 76].

До появления BitFun, современные подходы ТСУБД рассматривали два последующих запроса вычисления SAVI с немного разными значениями L (например, 0.7 и 0.8) в качестве двух разных запросов. Это приводило к вычислению SAVI для нового значения L заново, несмотря на то, что обычно только небольшая часть результирующего тензора (массива) будет существенно отличаться от предыдущего результата, приводя к излишнему расходу ресурсов [53].

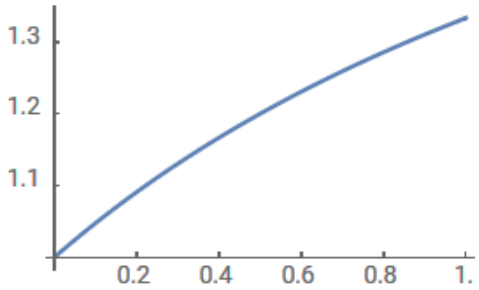
2.3.2 Новые подходы индексирования функций

Мы разработали новые подходы индексирования для трех типов настраиваемых запросов: (1) вычисление значений $f(\tau)$, (2) классификация $f(\tau)$, (3) оценка неравенства $f(\tau) < const$, где τ – настраиваемый параметр и $f(\tau)$ – дифференцируемая,

возможно, нелинейная функция [53]. В качестве примера, кратко опишем основные идеи (1) на примере SAVI.



(a) NIR = 700, R = 100



(b) NIR = 2, R = 0

Рис. 2.11: SAVI значения для всего диапазона $L \in [0, 1]$

Давайте зафиксируем значения NIR и R. Теперь SAVI становится функцией только одной переменной. Если мы посмотрим еще внимательнее, то заметим, что в подавляющем большинстве случаев SAVI – это почти прямая для большинства комбинаций значений NIR и R, рис. 2.11. В общем случае SAVI не является аффинной функцией. Существуют значения NIR и R, для которых SAVI демонстрирует слегка нелинейное поведение, рис. 2.11b. Однако даже в таких случаях наши методы могут обеспечить эффективное индексирование с хорошей точностью.

Мы можем аппроксимировать SAVI с помощью значительно более простого выражения, линейной зависимости: $aL+b$, где $a, b \in \mathbb{R}$. Чтобы построить его, нам нужны две пары значений L и SAVI, чтобы вычислить a и b. Поскольку в СУБД индексирование чередуется с ответом на запрос (вычислением $f(\tau)$), мы уже можем иметь одну из пар для каждой ячейки массива. Вторую пару мы можем получить, выбрав некоторое L в его диапазоне значений. Однако не любое L может быть использовано для вычисления a и b, поскольку потенциальная ошибка аппроксимации может превысить заданную пользователем допустимую точность.

Решение $[\text{SAVI}(L) - (aL+b)]'dL = 0$ дает максимальную ошибку. После решения и его упрощения мы получаем несколько корней: $L_{1,2} = \pm(\sqrt{a(\text{NIR} - \text{R})(\text{NIR} + \text{R} - 1)} \mp a(\text{NIR} + \text{R}))/a$. Далее мы можем использовать $L_{1,2}$ для вычисления потенциальной ошибки и принять дальнейшие решения по индексированию [53].

Теперь у нас есть набор пар a, b для каждой ячейки. Далее мы присваиваем уникальный $\texttt{ID} = (\bar{a}, \bar{b})$ для $\forall \overline{f} = aL+b$ так, что $\bar{a}, \bar{b} = a\times, b\times 10^{|\lg(\Upsilon)|}$, где Υ – заданная пользователем точность. Далее, наша новая структура данных индексирует {(ID, Freq)}, где Freq – частота встречаемости ID, разд. 2.3.3.

2.3.3 Новая и быстрая иерархическая структура данных

Наша новая иерархическая структура данных может индексировать объекты {(ID, Freq)}, определенные в разд. 2.3.2. Структура данных быстро создается и читается [53], позволяя отвечать на настраиваемые запросы до 8 раз быстрее по сравнению с предыдущими подходами, разд. 4.2.

Пусть $E = \{e_1, e_2, \ldots, e_n\}$ является множеством объектов (например, функций), $A\langle l_1, l_2\rangle : E$ есть 2-мерный массив (тензор), который нужно проиндексировать (нотация массива/тензора определена в разд. 2.1.2), $n \leqslant |A|$ и $\text{Freq}(e_j)$ – количество e_j в A, где $j \in [1, n]$. Уровень i (L_i) есть 1-мерный массив, в котором каждая ячейка

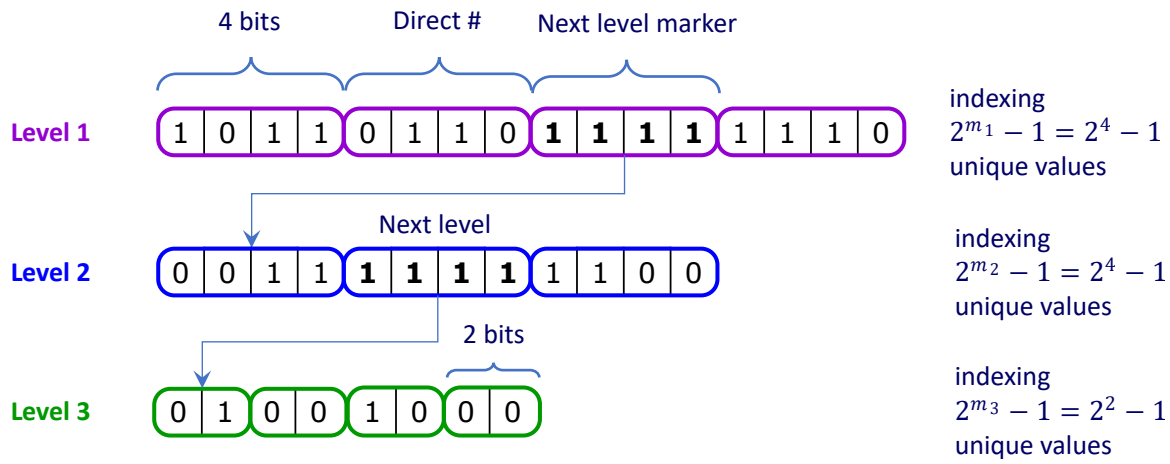


Рис. 2.12: Пример битового индекса [53]

представляет собой код фиксированной длины из m_i бит, где $m_i \leqslant \lceil \log_2 |A| \rceil$. Индекс является иерархическим и имеет не более K уровней, где $K \in \mathbb{N}$, рис. 2.12.

Например, структура на рис. 2.12 индексирует A следующим образом. Первые 4 бита 1011 на уровне 1 отображаются на e_{12} ($1011_2 = 11_{10}$) при условии, что $\text{Freq}(e_j) \geqslant \text{Freq}(e_{j+1})$. Биты 1111 переводят нас на уровень 2 чтобы узнать индекс объекта ($1111_2 = 2^{m_1} - 1$). Это первая такая комбинация на уровне 1, поэтому мы должны получить первые m_2 битов 0011 с уровня 2, чтобы продолжить. Эти биты индексируют возможно менее частый объект e_{16} ($2^{m_1} - 2 + 1 = 15$).

2.4 Новое направление R&D: моделирование полностью внутри ТСУБД

2.4.1 Обоснование, недостатки и преимущества

Представим новое направление R&D в области ТСУБД, которое открывает широкие возможности для дальнейших исследований [56, 61]. ТСУБД управляют хранением больших многомерных массивов/тензоров, их обработкой, а в некоторых случаях даже визуализацией и машинным обучением. Моделирование физического мира традиционно осуществляется на различных типах сеток, которые могут быть представлены в виде многомерных массивов (тензоров). Поскольку многомерные массивы лежат в основе ТСУБД, логично применить ТСУБД к моделированию, чтобы воспользоваться преимуществами ТСУБД.

Различные сетки, полосы и многие другие типы данных хранятся в компьютерных системах во время моделирования или обмена данными. Многие из таких типов данных, даже экзотические, могут быть представлены в известных тензорных моделях данных и конвенциях (подробные примеры приведены в [6, 39]).

С точки зрения ТСУБД, моделирование открывает широкий спектр новых возможностей R&D и расширяют область применения ТСУБД. С точки зрения модельера, ТСУБД может обеспечить множество преимуществ и управление данными моделирования в стиле СУБД. Мы использовали нашу модель данных, разд. 2.1 и приступили к интеграции моделирования в ТСУБД начиная с клеточных автоматов

(КА). Мы впервые внедрили возможности моделирования в ТСУБД с использованием клеточных автоматов дорожного движения (ТСА) на SIGMOD 2021 [56].

В настоящее время применение ТСУБД (SimDB, разд. 3.3) для моделирования с помощью КА имеет некоторые недостатки. На сегодняшний день SimDB вносит накладные расходы на планирование моделирования КА, а пользователям приходится писать 2 типа UDF (User Defined Functions): на Java и на новом, но относительно простом языке. Мы считаем, что эти недостатки очень незначительны, и в будущем усилия R&D позволят смягчить или даже устранить их. Таким образом, многочисленные преимущества перевешивают несколько недостатков в отношении использования SimDB.

Мы показали, что ТСУБД SimDB можно использовать для сквозного моделирования КА, начиная инициализацией и оканчивая вычислением итоговой статистики. Оказывается, SimDB отлично подходит для этой рабочей нагрузки. Это сразу демонстрирует многочисленные преимущества использования ТСУБД, и SimDB в частности, для моделирования КА:

- **Усвоение и слияние данных**. КА может использовать различные наборы данных в качестве входных, поэтому SimDB может оперативно подготавливать (например, усваивать, выполнять ресемплинг и фрагментацию) данных для моделирования.
- **Автоматическое распараллеливание**. SimDB выполняет моделирование параллельно, управляя всеми необходимыми обменами данными между узлами кластера в Облаке.
- **Отладка UDFs**. Легко отладить шаг за шагом UDF, используемые SimDB, подготовленные для модели КА.
- **Интерактивная визуализация** необходима для понимания данных, поэтому SimDB предоставляет изображения сетки КА посредством открытого, популярного протокола OGC WMTS [75].
- **Управление данными**. Можно архивировать, выполнять запросы и сравнивать данные моделирования с помощью SimDB.
- **Интероперабельность**. Слой хранения SimDB построен поверх сырых файлов в стандартных форматах. Массивы моделирования – полноценные файлы GeoTIFF с географической привязкой, легко доступные для другого программного обеспечения.
- **Сквозное моделирование**. SimDB обслуживает все этапы моделирования в рамках единой системы, даже вычисление статистики (цель моделирования).

2.4.2 Новый клеточный автомат дорожного движения

Для обеспечения возможности моделирования в ТСУБД, мы начали с моделирования дорожного движения: оно очень важно на практике и используются для планирования дорожных изменений, оптимизации работы светофоров, анализа пропускной способности и интеграции объектов, и это только некоторые примеры [3]. Модели дорожного движения КА также могут быть очень сложными для моделирования.

Опираясь на литературные данные (хороший обзор моделей дорожного движения КА есть в [32]), мы исследовали, какие объекты (например, транспортные средства, светофоры, перекрестки) могут быть представлены клеточным автоматом. Мы

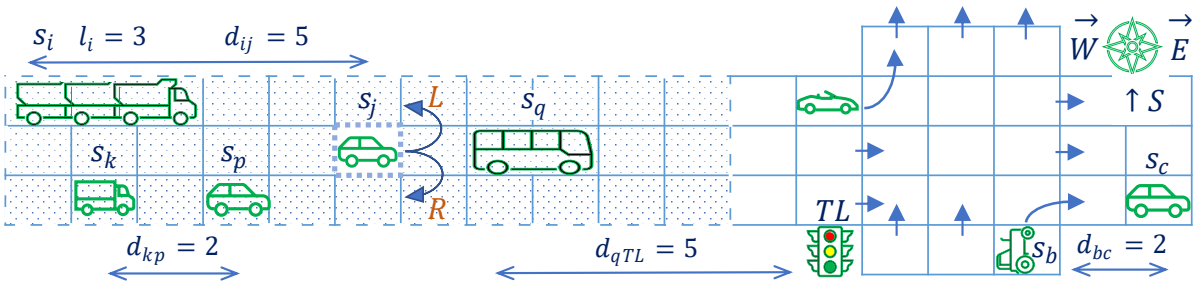


Рис. 2.13: Клеточный автомат для моделирования дорожного движения [56]

разработали новый клеточный автомат дорожного движения (TCA), который достаточно сложен для того, чтобы выявить и поставить новые проблемы перед ТСУБД, разд. 2.4.3. Решение поставленных проблем повысит эффективность моделирования внутри ТСУБД, поможет им стать более гибкими системами в целом, а также предоставит исследователям и практикам преимущества в области моделирования, характерные для ТСУБД, разд. 2.4.1.

Наша модель КА состоит из 4 традиционных компонентов: физическое окружение, состояния и окрестности ячеек, локальные правила перехода [56], рис. 2.13.

В свою очередь, физическая среда нашей модели представляет всю дорожную сеть в виде 2-мерной сетки (2-мерного массива). Отличительные особенности физической среды нашей модели перечислены ниже и основываются на рис. 2.13.

- Сетка имеет различные типы ячеек, например, транспортные средства и непроходимые участки.
- Мы поддерживаем несколько светофоров, расположенных по всей сетке (TL).
- Мы учитываем транспортные средства различной длины (обозначаемые l_*).
- Транспортные средства могут иметь различные направления движения и скорости (обозначаются s_*).
- Модель содержит дорожные перекрестки (RI), регулируемые светофорами.
- Транспортные средства могут менять направление движения на RI (см. s_b).
- Каждая дорога может состоять из нескольких полос; транспортным средствам разрешено менять полосы движения, например, для того, чтобы обогнать более медленное транспортное средство (см. s_j).
- Можно расположить транспортные средства на любой ячейке дороги.

Состояния и окрестности клеток имеют следующие особенности.

- Ячейки меняют свои состояния через дискретные временные шаги (итерации).
- Все ячейки в определенном окне составляют её локальную окрестность, включая саму ячейку, 5 ячеек назад и вперед (см. s_j).
- Локальные правила перехода применяются ко всем клеткам одновременно (водители принимают решения независимо друг от друга, параллельно, но подчиняясь некоторым общепринятым правилам).

Наконец, большой и разнообразный набор локальных правил перехода отвечает за корректную эволюцию автомата во времени. Локальные правила перехода в нашей модели можно разделить на следующие категории.

- Правила движения вперед учитывают длину транспортного средства и включают в себя правила ускорения, торможения, рандомизации и движения, которые позволяют избежать столкновения транспортных средств и учитывают индивидуальное поведение водителя.
- Правила смены полосы движения позволяют сменить полосу на левую либо правую. Они проверяют наличие ограничений на межавтомобильное расстояние (например, d_{ij}) и скорость автомобиля.
- Правила светофора помогают избежать столкновений на перекрестках. Светофоры могут гореть красным, желтым и зеленым цветом. Транспортные средства могут стоять в очереди перед светофором.
- Правила пересечения дорог регулируют поведение транспортных средств на перекрестках. Транспортные средства проверяют наличие различных ограничений и могут поворачивать налево/направо (см. d_{bc}).

2.4.3 Проблемы и новые вспомогательные компоненты

Может показаться, что ТСУБД могут легко выполнять моделирование КА, поскольку у них общая модель данных. Однако такое моделирование ставит перед ТСУБД сложные задачи проектирования. Мы решили множество проблем, связанных с итеративными рабочими нагрузками, правилами переходов КА, параллельным выполнением и другими проблемами, чтобы сделать возможным проведение моделирования КА в ТСУБД. Мы кратко излагаем основные проблемы ниже.

Design Challenge 1. *Каким образом поддерживать произвольные локальные правила переходов клеточных автоматов в ТСУБД?*

Правило КА похоже на хорошо известный оператор свертки [64]. Однако правило КА гораздо сложнее: это процедура, которая (1) применяется для каждой ячейки нескольких входных массивов, (2) проверяет наличие ограничений, (3) может обновлять несколько ячеек в окрестности. Для поддержки произвольных локальных правил КА, мы представили новый оператор свертки для ТСУБД, разд. 3.3.1.

Обратите внимание, что мы не можем просто расширить нашу модель ТСУБД для хранения кортежей в ячейке, поскольку это приведет к потере совместимости с промышленным программным обеспечением, популярными стандартизированными форматами файлов и возможностью работы на месте (in situ). Некоторые форматы поддерживают записи, но они сложны и редко используются на практике.

Design Challenge 2. *Каким образом эффективно (нативно) поддерживать итерации непосредственно внутри ТСУБД?*

Итерации присущи моделированию физического мира. Python/C++ UDFs (User Defined Functions) поддерживаются некоторыми ТСУБД, поэтому пользователь может программировать итерации на этих языках. Однако такие UDFs являются “черными ящиками”, которые ТСУБД не могут оптимизировать [36]. Существующие языки запросов не могут выразить итерации. Выполнение итераций запрос за запросом требует полной материализации результата запроса перед следующим запросом. Кроме того, целостные оптимизации на несколько итераций вперед недоступны, поскольку общая картина неясна.

Чтобы решить эту проблему, мы представили первый нативный язык ТСУБД для пользовательских функций (UDFs), разд. 3.3.2.

Design Challenge 3. *Каким образом правильно и эффективно выполнить нативную пользовательскую функцию (UDF) для ТСУБД?*

Хотя UDFs, написанные с помощью нашего нового языка, выглядят относительно небольшими, они сложны в исполнении. Мы представили множество новых компонентов ТСУБД, включая проактивные планы моделирования, механизмы версионирования и блокировок, чтобы решить эту проблему, разд. 3.3.3.

2.5 Науки о данных: масштабируем подходы

Машинное обучение (ML) и наука о данных (DS) только прокладывают свой путь к ТСУБД [52, 72]. Хотя интеграция методов ML и DS в ТСУБД сталкивается с множеством проблем, она также имеет многочисленные преимущества. Одним из таких преимуществ является отсутствие необходимости в трудоемком перемещении данных между ТСУБД и системами ML/DS для запуска алгоритмов ML/DS на данных, находящихся под управлением ТСУБД. Пример важного практического приложения – быстрое создание высококачественной бесшовной мозаики.

2.5.1 Проблемы создания мозаики из массивов (тензоров)

Оператор MOSAIC, встречающийся в ТСУБД, выполняет усвоение коллекции перекрывающихся тензоров в один большой тензор, называемый мозаикой. Оператор может применять сложные методы ML/ DS для создания высококачественной бесшовной мозаики, в которой видимые контрасты между значениями ячеек, взятых из входных перекрывающихся тензоров, устранены насколько возможно.

Например, ChronosDB имеет команду MOSAIC [54], а Oracle Spatial предоставляет процедуру `SDO_GEOR_AGGR.mosaicSubset` [42]. В свою очередь, RasDaMan оснащена рецептом MOSAIC [49].

Высококачественные методы мозаики основаны на сложных подходах. Например, IR-MAD (Iteratively Re-weighted MAD), где MAD означает Multivariate Alteration Detection, использует канонический корреляционный анализ (CCA) [29]. CCA – это популярный подход для определения корреляций в многомерных наборах данных [23]. CCA широко используется в Data Science для уменьшения размерности и обнаружения латентных переменных.

Однако проблемы с производительностью становятся узким местом при применении таких методов к все более растущим объемам тензоров. В последних исследованиях скорость была названа ключевой проблемой и аспектом улучшения методов мозаики нового поколения [29, 31].

Для примера рассмотрим следующий случай использования. Один тензор, напр., спутниковый снимок, часто не может полностью охватить область интереса. В этом случае оператор мозаики, оснащенный методами Data Science, используется для объединения большой коллекции входных тензоров в единую бесшовную мозаику – большой многомерный массив, в котором видимость швов между отдельными входными массивами уменьшена насколько возможно. Для получения высококачественных мозаик алгоритмы прибегают к сложным приёмам, упомянутым выше. Рисунки 4.15 и 4.16 иллюстрирует входные сцены и полученную мозаику.

2.5.2 Масштабируем MAD & IR-MAD

FastMosaic улучшает IR-MAD (используя новый, масштабируемый подход к CCA), поскольку он может работать практически без ручного взаимодействия, что является важным свойством для приложений Big Data. IR-MAD также может создавать высококачественные мозаики тензоров, даже для сложных распределений значений перекрывающихся ячеек входных тензоров [29].

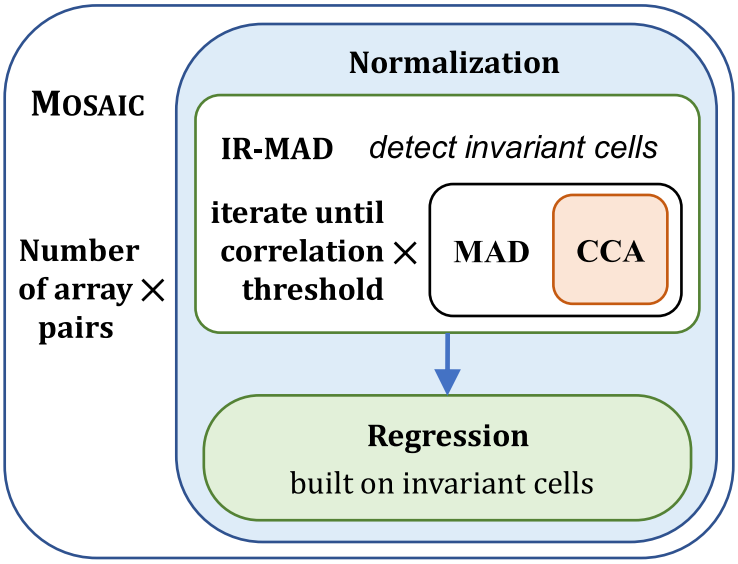


Рис. 2.14: Структура FastMosaic

Рисунок 2.14 представляет структуру FastMosaic. В качестве входных данных FastMosaic принимает набор перекрывающихся массивов (тензоров) и выполняет их попарное слияние. Мы рассматриваем M пар ячеек из двух перекрывающихся массивов (тензоров) в качестве пары случайных величин X и Y размерности K (X и Y представляют массив-субъект и опорный массив соответственно): $X_k = \{X_{k,1}, X_{k,2}, \ldots, X_{k,M}\}$, где $X_{k,j}$ – значение ячейки $j \in [1, M]$ и спектрального канала $k \in [1, K]$. Аналогично определяем Y [59].

Для каждой пары ячеек MAD оценивает вероятность отсутствия изменений P_j с помощью нашего нового, масштабируемого подхода к выполнению CCA (разд. 2.5.3) на взвешенных $X_k w$ и $Y_k w$, где $w = \{w_1, w_2, \ldots, w_M\}$. IR-MAD выполняет итерации до тех пор, пока w существенно изменяется. Ячейки с $P_j > \Theta \in [0.95, 0.99]$ называются инвариантными. Относительная нормализация строит ортогональную регрессию на инвариантных ячейках $Y_k = \beta_k X_k + \epsilon_k$, где $\beta_k, \epsilon_k \in \mathbb{R}$, чтобы получить K пар коэффициентов трансформации β_k, ϵ_k для применения ко всем ячейкам массива-субъекта. Наконец, трансформированный массив (тензор) объединяется с опорным; этот увеличенный массив (тензор) заменяет пару во входном наборе. Данные шаги повторяются до тех пор, пока в наборе не появится только один массив (тензор) – результирующая мозаика.

Раздел 4.4 демонстрирует пошаговое построение мозаики на реальных геопространственных массивах (тензорах).

2.5.3 Масштабируем Canonical Correlation Analysis

Вспомним, что CCA максимизирует корреляцию $corr(a^T X, b^T Y)$, подбирая коэффициенты a и b. Случайные переменные $U = a^T X, V = b^T Y$ называются парой канонических переменных. Мы разработали формулы для вычисления U, V вместе с коэффициентами нормализации IR-MAD β_k, ϵ_k (разд. 2.5.2) за тот же проход по входным данным с линейным временем выполнения за $O(M \times K^2 + K^3)$, рис. 2.15.

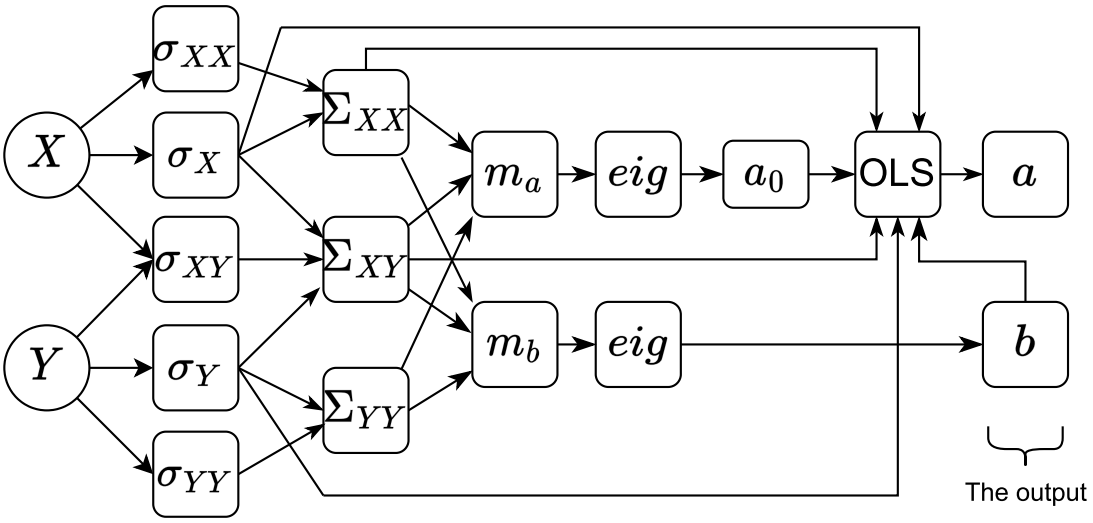


Рис. 2.15: Граф вычислений CCA: предложенный подход [59]

Идея заключается в сборе определенной статистики (занимает более 95% общего времени работы на реальных данных) для вычисления U, V и β_k, ϵ_k (5% общего времени работы), все за тот же проход. Это позволяет оснастить ТСУБД масштабируемым и высококачественным оператором MOSAIC. Сначала мы вычисляем $\sigma_{X,k} = \sum_j^M X_{k,j} w_j$, взвешенную сумму ячеек для канала k: $\sigma_X = \{\sigma_{X_1}, \sigma_{X_2}, \ldots, \sigma_{X_K}\}$. Затем вычисляем σ_{XY}, матрицу взвешенных сумм произведений X и Y: $\sigma_{XY} = X_k^T (Y_k \odot w)$, $\sigma_{XX} = X_k^T (X_k \odot w)$, $\sigma_{YY} = Y_k^T (Y_k \odot w)$, где $\odot$ – поэлементное произведение. Кроме того, мы вычисляем Σ_{XY}, Σ_{XX} и Σ_{YY}, матрицы взвешенных ковариаций (формулы для Σ_{XX} и Σ_{YY} аналогичны):

$$\Sigma_{XY} = \frac{\sigma_{XY}}{\sum w - 1} - \frac{\sigma_X \sigma_Y^T}{\sum w (\sum w - 1)}$$

Используя эту статистику, мы вычисляем матрицы m_a, m_b, векторы a_0, b как собственные векторы матриц m_a, m_b, получаем a и $M = \{M_1, M_2, \ldots, M_K\}$, где $M_k = ((U_k - V_k) - \overline{M_k})/std(M_k)$, чтобы оценить $P(no\ change) = \chi^2_{cdf}(\sum M_k^2)$ и получить β_k, ϵ_k за тот же проход. Полный набор формул приведен в [59].

Глава 3

Программное обеспечение: архитектурные и реализационные аспекты

В контексте R&D недостаточно предложить только теоретические результаты: необходимо также иметь должные архитектуры и реализации, представленные в этой главе, чтобы обеспечить и усилить алгоритмический эффект и конкурировать с популярными программными системами. Новые аспекты обеспечивают управление тензорами в стиле СУБД и эффективно организуют наши тензорные алгоритмы и подходы, позволяя добиться значительного ускорения.

Результаты этой главы – новые архитектурные и реализационные аспекты систем ChronosDB (разд. 3.1), BitFun (разд. 3.2), SimDB (разд. 3.3), WebArrayDB (разд. 3.4.2), ArrayGIS (разд. 3.4.3) и FastMosaic (разд. 3.5). Глава 4 представляет дополнительные аспекты.

ChronosDB превосходит SciDB **в среднем в 75 раз**. Она всегда быстрее и может превзойти SciDB **до 1024 раз**. SciDB разработана компанией Paradigm4 под руководством М. Стоунбрейкера, лауреата премии А. Тьюринга ("Нобелевская премия по компьютерным наукам").

BitFun оснащен новыми стратегиями индексирования тензоров при запросах с похожими математическими функциями. Он может быть **до 8 раз быстрее**, чем вычисление результатов с нуля.

SimDB – первая ТСУБД с полным циклом моделирования полностью внутри себя: от подготовки данных до моделирования и вычисления статистики; **сравнима по скорости с написанным вручную кодом**.

WebArrayDB – первая ТСУБД, работающая полностью в Web-браузере. При этом ArrayGIS – новая Web-ГИС на основе WebArrayDB. Вместе они могут быть **более 2 раз быстрее**, чем запросы только к Sentinel-Hub, популярному облачному сервису для доставки данных Sentinel.

3.1 ChronosDB: инновационная ТСУБД

3.1.1 Архитектура и компоненты ChronosDB

Рисунок 3.1 представляет архитектуру ChronosDB с некоторыми ее ключевыми компонентами и особенностями.

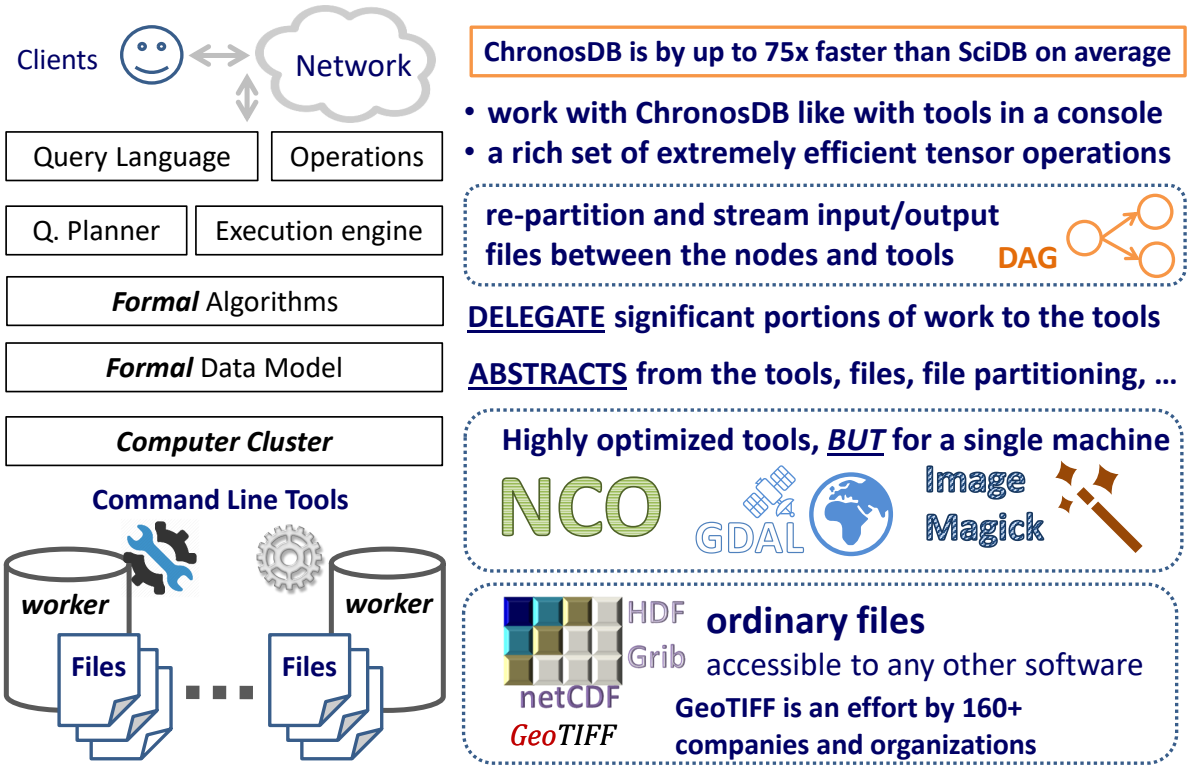


Рис. 3.1: Архитектура ChronosDB

Формальная модель данных абстрагируется от файлов и рассматривает взаимосвязанные файлы в виде одного большого тензора, разд. 2.1. Файлы могут быть распределены между узлами кластера и обрабатываться параллельно. Это позволяет ChronosDB работать на компьютерном кластере из обычного оборудования, возможно в облаке, и основывать свой уровень хранения на устоявшихся, стандартизированных форматах файлов [54].

Более того, в отличие от других СУБД, ChronosDB работает на месте (in situ, непосредственно с различными форматами файлов, например NetCDF и GeoTIFF) и не требует длительного этапа импорта во внутренний формат СУБД. Такой подход обеспечивает мощные возможности хранения данных, включая эффективную поддержку различных типов данных, отсутствующих значений и многочисленных методов сжатия “из коробки”. Вдобавок массивы ChronosDB легко доступны для любого другого программного обеспечения в виде набора обычных файлов. Например, для географических информационных систем (ГИС), R, Python и многого другого программного обеспечения, которое может читать такие файлы.

Модель также способствует разработке формальных алгоритмов управления, обработки и визуализации тензоров, разд. 2.2. Они предназначены для работы на месте (in situ) и делегируют значительную часть работы проработанным и оптимизированным инструментам командной строки, разд. 4.1.1. Новые архитектурные и реализационные аспекты позволяют алгоритмам, реализованным в ChronosDB, работать с исключительной эффективностью [54].

В частности, ChronosDB имеет развитый планировщик запросов и модуль выполнения. Осуществляется перекомпоновка и стриминг входных/выходных файлов между узлами и инструментами для масштабирования обработки. Для этого стро-

ятся планы выполнения, разд. 3.1.3. ChronosDB использует многоядерные процессоры и может получить преимущества от многодисковых узлов кластера.

Наконец, с точки зрения пользователя, работать с ChronosDB легко даже новичку, поскольку он предоставляет язык запросов, напоминающий синтаксис известных инструментов командной строки, разд. 3.2 [54].

3.1.2 Новые подходы управления тензорами

Давайте сформируем набор данных в качестве рабочего примера: 24937×38673 массивы R и NIR, мозаика из набора 4×8 сцен Landsat 8, каналы 4 (видимый красный, R) и 5 (ближний инфракрасный, NIR), пути 191–198, строки 24–27, 01–15 июля 2015 г., GeoTIFF. Landsat – самая продолжительная, непрерывная программа по наблюдению за Землёй из космоса с 1972 г. по настоящее время [27]. Amazon и Google предоставляют сцены Landsat через коммерческие облака из-за её популярности [18].

В отличие от всех существующих ТСУБД, ChronosDB поддерживает иерархическое пространство имен наборов данных, чтобы облегчить навигацию в большом количестве наборов данных. Например, полное имя набора данных может выглядеть таким образом: `Landsat8.Level_1.SurfaceReflectance.Band4` (точки разделяют имена коллекций наборов данных, имя набора данных – последнее).

Массивы (тензоры) системного уровня (подмассивы, файлы различных форматов) могут быть размещены на узлах кластера ручным копированием или с помощью специальной команды ChronosDB. В отличие от параллельных или распределенных файловых систем, ChronosDB всегда хранит файл полностью на одном узле. Можно реплицировать подмассив на несколько узлов для обеспечения отказоустойчивости и балансировки нагрузки.

Рабочие узлы должны обнаруживать свои наборы данных при запуске. Они получают иерархию наборов данных от узла-координатора и сканируют свои локальные системы хранения, чтобы вернуть свойства набора данных. Они получают эту информацию путем разбора имен файлов подмассивов или чтения метаданных файлов.

ChronosDB выполняет усвоение данных перед запросом к заданному набору данных (множеству файлов определенного формата). Существующие ТСУБД конвертируют файлы (или требуют использования сторонних инструментов для преобразования файлов) во внутренний формат СУБД, а ChronosDB работает с файлами на месте (in situ), в их исходных форматах. ChronosDB устраняет трудоемкую и подверженную ошибкам фазу преобразования форматов которая может длиться дольше, чем типовой запрос к данным. ChronosDB делает некоторые предположения о заданных файлах, например, одинаковый набор ключей метаданных.

ТСУБД также могут выводить схему массива (тензора). ChronosDB схема Band4 из нашего набора данных взята из [54] и представлена ниже.

```
gdalinfo Landsat8.Level_1.SurfaceReflectance.Band4
Driver: GTiff/GeoTIFF
Size is 38673, 24937
Coordinate System is:
PROJCS["WGS 84 / UTM zone 32N", ...skipped... AUTHORITY["EPSG","32632"]]
Origin = (-53110.000000000000000,5878570.000000000000000)
Pixel Size = (30.000000000000000,-30.000000000000000)
Metadata:
  AREA_OR_POINT=Area
Corner Coordinates:
Upper Left  (  -53110.000, 5878570.000) (  0d47'37.30"E, 52d46'20.14"N)
Lower Left  (  -53110.000, 5130460.000) (  1d50'34.16"E, 46d 6'11.15"N)
Upper Right ( 1107080.000, 5878570.000) ( 17d59'48.38"E, 52d42'52.21"N)
Lower Right ( 1107080.000, 5130460.000) ( 16d50'57.11"E, 46d 3'26.67"N)
```

```
Center      (  526985.000, 5504515.000) (  9d22'26.95"E, 49d41'33.18"N)
Subarray=2048x2048 Block=2048x1 Type=UInt16, ColorInterp=Gray
  NoData Value=0
```

Значения полей в примере выше те же, что и для известной утилиты `gdalinfo`, обрабатывающей только один файл за раз. Поскольку `gdalinfo` довольно популярна, многие пользователи избегают изучения нового вида схемы массива, работая с тензорами ChronosDB привычным для них способом, в отличие от других ТСУБД.

Управление метаданными в ChronosDB происходит посредством команд, которые отражают функциональность соответствующих инструментов командной строки. Например, ChronosDB предоставляет возможности `ncatted` (attribute editor, NCO) для добавления, создания, удаления, изменения и перезаписи внутренних метаданных файла [41]. Однако, в отличие от инструмента `ncatted`, команда `ncatted` применяется к потенциально большому количеству файлов одновременно.

3.1.3 Новые и эффективные подходы выполнения запросов

Мы иллюстрируем наши новые подходы эффективного выполнения запросов на примере сложного аналитического конвейера, разд. 3.4 [54]. Входные данные – массивы R и NIR с различными формами подмассивов, во избежание их совмещения на узлах кластера с целью вызвать распределенный ретайлинг, включающий сетевой обмен (разд. 2.2.1):

1. Интерполировать в 2 раза Band4 $\mapsto$ Warp4
2. Интерполировать в 2 раза Band5 $\mapsto$ Warp5
3. Выполнить 2-путевое соединение Warp4 и Warp5 $\mapsto$ SAVI
4. Выполнить даунсемплинг в 64 раза SAVI $\mapsto$ SAVIoutlook

Сначала выполняется интерполяция в 2 раза (разд. 2.2.3) и выходные данные объединяются (разд. 2.2.2) для вычисления SAVI (разд. 2.3). Наконец, строится небольшой массив (тензор) 1209×780 "быстрой оценки" (меньше результата объединения в 64 раза) для визуальной оценки результата (разд. 2.2.3). Назовём этот сценарий конвейером SAVI.

ChronosDB строит планы выполнения и работает в соответствии с этими планами. Они используются для определения порядка создания подмассивов и их обмена между рабочими узлами. Построение таких планов возможно благодаря формальной модели данных и формальным алгоритмам, которые имеют четко определенные входы/выходы. Команды ChronosDB не могут быть вложенными, но выходной набор данных может быть входным для других команд. Этот понятный и простой подход не является ограничением и не снижает производительность: весь выход команды не материализуется до запуска следующей команды.

Формально, план выполнения – ориентированный ациклический граф $G = (V, E)$, где $E = \{((\mathbb{D}_i, key_i^j), (\mathbb{D}', key_m)) : \mathbb{D}_i\langle\langle key_i^j \rangle\rangle$ требуется для вычисления $\mathbb{D}'\langle\langle key_m \rangle\rangle\}$, где $\mathbb{D}_i$ – входной, а $\mathbb{D}'$ – выходной набор данных соответственно (формально определены в разд. 2.1). Планы неизменяемы во время выполнения, что облегчает анализ потока данных. Пример плана выполнения приведен в разделе 3.4 [54]. Промежуточные наборы данных, напр. Warp4, -5 и SAVI подлежат различным оптимизациям. Их можно повторно использовать в сценарии, но нельзя регистрировать в качестве постоянных наборов. Для контроля размера выходных наборов и балансировки нагрузки применяются эвристики, разд. 3.4 [54].

3.2 BitFun: настраиваемый запрос, быстрый ответ

BitFun – компонент ChronosDB [54, 55], предоставляющий новые методы индексирования для запросов с настраиваемыми математическими функциями и новую, эффективную иерархическую структуру битмап индексов для поддержки настраиваемого индексирования. BitFun применим к ряду важных приложений (разд. 4.2), обладает специализированной архитектурой, новой рабочей процедурой индексирования и специально разработанным пользовательским интерфейсом (разд. 3.2.2).

3.2.1 Архитектура BitFun

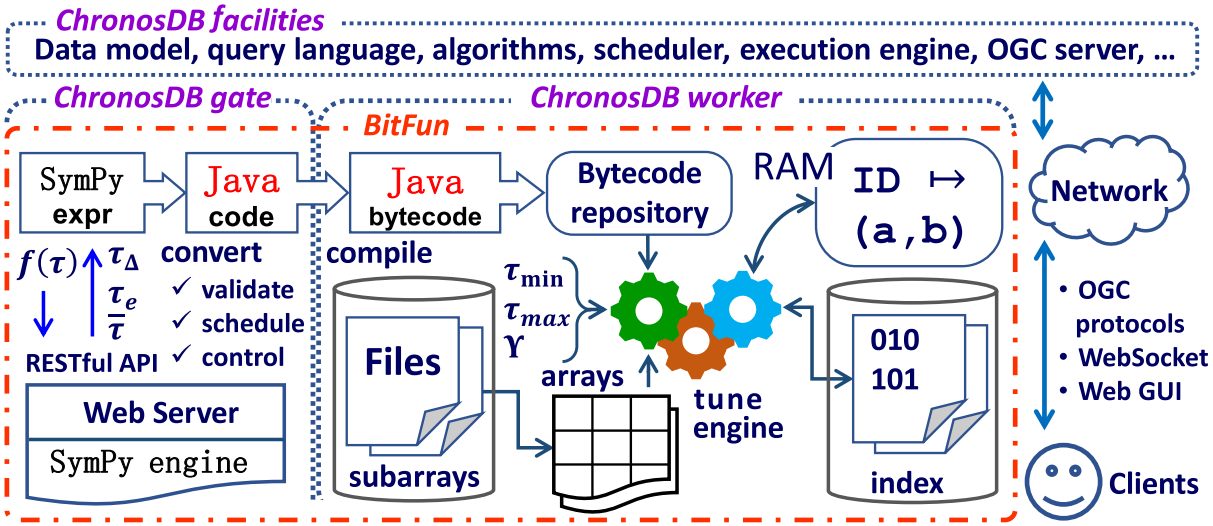


Рис. 3.2: Архитектура BitFun [53]

Поскольку BitFun является компонентом ChronosDB, его основной язык – Java, но он также использует Python для своего Web-сервера. В Java отсутствуют библиотеки символьных вычислений. Поэтому BitFun использует SymPy, чтобы находить производные, решать уравнения, упрощать выражения и выполнять другие необходимые действия в соответствии с подходом BitFun, разд. 2.3.2.

Jython и подобные инструменты ограничены по функциональности по сравнению с SymPy. Поэтому в качестве составной части BitFun мы разработали Web-сервер на Python, чтобы отправлять формулы, связанные с ними параметры и получать результаты SymPy с помощью разработанного нами RESTful API.

BitFun вычисляет символьные выражения сотни миллионов раз. Следовательно, BitFun транслирует вывод SymPy в код Java и запускает компилятор для генерации Java байткода. Этот код служит частью разнообразных процедур индексирования и вскоре компилируется в машинный код. Таким образом, нативный код выполняется гораздо быстрее, чем вычисление выражений в символьном виде. BitFun-архитектура изображена на рисунке рис. 3.2.

3.2.2 Интерактивный пользовательский интерфейс

Web-интерфейс предназначен для того, чтобы (a) продемонстрировать производительность BitFun в ответах на запросы, (b) дать возможность пользователям исследовать новую структуру индекса и изучить особенности рабочего процесса индекси-

рования, (c) предоставить возможность расширить знания пользователей о ТСУБД и о практических приложениях настраиваемых запросов.

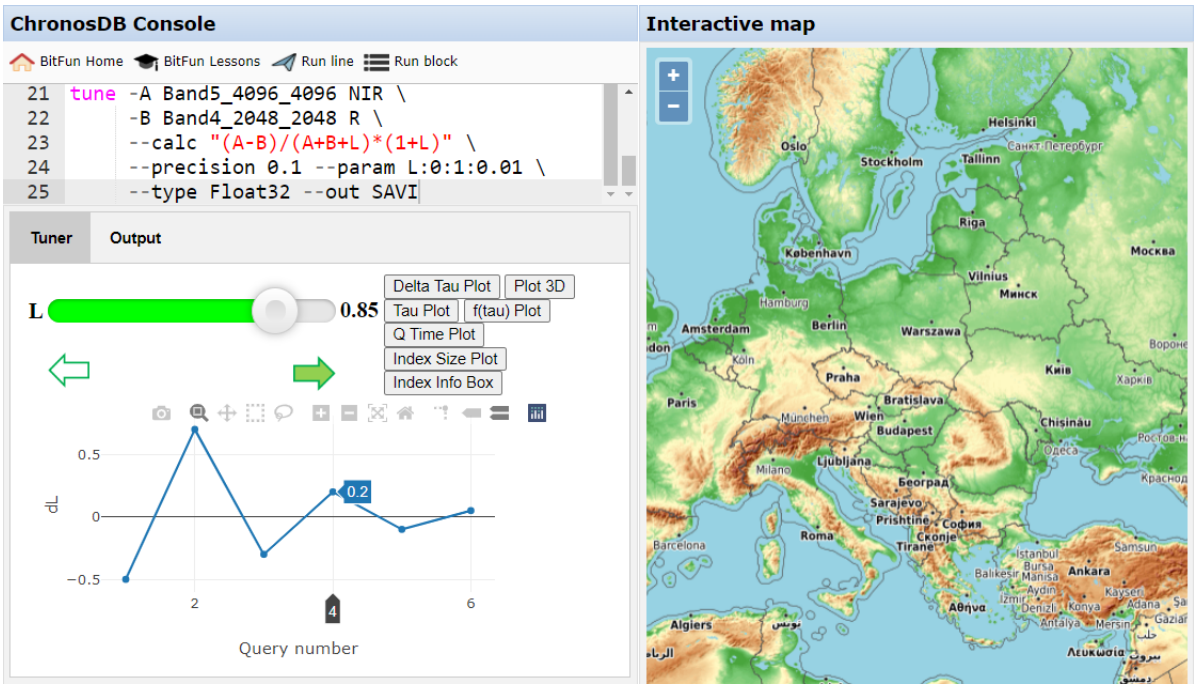


Рис. 3.3: BitFun Web GUI с его некоторыми компонентами

Web-интерфейс состоит из (1) редактора с подсветкой синтаксиса для создания и ввода запросов; (2) 2-мерных и 3-мерных диаграмм, показывающих свойства индекса, процесс индексирования и особенности ответа на запрос; визуальных компонент для (3) помощи в настройке выражений и (4) обеспечения руководства к уроку; (5) интерактивной карты с результатами запроса и исходными данными.

Кроме того, графический интерфейс предоставляет увлекательные уроки на основе реальных приложений. Чтобы добиться успеха, пользователь должен найти правильное значение параметра математической функции, экспериментально настроив его. Каждое новое значение параметра вызывает рендеринг новой интерактивной карты по результатам повторного применения функции к входному тензору.

BitFun может быть до 8 раз быстрее, чем вычисление результатов с нуля. Должный вид карты получается в результате использования правильного значения параметра. Цель – проиллюстрировать быстрые вычисления благодаря новым методам индексирования. Во время урока пользователь получает интерактивные подсказки.

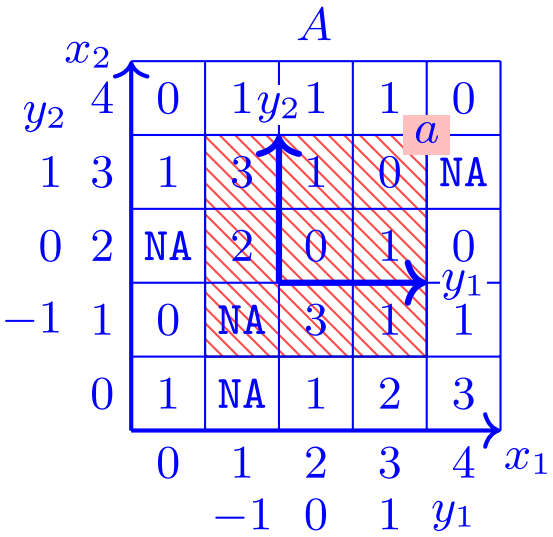
Дополнительная информация об уроках и компонентах интерфейса, например, слайдер настройки, подсказки по настройке, графики, информационное окно индекса, интерактивная карта и прочее находится в [53].

3.3 SimDB: моделирование полностью внутри ТСУБД

SimDB решает ключевые проблемы проектирования (разд. 2.4.3) и впервые позволяет проводить полный производственный цикл моделирования клеточных автоматов полностью внутри ТСУБД посредством новых компонентов ТСУБД [56, 61].

3.3.1 Новый оператор свёртки для ТСУБД

Для поддержки произвольных моделирований с помощью клеточных автоматов (КА) мы ввели новый оператор свертки для ТСУБД, рис. 3.4.




```
interface ConvolveWindow<T> {
   enum Degrees
     {_0, _90, _180, _270}

   int getArrayX();      // x1
   int getArrayY();      // x2
   int xWindowSize();    // k1
   int yWindowSize();    // k2
   int xSubarraySize(); // l1
   int ySubarraySize(); // l2
   T rotate(Degrees degrees);
   Random random();
   void move(int x, int y);
}
```

Рис. 3.4: Иллюстрация нового оператора свертки и его интерфейса [56, 61]

Давайте формально определим оператор свертки Ξ, используя нашу новую модель данных ТСУБД, разд. 2.1. Оператор $\Xi : K, A_1, A_2, \ldots, A_n \mapsto B_1, B_2, \ldots, B_m$ принимает n входных и выдает m выходных 2-мерных массивов, каждый из которых имеет форму $l_1 \times l_2$. Ядро $K\langle k_1, k_2\rangle$ есть отображение n входных (окна A_j, рис. 3.4) на m выходных 2-мерных массивов $K : a_1^x, a_2^x, \ldots, a_n^x \mapsto b_1^x, b_2^x, \ldots, b_m^x$, каждый из которых имеет форму $2k_1 + 1 \times 2k_2 + 1$, индексируется $(y_1, y_2) : y_i \in [-k_i, +k_i] \subset \mathbb{Z}$, $a_j^x[y_1, y_2] = A_j[x_1 + y_1, x_2 + y_2]$, $j \in [1, n]$, $a_j^x[y_1, y_2] = \texttt{NA}$ если $x_i + y_i \notin D_i$, $k_i \leqslant |l_i|$ div 2, $B_q[u_1, u_2] \in \{b_q^x[y_1, y_2] : x_i + y_i = u_i\}$ (выбираем последнее полученное значение), где $q \in [1, m]$, $x = (x_1, x_2)$, $x_i, u_i \in D_i$ [56].

Пользователи предоставляют логику в виде UDF на языке высокого уровня, в настоящее время на Java. SIMDB итерирует по массивам, формирует окна чтения/записи, оснащенные вспомогательными функциями, рис. 3.4. В отличие от традиционной свертки, наш оператор передает UDF свертки несколько входных окон и позволяет UDF изменять произвольное количество ячеек в нескольких выходных окнах. Это позволяет оператору создавать несколько выходных массивов [56, 61].

3.3.2 Первый нативный язык UDF для ТСУБД

Для эффективной и нативной поддержки моделирования непосредственно и полностью внутри ТСУБД, решая проблему № 2, мы представили первый нативный язык UDF для ТСУБД [56, 61], рис. 3.5.

UDF SIMDB просты в написании: они состоят из команд, синтаксис которых похож на синтаксис инструментов командной строки, знакомый большинству пользователей. Фрагмент UDF для моделирования ТСА представлен на рис. 3.5. Моделирование дорожного движения состоит из следующих шагов, повторяющихся в цикле несколько раз:

Каждое транспортное средство

1. Продвигается на несколько шагов вперёд

```
cp tca.speed $speed ↲ [newline] cp tca.length $length
val {window} "-xWindowSize 11 -yWindowSize 11"
foreach {step} -from 0 -to 100 ↲ begin
   calc tca.lane:in $speed:in $length:in tca.temperature:in \
        --overwrite $speed_move:out $length_move:out     \
        -ot Int16 {window} -classfile "TCA.java"         \
        -method_name moveForwardPhase
   ...
   calc tca.lane:in $speed_lights:in $length_lights:in \
        --overwrite $speed:out $length:out           \
        -ot Int16 {window} -classfile "TCA.java"     \
        -method_name lightsPhase
   append $speed speedh ↲ append $length lenh ↲ end
```

Рис. 3.5: Фрагмент нативной UDF ТСУБД для моделирования TCA [61]

2. Решает, повернуть ли налево, и, возможно, делает поворот
3. Решает, поворачивать ли направо, и, возможно, делает поворот

Затем

4. Светофоры изменяют свой цвет
5. Таймер продвигается вперед

Для сохранения входных массивов, `tca.speed` и `tca.length` копируются в промежуточные массивы `$speed` и `$length`, управление которыми подлежит оптимизации.

Команда `calc` запускает оператор свертки, представленный в виде Java UDF, разд. 3.3.1. SimDB компилирует Java UDF в байткод для более быстрого выполнения. `calc` принимает/выдает произвольное количество входных/выходных массивов. Параметр `-ot` задает $\mathbb{T}$, разд. 2.1. Квантили `:in` и `:out` различают входные/выходные массивы, поскольку их количество не фиксировано. Итерация заканчивается добавлением новых 2-мерных массивов `$speed` и `$length` к 3-мерным массивам `speedh` и `lenh` вдоль виртуальной оси *time*.

Виртуальные оси также были введены в нашей работе для того, чтобы сделать возможным и эффективным моделирование полностью внутри ТСУБД [56].

3.3.3 Планирование, управление версиями и блокировки

Хотя UDF выглядит небольшой (рис. 3.5, её выполнение является сложной задачей. Поэтому мы сразу же столкнулись с проблемой № 3. Мы должны были научить SimDB, расширение ТСУБД ChronosDB, планировать выполнение новым способом: построением проактивных планов моделирования (PSPs) [56, 61].

Например, во время разворачивания цикла, одно и то же имя массива встречается 100 раз: например, последняя команда `calc` удаляет текущий массив `$speed` и создает новый массив с тем же именем. Мы должны иметь возможность хранить и обращаться ко всем массивам (удаленным и новым) и записывать/читать их все одновременно, когда мы строим и выполняем план моделирования.

SimDB использует строгие формальные определения операций с массивами (тензорами), разд. 2.2 и компиляторные технологии для построения и выполнения PSPs

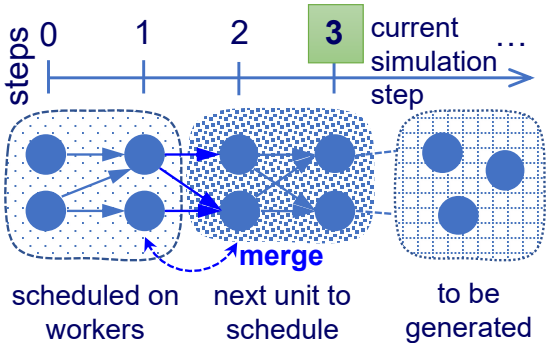


Рис. 3.6: Проактивный план моделирования [61]

на несколько шагов вперед, позволяя SimDB избежать избыточных материализаций и уменьшить накладные расходы на планирование; например, задачи "`scheduled on workers`" могут выполняться без коммуникации с координатором, рис. 3.6.

Рис. 3.7: Состояния исключительной блокировки: круг – состояние, стрелка – разрешенных переход

Чтобы физически поддерживать несколько массивов (тензоров) с одним и тем же именем во время выполнения, мы ввели механизмы версионирования и блокировок массивов (тензоров), которые работают в тандеме. SimDB обращается к массиву (тензору) по его имени и версии, одновременно оперируя и с удаленными, и с новыми массивами (тензорами): (n, v), где n – полное имя (разд. 3.1.2), а $v \in \mathbb{Z}$ – версия. Наборы данных разных версий, но с одинаковым именем, хранятся отдельно. Таким образом, мы создаем PSPs, содержащие удаленные массивы (тензоры), от которых зависят другие массивы (тензоры) [56, 61].

В отличие от ChronosDB, SimDB поддерживает два новых дополнительных типа наборов данных: staging и permanent. При задании имени n можно получить исключительную или неисключительную блокировку на n. Последняя позволяет UDF совместно использовать набор данных только для чтения. Первая позволяет UDF контролировать состояние набора данных (рис. 3.7) и не позволять другим UDF изменять набор данных. Заблокированные наборы данных становятся staging: их метаданные находятся в специальном хранилище, не видимом для пользователей. Staging набор данных может стать permanent, см. ниже.

Жизненный цикл набора данных SimDB выглядит следующим образом. Команда UDF должна запросить у Dataset Pool получение исключительной блокировки для имени n и последней версии, если она собирается выполнить `этап`: обнаружить (разд. 3.1.2), прочитать, удалить или создать набор данных с именем n. Для перезаписи необходимо выполнить этапы удаления и создания. Обратите внимание, что `удаление` – новый этап по сравнению с ChronosDB, вызывающий наибольшие сложности. Чтение также поддерживается неисключительными блокировками, поэтому мы опускаем его описание. Любой этап работы с набором данных не является мгновенным, например, невозможно немедленно удалить большой распределенный

набор данных, поэтому существуют промежуточные состояния, рис. 3.7. Операция `commit` завершает любой этап.

Если исключительная блокировка n успешна, создается новый набор данных в состоянии ①, если не существует постоянного набора данных с именем n, или в состоянии ② в противном случае. Команда UDF может выполнить этап `создать`: ① инициировать создание $\mapsto$ ⑤ указать метаданные, заполнить набор данных новыми подмассивами, commit $\mapsto$ ③ успешно зарегистрировать метаданные и подмассивы $\mapsto$ ②. Аналогично для `удалить`: ② инициировать удаление $\mapsto$ ⑦ commit $\mapsto$ ⑧ успешно применить изменения в Dataset Pool $\mapsto$ ⑨. Метаданные и подмассивы удаленного набора данных все еще физически существуют и могут быть использованы командами UDF, которые ранее обращались к ним. Если набор данных с именем n удален, команда UDF может создать новый набор данных с именем n с новой версией v', но, возможно, с совершенно другими метаданными: через состояния ⑩, ⑪ и ⑫. Обратите внимание, что v одинаково во всех состояниях. Состояние ⑫ указывает на то, что (n, v) не может развиваться дальше: он удален и, возможно, существует более новая версия.

Когда блокировка снята (например, при завершении UDF), а постоянного набора данных (n, v) не существует, SimDB регистрирует набор данных в состоянии ② в качестве постоянного. В противном случае, SimDB перезаписывает (n, v), если существует более новый набор данных (n, v') или удаляет (n, v), если его состояние ⑨. Сборщик мусора SimDB физически удаляет подмассивы остальных staging наборов данных.

Когда пользователи пишут UDF, им не нужно ничего знать о версиях наборов данных, блокировках или планировании: все механизмы, описанные в этом разделе, прозрачны для пользователей. Они обеспечивают эффективное моделирование, но потребовали глубоких модификаций нашей ТСУБД [56, 61].

3.4 Первая ТСУБД полностью в Web-браузере

3.4.1 Время работать с тензорами в Web-браузерах

Большие тензорные данные с их быстрым ростом стимулируют развитие клиент-серверных приложений, особенно на основе Web. Поэтому, Web-браузеры становятся все более популярными платформами для клиентских приложений, предоставляющих возможности работы с тензорами: Web-ГИС (географические информационные системы). Однако Web-ГИС еще далеки от зрелости, поскольку они в существенной степени недостаточно используют возможности современных Web-браузеров.

Современные Web-ГИС выполняют всю обработку массивов (тензоров) на стороне сервера. Они используют различные протоколы, открытые и/или проприетарные, для отправки запросов (команд обработки массивов/тензоров) в облако (на стороне сервера) и получения результатов небольшими порциями (например, 2-мерных тайлов изображений для отображения в Web-браузере на стороне клиента). Это увеличивает время отклика, вплоть до нескольких секунд, значительно ухудшая пользовательский опыт [63].

Компании утверждают, что снижение задержки всего на 0,1 с может повлиять на поведение пользователя и повысить конверсию [37]. ArrayGIS и WebArrayDB, наши новые Web-ГИС и ТСУБД на базе Web, доказывают, что Web-ГИС могут на-

правлять множество связанных с тензорами запросов в ТСУБД, работающую полностью в Web-браузере, чтобы значительно сократить время отклика на запросы.

ArrayGIS основана на WebArrayDB. Они могут быть более чем в 2 раза быстрее по сравнению с запросами только к Sentinel-Hub [68], облачному сервису для распространения и обработки популярных данных Sentinel, разд. 4.2.3.

3.4.2 Организация WebArrayDB

WebArrayDB – первая ТСУБД на чистом JavaScript, которая полностью работает в Web-браузере, рис. 3.8 [63].

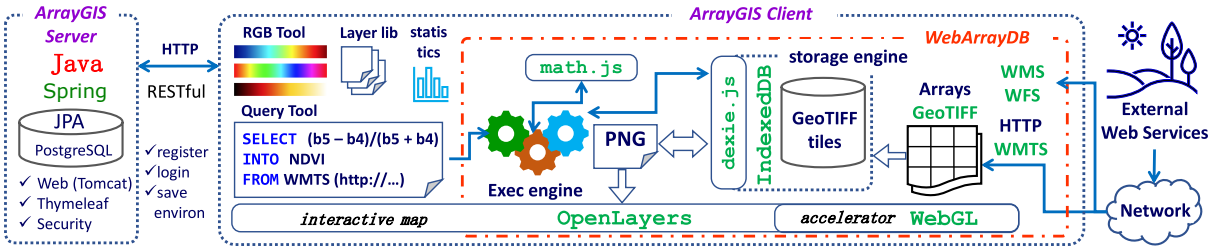


Рис. 3.8: Архитектуры WebArrayDB и ArrayGIS [63]

Форматы данных. WebArrayDB использует оба типа форматов: те, которые содержат сырые данные и изображения, например, GeoTIFF и PNG. Первые форматы довольно сложны и традиционно используются в настольных приложениях, но последние достижения в области Web-разработки позволили работать с GeoTIFF и подобными форматами в Web-браузере.

Усвоение данных. Поскольку WebArrayDB запускается в Web-браузере, она ожидает, что другие Web-сервисы (а не локальные файлы) будут её основными источниками данных. WebArrayDB может загружать тайлы GeoTIFF через популярный протокол OGC WMTS (Web Map Tile Service) [75]. WebArrayDB выполняет (1) преобразование файлов GeoTIFF "на лету" в структуру, поддерживаемую OpenLayers (для визуализации) и (2) сохраняет сырые тайлы в хранилище WebArrayDB для дальнейшего использования.

Хранилище. WebArrayDB хранит сырые тайлы данных в виде BLOB напрямую через API Web-браузера: тайлы, массивы (тензоры), вместе с определенными метаданными, например, ключом URL и границами. WebArrayDB хранит на стороне клиента только ограниченный объем данных, задаваемый параметром. Когда пространство исчерпывается, наименее часто используемый тайл удаляется.

Разбор запроса. WebArrayDB использует синтаксис SQL-подобных запросов:

```
SELECT (band8 - band4)/(band8 + band4)
INTO NDVI
FROM WMTS (https://services.sentinel-hub.com/ogc/wmts/
          <personal_api_key>?REQUEST=GetCapabilities)
```

Запрос вычисляет NDVI, популярный индекс растительности [78]. В отличие от обычного SQL, запрос принимает на вход тензоры: каналы 8 и 4 Sentinel (2-мерные массивы). `FROM` предписывает WebArrayDB получить входные данные с дистанционного сервиса по протоколу WMTS. В результате запрос также генерирует тензоры, в данном случае 2-мерный массив с именем NDVI. Он попадёт в хранилище внутри Web-браузера.

Планы **выполнения запроса** состоят из следующих фаз: (1) загрузка, (2) соединение, (3) вычисление и (4) рендеринг. Системы ArrayGIS и WebArrayDB работают в тандеме: первый запрашивает у второго только те результирующие тензорные плитки, которые будут сразу видны пользователю. Когда пользователь перемещает или масштабирует карту, ArrayGIS и WebArrayDB быстро генерируют новые выходные плитки на лету. Можно использовать объединения массивов и GPU.

Соединения массивов. Когда запрос включает несколько массивов (тензоров), может потребоваться операция соединения [52]. WebArrayDB поддерживает извлечение входных тензоров (1) из ответа WMTS или (2) различных слоев. Выходной тензор разбивается на плитки, используя наименьшую входную плитку. При этом вместо полного ретайлинга [54] WebArrayDB выдает тайлы инкрементально [63].

3.4.3 ArrayGIS: компоненты WebGIS

ArrayGIS – инновационная Web-ГИС (географическая информационная система) с интерактивным графическим Web-интерфейсом [63], рис. 3.9. WebArrayDB является основой ArrayGIS. Отличительной особенностью ArrayGIS является то, что она может работать с сырыми тензорными данными непосредственно в Web-браузере, что мы и демонстрируем в разд. 4.2.3.

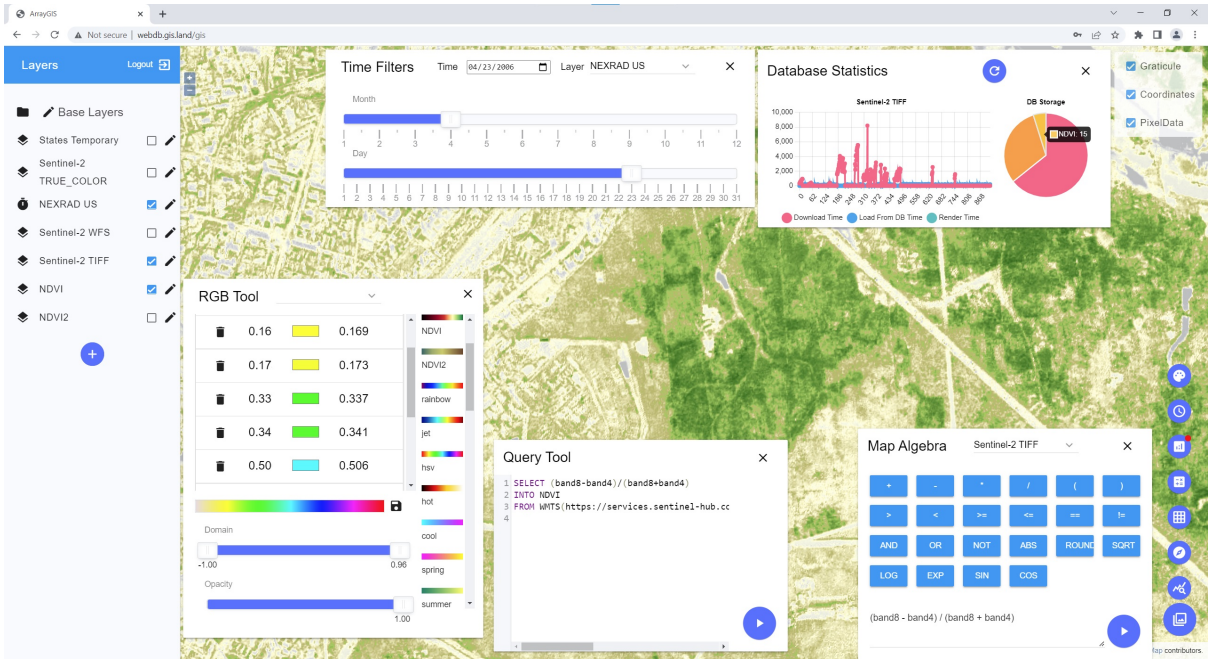


Рис. 3.9: Интерактивный графический интерфейс ArrayGIS [63]

Библиотека слоев позволяет пользователям управлять 2-мерными и 3-мерными слоями по протоколам WMS, WFS и WMTS. Поддерживаются GeoTIFF или обычные изображения.

Инструмент “Значение под курсором” отображает исходные сырые значения ячеек по щелчку мышью. Это очень быстро, поскольку происходит без взаимодействия клиента и сервера: ArrayGIS может работать с сырыми исходными данными.

RGB Tool предоставляет предопределенные цветовые палитры которые пользователи могут изменять или создавать новые (например, добавлять/удалять/переставлять цвета, изменять прозрачность). Слой с новыми цветами перерисовывается

почти мгновенно, поскольку ArrayGIS может устанавливать/настраивать цветовые палитры без клиент-серверной коммуникации.

Алгебра массивов (карт) – популярный язык анализа [70] и одна из наиболее частых рабочих нагрузок ТСУБД [52]. ArrayGIS и WebArrayDB принимают SQL-запросы, содержащие выражения алгебры массивов (карт) и выполняют быстрые вычисления непосредственно в Web-браузере.

Производительность WebArrayDB и ArrayGIS можно оценить по отзывчивости графического интерфейса и интерактивным статистическим диаграммам в графическом интерфейсе. ArrayGIS и видео о ней находятся в свободном доступе на её домашней странице*.

3.5 FastMosaic: новый, масштабируемый оператор мозаики

3.5.1 Процесс создания мозаики

FastMosaic реализует подходы из раздела 2.5, рис. 3.10. В тексте мы используем M и K вместо N и k в [59]. Доступны два режима мозаики, предназначенные для разных целей. Каждый режим генерирует мозаику.

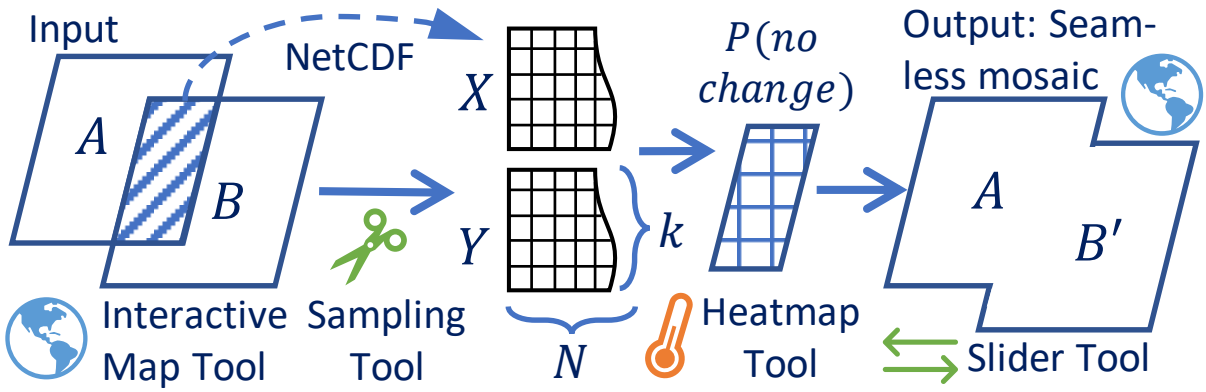


Рис. 3.10: Обзор рабочего процесса FastMosaic (2 входных массива)

Первый режим (ручное планирование) – пакетный запуск на всех входных массивах с использованием ранее созданного плана выполнения мозаики. Пользователь взаимодействует с FastMosaic для предоставления плана и получает в результате большую выходную мозаику, построенную на всех входных массивах в соответствии с планом. Этот режим позволяет экспериментировать с влиянием порядка добавления тензоров в итоговую мозаику. Это связано с тем, что коэффициенты трансформации вычисляются для пар тензоров: в процессе мозаики они накладываются друг на друга, что приводит к нелинейному преобразованию входных тензоров.

Второй режим (пошаговое руководство пользователя) принимает на вход только два тензора: план выполнения мозаики, который предписывает FastMosaic объединить два тензора. Этот режим позволяет пользователям проводить глубокое исследование механизмов FastMosaic на каждом этапе процесса мозаики.

*https://wikience.github.io/webdb2022

В подробном, пошаговом режиме (рис. 3.10), пользователь может изучить входные тензоры на интерактивной карте. Пользователь должен выполнить ряд действий, чтобы построить мозаику из двух перекрывающихся тензоров. Во-первых, пользователь запускает Sampling Tool для извлечения ячеек из области перекрытия двух тензоров: X и Y (массивы $M \times K$, где M есть количество перекрывающихся ячеек, а K есть размерность A и B, без учета пространственных измерений). Далее пользователь запускает CCA и другие алгоритмы для создания карты вероятностей отсутствия изменений, которую можно интерактивно исследовать. Наконец, инструмент Mosaic Tool позволяет применить коэффициенты трансформации и построить окончательную бесшовную мозаику.

3.5.2 Насыщенный и интерактивный интерфейс

Графический интерфейс пользователя FastMosaic имеет несколько окон: главное, руководства, консоли, коэффициентов и графиков корреляций. Кроме того, пользователь видит папку для хранения данных во время работы FastMosaic, рис. 3.11.

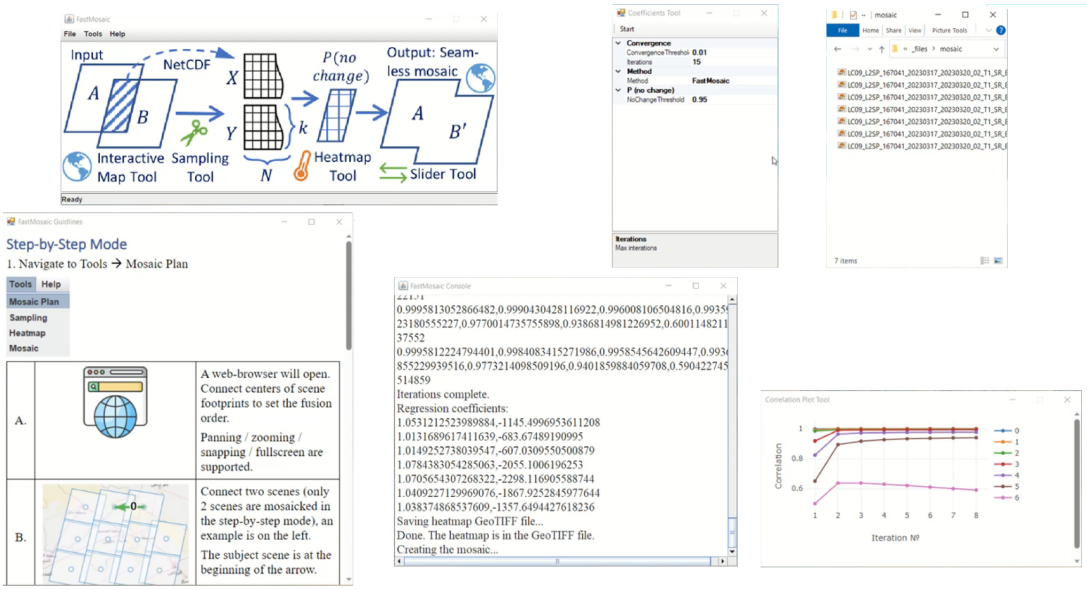


Рис. 3.11: Экранный снимок интерфейса FastMosaic

Окно руководства содержит подробные инструкции с иллюстрациями по пошаговому построению бесшовной мозаики (создание плана выполнения мозаики, запуск выборки данных и других алгоритмов FastMosaic), а также рекомендации по интерактивному исследованию входных и выходных данных.

Окно консоли доступно только для чтения и выводит важную информацию во время работы алгоритмов FastMosaic. Например, в окне консоли отображаются планы выполнения мозаики и канонические коэффициенты корреляции вместе с итоговыми коэффициентами трансформации в виде человекочитаемого текста во время выполнения CCA, MAD и IR-MAD.

Окно коэффициентов позволяет задавать параметры для выполнения CCA, MAD и IR-MAD, а окно графиков корреляций содержит интерактивную диаграмму, которая обновляется на каждом шаге итерации во время выполнения FastMosaic. В окне отображаются корреляции пар канонических переменных. С помощью этого инструмента пользователь может исследовать сходимость алгоритма.

Глава 4

Приложения: реальные данные и варианты использования

Каким образом представленные ранее результаты применены для решения важных практических задач? Мы используем реальные данные и демонстрируем эффективность новых теоретических подходов, архитектурных и реализационных аспектов программного обеспечения. Это оценка на реальных данных и проблемах, а также ещё один способ продемонстрировать практическую значимость наших результатов.

Мы называем это "пересмотром" реальных вариантов использования: рассматриваемые проблемы могут быть решены и по-другому. Например, можно использовать сценарии пакетной обработки данных или MPI-программы для суперкомпьютера. Однако, это сопряжено с проблемами управления данными [9, 54]. Поэтому, ТСУБД, с присущими им преимуществами, являются привлекательными альтернативами или вспомогательными системами в связи с необходимостью управления большими объемами тензоров, например, в национальных инициативах [9].

ТСУБД предлагают новые пути решения проблем, более надежно и эффективно по сравнению с существующими платформами [54]. Зачастую ТСУБД – новый вспомогательный инструмент, более быстрый, простой и масштабируемый по сравнению с существующими решениями.

Даже если ТСУБД не полностью вытеснят некоторые конкретные системы, они определенно могут служить отличным дополнением к миру подходов по управлению большими тензорами, их обработке, визуализации и другим задачам, связанным с тензорами. Таким образом, ТСУБД изменяют уже существующие конвейеры, разработанные для реальных вариантов использования, а также открывают новые возможности, особенно с учетом быстрого и непрерывного роста объема тензоров [9].

Современная роль ТСУБД заключается в предоставлении их качественных и количественных преимуществ (стр. №11) конвейерам, в которых используются большие многомерные массивы (тензоры).

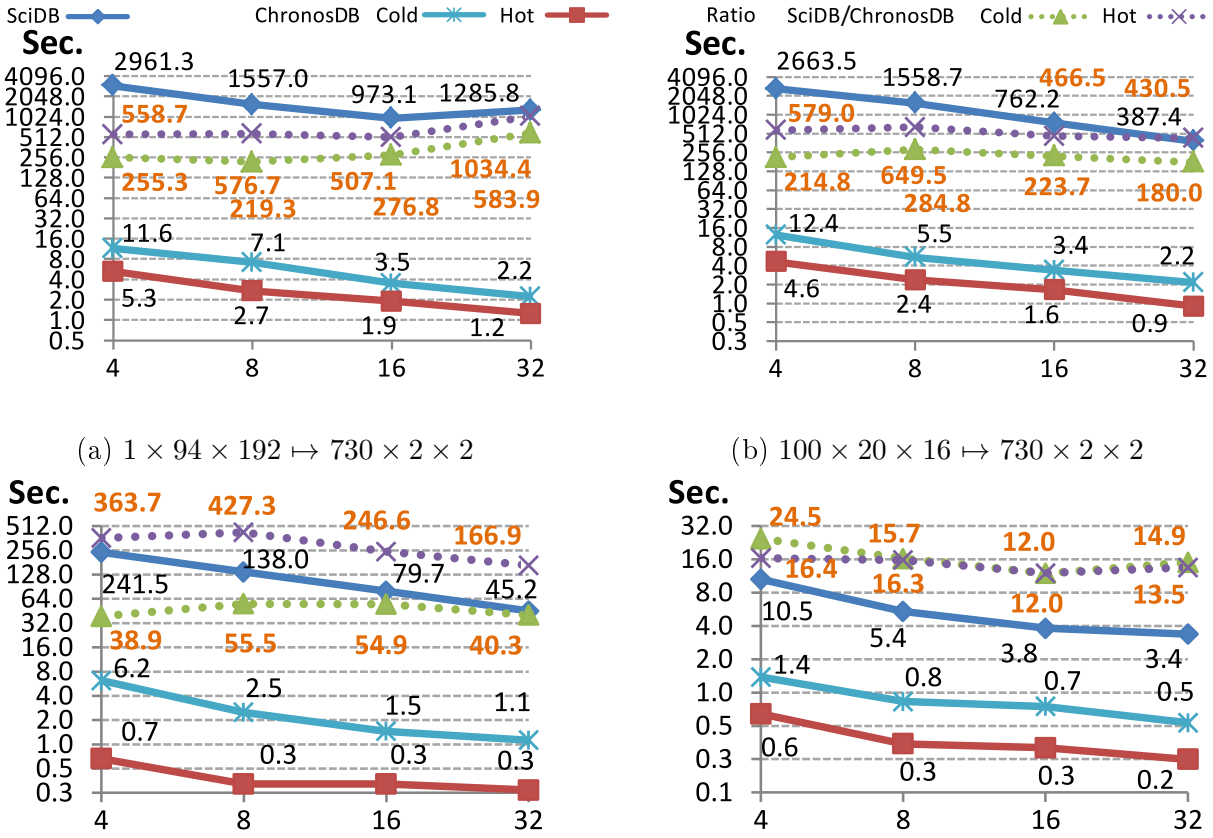
4.1 Данные о Земле и климате: управление, обработка и визуализация

4.1.1 Высокопроизводительные управление и обработка

Новая модель данных (разд. 2.1), новые алгоритмы работы с массивами (тензорами) (разд. 2.2) и подходы к управлению (разд. 3.1.2) реализованы в ChronosDB и пре-

восходят SciDB в среднем в 75 раз. Они всегда быстрее и могут опережать SciDB до 1024 раз. На момент сравнения SciDB была единственной свободно распространяемой распределенной ТСУБД [54]. SciDB разработана компанией Paradigm4 и М. Стоунбрейкером, лауреатом премии А. Тьюринга (“Нобелевская премия по компьютерным наукам”).

Мы развернули в облаке компьютерные кластеры, состоящие из 4, 8, 16 и 32 виртуальных машин. ChronosDB и SciDB были развернуты на собственных компьютерных кластерах в облаке. Подробные характеристики аппаратного и программного обеспечения, используемого для оценки производительности см. в [54]. Чтобы добиться максимальной производительности SciDB, мы тщательно настроили её [54].



(a) $1 \times 94 \times 192 \mapsto 730 \times 2 \times 2$

(b) $100 \times 20 \times 16 \mapsto 730 \times 2 \times 2$

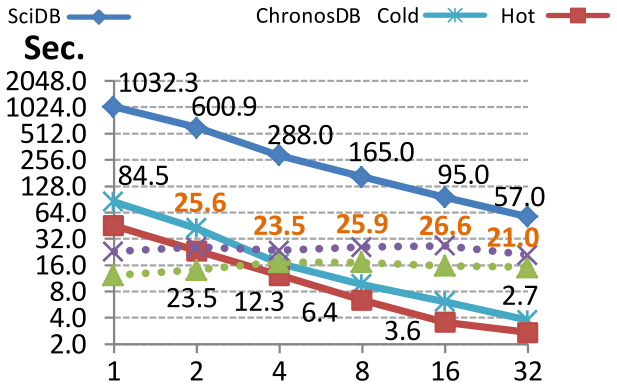
(c) $[, 0 : 20, 0 : 20]$, $1 \times 94 \times 192$

(d) $[, 0 : 20, 0 : 20]$, $100 \times 20 \times 16$

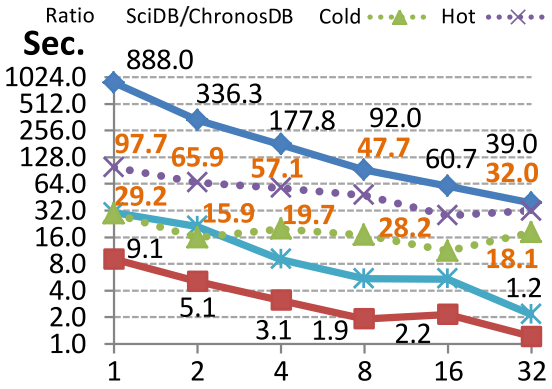
Рис. 4.1: (a, b) чанкинг, (c, d) извлечение гиперсреза $[0 : 46751, 0 : 20, 0 : 20]$, По горизонтальным осям отложено количество узлов кластера.

Мы экспериментировали с набором данных Landsat из разд. 3.1.2. Сырой набор данных должен был быть подвергнут ретайлингу в регулярный набор данных, что заняло **≈30 секунд** на кластере из 8 узлов. Для SciDB, для каждого канала сцены, мы должны были объединить несколько файлов GeoTIFF в одну большую мозаику. SciDB импортировал такую мозаику за **≈2 часа** на мощном сервере, чтобы не терять время Облака. Напротив, ChronosDB работает с файлами GeoTIFF на месте, без импорта во внутренний формат СУБД.

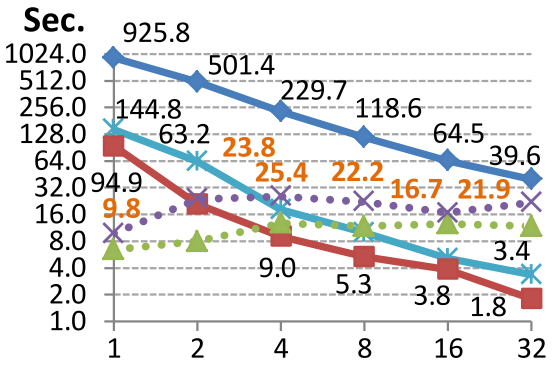
Мы также сформировали набор данных по скорости восточного (u-wind) и северного (v-wind) ветра на высоте 10 метров над поверхностью за 1979-2010 годы (32 года) реанализа NCEP/DOE AMIP-II (R2) [40]: гауссовы сетки в формате NetCDF3.



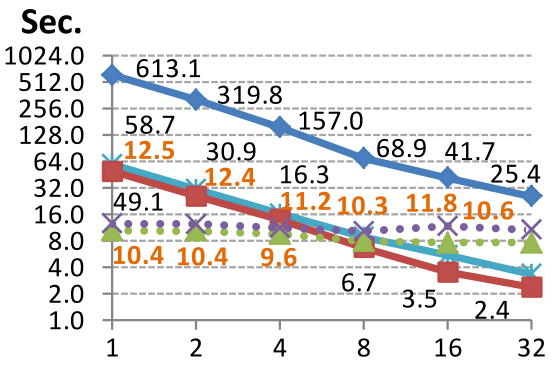
(a) Решейпинг



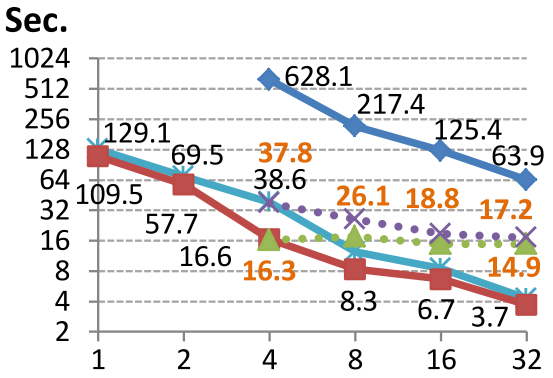
(b) Пирамида 1024 & 4096



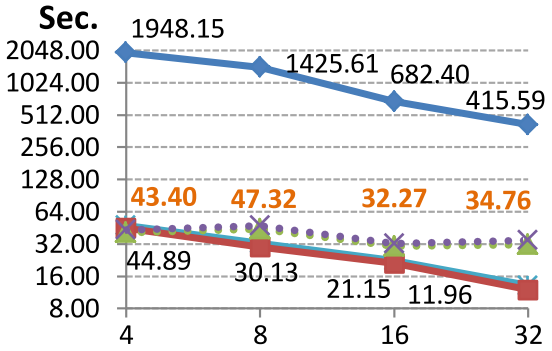
(c) Скорость ветра



(d) Интерполяция 1024 & 2048



(e) NDVI



(f) SAVI 1024 & 4096

Рис. 4.2: a & b означает, что использовались $a \times a$ чанки SciDB и $b \times b$ подмассивы ChronosDB. По горизонтальным осям отложено количество узлов кластера.

ChronosDB работает с файлами NetCDF напрямую, на месте (in situ) и может с готовностью выполнять запросы над данными в NetCDF, но нам пришлось разработать специальное программное обеспечение для импорта данных в SciDB. Импорт занял более **45 часов** на мощном сервере, чтобы не тратить время Облака.

Мы оценили холодный и горячий запуск запросов: запрос выполняется в первый и второй раз соответственно. ChronosDB использует преимущества встроенного кэширования ОС и работает гораздо быстрее во время горячих запусков. Это особенно полезно для повторных экспериментов с одними и теми же данными, разд. 2.3.1.

Существенной разницы во времени работы SciDB между холодным и горячим запуском нет.

Рисунок 4.1 представляет производительность чанкинга и извлечения гиперсреза. Рисунок 4.2a сообщает о производительности решейпинга $(time, lat, lon) \mapsto (lon, lat, time)$. Соотношение достигает 26 раз. Мы оценили создание 3 уровней многоуровневой пирамиды (рис. 4.2b) и интерполяции в 2 раза (рис. 4.2d). ChronosDB превосходит SciDB по производительности в 97 раз и 12 раз соответственно.

Скорость ветра (ws) в каждой ячейке сетки и временной точке рассчитывается по формуле $ws = \sqrt{\text{u-wind}^2 + \text{v-wind}^2}$. Отношение достигает 25 раз (рис. 4.2c).

Вычисление NDVI демонстрирует распределенное K-путевое соединение массивов (тензорнов), разд. 2.2.2. SciDB не удается вычислить NDVI на 1- и 2-узловых кластерах и завершается аварийно с ошибкой `недостаточно памяти`. ChronosDB значительно превосходит SciDB (до 37 раз).

Мы также провели сравнительный анализ вычислений SAVI (не только сам SAVI, но и полный сложный план выполнения: конвейер SAVI), разд. 3.1.3. В результате ChronosDB оказывается от 32 раз до 47 раз быстрее SciDB, рис. 4.2f. Это доказывает эффективность ChronosDB при работе со сложными аналитическими конвейерами.

Более подробная информация об оценке производительности находится в [54].

4.1.2 Графический интерфейс и DWMTS для ТСУБД

Используя ChronosDB, можно интерактивно визуализировать большие объемы данных массивов (тензоров). Графический Web-интерфейс и специализированная, собственная реализация DWMTS (Distributed WMTS), предоставляемая ChronosDB “из коробки”, способствуют достижению вышеупомянутой цели [55].

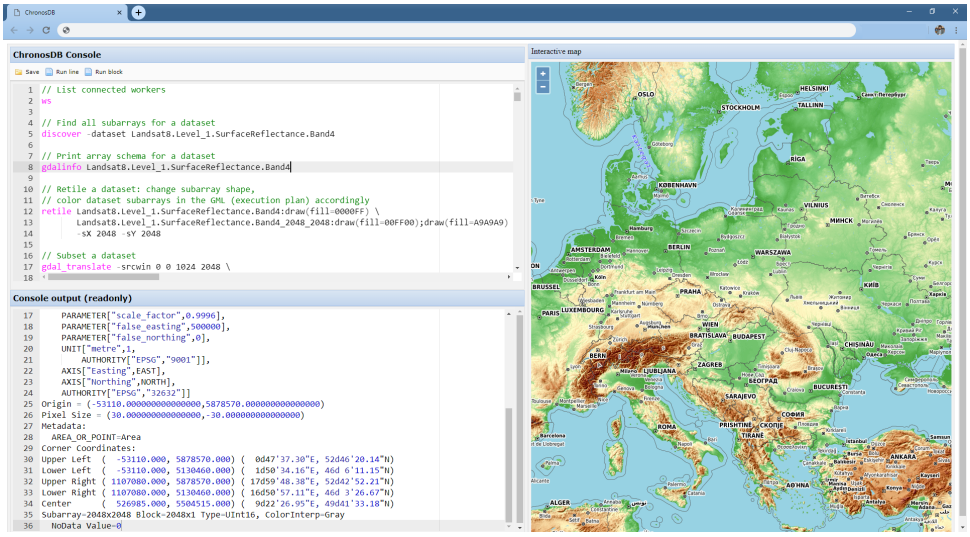


Рис. 4.3: Графический Web-интерфейс ChronosDB [55]

Графический Web-интерфейс ChronosDB состоит из трех основных частей: (1) консоль ChronosDB, (2) вывод консоли, и (3) интерактивная карта, рис. 4.3. Можно редактировать несколько сценариев в консоли ChronosDB (поддерживается подсветка синтаксиса), отправлять на выполнение одну строку или фрагмент кода. Web-интерфейс устанавливает сессию с ChronosDB через собственный сетевой

протокол. Сообщение GUI $\mapsto$ ChronosDB содержит сценарий, ChronosDB $\mapsto$ GUI сообщения содержат вывод сценария, присоединяемый к выводу консоли.

Визуализация играет важную роль в понимании данных. Интерактивная карта отображает наборы данных ChronosDB. Пользователи могут переключать базовые слои, добавлять/удалять массивы с карты и настраивать цветовые схемы. ChronosDB визуализирует массивы путем рендеринга подмассивов и доставки изображений посредством WMTS (Web Map Tile Service), популярного протокола OGC для передачи привязанных геопривязанных тайлов по HTTP [75]. Любая интерактивная Web-карта или настольное программное обеспечение, поддерживающее WMTS, может визуализировать наборы данных ChronosDB.

ChronosDB анализирует запрос тайла WMTS и предоставляет отрисованное изображение непосредственно с узла, на котором находится подмассив. Большинство популярных серверов WMTS работают на одной машине. ChronosDB позволяет визуализировать большие массивы (тензоры), сокращая при этом перемещение данных между узлами. ChronosDB облегчает пользователям восприятие входных данных, а также визуальную оценку производных данных, полученных в результате выполнения сценариев [55].

4.2 Быстрая интерактивная наука о данных: быстрый пересчет тензоров (обновления)

Специалисты по данным могут столкнуться с увеличением времени отклика программного обеспечения, позволяющего работать с большими массивами (тензорами) в интерактивном режиме. Поскольку пользователи обычно проводят время в ожидании таких ответов за компьютером, каждая последующая задержка обработки данных, даже в пределах 1-2 секунд, повышает утомляемость человека и тем самым снижает качество работы и понимание данных.

Здесь мы описываем несколько важных практических приложений которые получают пользу от BitFun, WebArrayDB и ArrayGIS [53, 63], значительно ускоряя обновление тензоров (до 8 раз) в ответ на ручной или автоматический ввод.

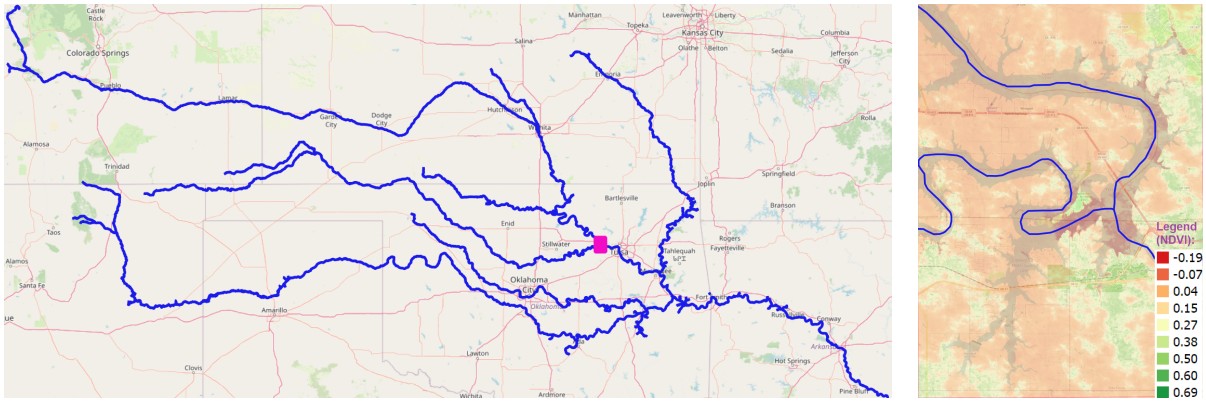
4.2.1 Управление водными ресурсами и карты наводнений

Это приложение иллюстрирует быструю оценку $f(\tau) < const$ благодаря новой технике индексирования разд. 2.3.2. Для создания карты наводнения мы создаем маску воды: 2-мерный массив с двумя значениями ячеек: 1 (вода) и 0 (воды нет).

В качестве примера мы выбрали бассейн реки Арканзас. Её длина составляет 1 469 миль (2 364 км), поэтому река Арканзас является 6-й и 45-й по длине рекой в США и мире соответственно. Площадь ее водосборного бассейна составляет 161 000 кв. миль (417 000 кв. км) [5]. Рисунок 4.4b (28 мая 2019 г.; Landsat 8) увеличивает розовую рамку на рис. 4.4a.

Весной 2019 года юг и центр США пережили сильное наводнение. Наиболее остро проблема проявилась в конце мая вдоль реки Арканзас. Во всех округах Оклахомы был введен режим чрезвычайного положения, а в нескольких населенных пунктах Арканзаса было приказано или рекомендовано провести эвакуацию [38].

Данные дистанционного зондирования широко используются на практике для прогнозирования речных наводнений, оценки нанесенного ущерба, выявления райо-



(a) Река Арканзас с ее наиболее заметными притоками (b) Zoomed NDVI box

Рис. 4.4: Район для BitFun урока “Вода” (картирование наводнения)

нов, подверженных наводнениям, выбора мест для строительства защитных дамб и быстрого реагирования на стихийные бедствия [4].

Значения NDVI, близкие к нулю или отрицательные, представляют собой зоны с присутствием воды, разд. 2.3.1. Задача состоит в том, чтобы быстро создать точную маску воды путем настройки τ в $f(\tau) = \text{NDVI} - \tau < 0$. В качестве примера, аналитик видит карту RGB, полученную маску, и набор точек двух цветов (точные данные) расположенных на затопленных и незатопленных территориях [53].

Задача состоит в том, чтобы настроить τ таким образом, чтобы для большинства точек, указывающих на наводнение и отсутствие наводнения, значения маски воды были равны 1 и 0, соответственно. Выбранная область служит хорошим примером, поскольку бассейн реки Арканзас занимает относительно большую площадь. BitFun быстро воссоздает маску воды для этой области каждый раз, когда пользователь настраивает τ [53]*.

4.2.2 Продовольственная безопасность и прогноз урожая

Теперь мы проиллюстрируем быстрое вычисление $f(\tau)$ благодаря новым приемам индексирования, разд. 2.3.2. Значения $f(\tau)$ необходимы для модели урожайности сельскохозяйственных культур.

Индексы растительности широко используются в ML/AI и продовольственной безопасности: классификация [16], сегментация [79], угрозы засухи [74], здоровье пахотных земель [21], точное земледелие [35], прогнозирование урожайности [2] и это лишь некоторые из них.

SAVI используется для засушливых регионов (например, сельскохозяйственных полей Саудовской Аравии) с редкой растительностью и открытыми поверхностями почвы [78]. SAVI работает намного лучше, чем другие вегетационные индексы, поскольку он стремится минимизировать влияние яркости почвы, вводя настраиваемый параметр L, почвенный коэффициент, изменяющийся от 0 до 1 в зависимости от почвы, разд. 2.3.1.

Централизованное орошение популярно в засушливых и гиперзасушливых регионах Земли. В Саудовской Аравии, рис. 4.5, основными культурами, выращиваемыми

*Домашняя страница BitFun: http://bitfun.gis.land

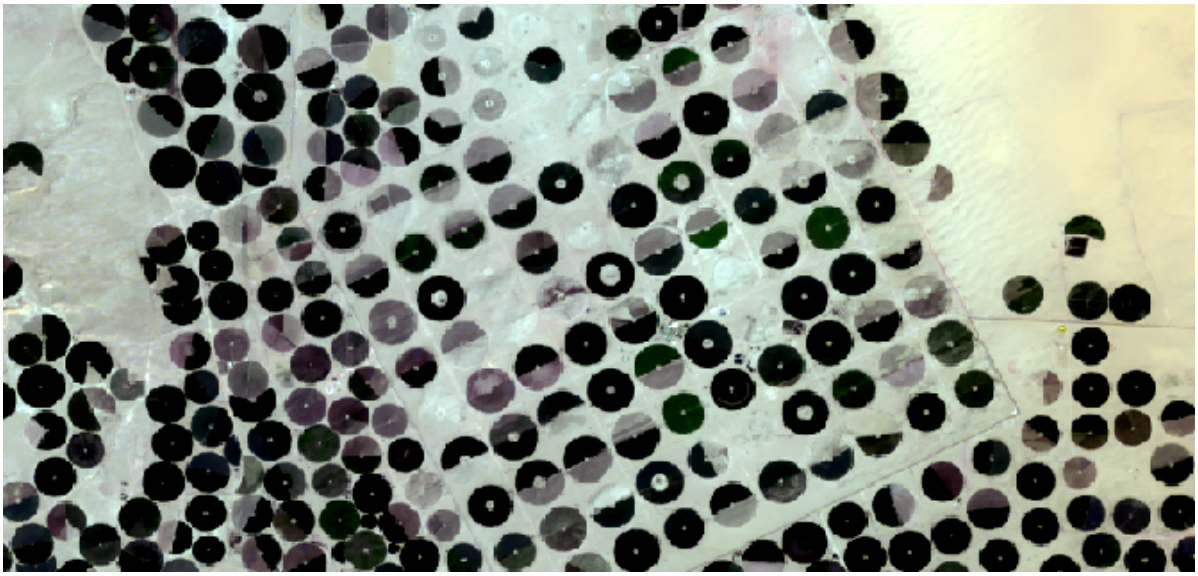


Рис. 4.5: Саудовская Аравия: системы Center Pivot Irrigation (район Wadi Al-Dawasir, к югу от Riyadh (столицы), 11/02/2016, RGB, Landsat 8 коллекция 1 уровень 2)

зимой, являются пшеница, картофель, томаты и дыни. Кормовые культуры выращиваются круглый год. Культивация проходит в пустыне при температуре до 43°C [2].

Центральное орошение – это метод орошения сельскохозяйственных культур с помощью поливочных машин, вращающихся вокруг центрального стержня [69]. Этот тип орошения требует меньше труда (меньше затрат) по сравнению с другими типами орошения и позволяет уменьшить сток воды, эрозию и уплотнение почвы.

В качестве примера, урок BitFun [53] использует работу [2]: авторы строят эмпирические модели урожайности культур для полей компании по развитию сельского хозяйства Саудовской Аравии (INMA) в этом районе, рис. 4.5. В зависимости от значения L в модели можно подавать различные значения SAVI. Проблема в том, что значение L не известно заранее и подбирается экспериментально. BitFun предоставляет новые методы индексирования, чтобы избежать вычисления SAVI с нуля.

BitFun быстро пересчитывает SAVI для большой области каждый раз, когда пользователь настраивает L чтобы проверить, являются ли новые, скорректированные значения SAVI для этого нового L более подходящими для моделей. Более того, поскольку урожайность оценивается до нескольких цифр после плавающей точки, возможность BitFun указать точность очень полезна в этом случае и значительно ускоряет вычисления[†].

4.2.3 Ускоренная Web-обработка и визуализация

Web-ГИС (географические информационные системы) – яркий пример того, каким образом тензорная обработка и визуализация становятся доступными через Web-браузеры. В то время когда другие Web-ГИС используют Web-браузеры в качестве тонких клиентов, отображая только сгенерированные сервером изображения, доступные только для чтения, WebArrayDB и ArrayGIS работают с сырыми данными полностью в Web-браузере, разд. 3.4.

[†]Домашняя страница BitFun: http://bitfun.gis.land

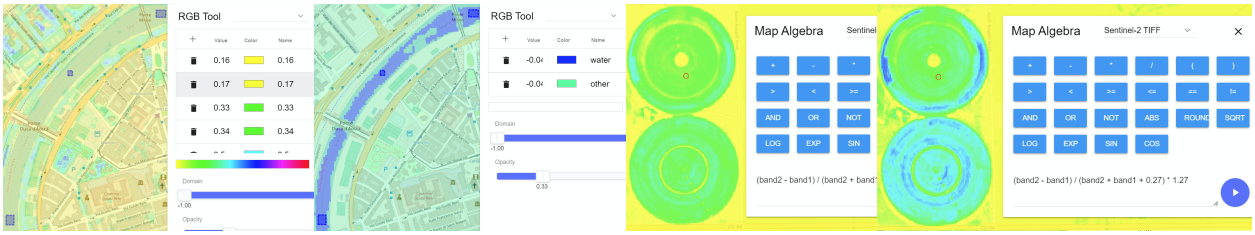


Рис. 4.6: Уроки WebArrayDB и ArrayGIS: исходные состояния и решения [63]

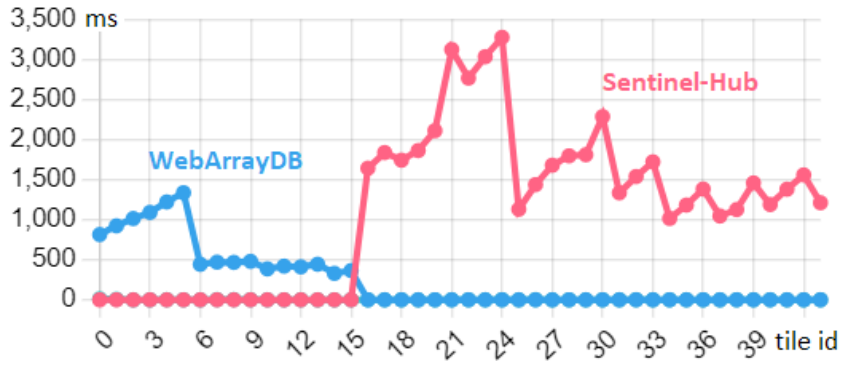


Рис. 4.7: Латентность: WebArrayDB и Sentinel-Hub

Мы иллюстрируем ускоренные вычисления на базе Web в двух уроках [63]: картирование водных объектов и оценка сельскохозяйственных культур, рис. 4.6. Хотя сценарии несколько напоминают те, что описаны в раздел. 4.2.1 и 4.2.2, используемые подходы ускорения отличаются друг от друга [63].

В частности, "Урок воды" демонстрирует практически мгновенное обновление цветовой палитры массива благодаря сохранению исходных данных массива в Web-браузере. WebArrayDB, основа ArrayGIS, обеспечивает доступ к исходным тензорам без клиент-серверного взаимодействия, в отличие от других Web-ГИС. Мы также используем популярный индекс NDVI в качестве индикатора воды, разд. 4.2.1, но для Рима (Италия) и данных Sentinel-2. Пользователи видят опорные данные и отправляют запросы **SELECT** NDVI $< \boldsymbol{\beta}$, где $\boldsymbol{\beta}$ – настраиваемый параметр, рис. 4.6.

ArrayGIS – это инновационная Web-ГИС, интерфейс для WebArrayDB. Она имеет интерактивный графический интерфейс и находится в свободном доступе по адресу `https://wikience.github.io/webdb2022` Следовательно, можно легко воспроизвести уроки и подтвердить преимущества ArrayGIS и WebArrayDB.

Пользователи могут проследить за производительностью систем WebArrayDB и ArrayGIS, изучив интерактивные графики в инфобоксе Database Statistics [63]. Рисунок 4.7 – скриншот из инфобокса Database Statistics.

ArrayGIS полагается на WebArrayDB. Вместе они могут быть более чем в 2 раза быстрее по сравнению с запросами только к Sentinel-Hub [63], популярного облачному сервису для распространения и обработки данных Sentinel, рис. 4.7[‡].

Подробное руководство по работе с графическим интерфейсом ArrayGIS находится в [63]. Видеопрезентации BitFun, WebArrayDB и ArrayGIS также можно найти на их домашних страницах [53, 63].

[‡]Обратите внимание, что WebArrayDB извлекает данные (обычно тайлы 256×256 либо 512×512) из Sentinel-Hub непосредственно в Web-браузер пользователя без посредников. Клиентская машина: Intel Core i5 1.6 GHz, 8 GB RAM, 256 GB SSD, Chrome 100 (64-bit), Windows 10, Internet до 100 MBit/s (WiFi).

4.3 Моделирование дорожного движения: полный производственный цикл с помощью ТСУБД

Этот раздел посвящен пошаговому, сквозному моделированию клеточных автоматов (КА) с помощью SimDB (разд. 3.3) в её интерактивном графическом интерфейсе [61], рис. 4.8. Мы также демонстрируем преимущества использования SimDB для моделирования КА, чтобы ответить на вопрос, почему SimDB является отличным инструментом для этой рабочей нагрузки.

Мы опираемся на статью [61] и Web-страницу [56] с очень подробными примерами[§]. Кроме того, мы также рекомендуем посетить домашнюю страницу [61], где также находится ссылка на соответствующее видео[¶].

Пользователи могут создавать дорожную сеть с помощью конструктора. Дорожная сеть преобразуется во входные массивы. Нативные UDF ТСУБД используются для инициализации моделирования и создания проактивных планов моделирования. Также поддерживается интеграция нативных и Java UDF, разд. 3.3. SimDB может анимировать массивы на интерактивной карте и является интероперабельной. Цель моделирования – получить статистику из массивов с историей моделирования для поддержки принятия решений. SimDB облегчает моделирование благодаря своим мощным возможностям.

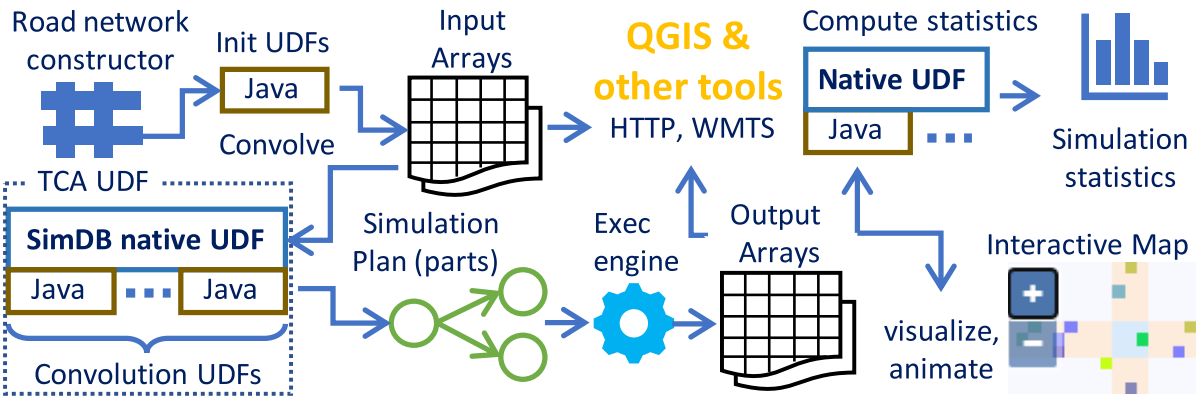


Рис. 4.8: Обзор сквозного моделирования с помощью SimDB [61]

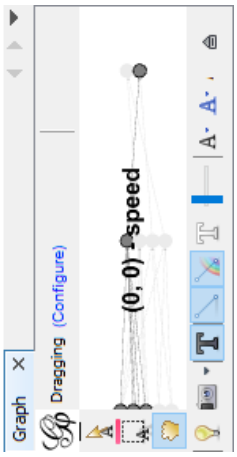
4.3.1 Инициализация моделирования и исследование плана

Физическая среда и состояния клеток могут быть смоделированы в виде 2-мерных массивов. Новые компоненты ТСУБД частично обусловлены нашим новым гибким оператором свертки и нативным языком UDF, что позволяет нам применять локальные правила перехода и задавать логику моделирования соответственно, разд. 3.3.

В качестве входных данных мы создаем и инициализируем минимум три 2-мерных массива с формой $lat \times lon$ (в качестве входных данных может использоваться большее количество массивов, например, погодные условия, рельеф местности):

- `tca.lane`: дорожная сеть, ячейки: -1: непроходимые, $0/1$: направление движения запад-восток/юг-север, 2: перекрестки, 3: светофоры

[§] http://sigmod2021.gis.gg (также содержит код (реализацию) нашего клеточного автомата TCA)
[¶] https://wikience.github.io/simdb2022



(a)

```
public void setSpeed(ConvolveWindows w) {
   double val = w.input(0).get(0, 0);
   Double output = null;
   if (val == 1 || val == 0) {
      output = (double) w.input(0).random().nextInt(4);
   } else {
      if (val == 4) {   // traffic lights
         int rnd = w.output(0).random().nextInt(2);
         output = (double) (200 + rnd + 1);
      }
   }
   w.output(0).set(output, 0, 0);
}
```

(b)

Рис. 4.9: (a): Часть плана, (b) UDF для задания скорости [61]

- `tca.speed`: начальные скорости транспортных средств, далее обновляемые по правилам CA (0 – означает, что транспортное средство не движется, положительные значения – скорость)
- `tca.length`: длина транспортных средств (транспортное средство любой длины моделируется ячейкой, содержащей его задний бампер)

Обратите внимание, что мы не храним положение автомобиля в явном виде поскольку они неявно задаются координатами ячеек. Обычно целью моделирования является получение статистических данных. Следовательно, мы инкрементально строим 3-мерные массивы истории ($time \times lat \times lon$): `speedh` и `lenh` (скорости и длины транспортных средств для каждого временного шага) путем добавления `tca.speed` и `tca.length` вдоль виртуальной оси $time$, разд. 3.3.2.

Инициализация состоит из двух последовательных этапов: (1) определить, занята ли клетка транспортным средством, (2) назначить длины и скорости для каждого транспортного средства.

UDF для установки скорости транспортного средства находится на рис. 4.9b. Мы назначаем случайную скорость ячейке после проверки того, позволяет ли окно свертки с центром в этой ячейке разместить транспортное средство. По пути мы также присваиваем количество тиков светофорам, если они нам встречаются.

SimDB также позволяет интерактивно исследовать проактивные планы моделирования (PSPs, разд. 3.3.3), чтобы получить представление о возможностях планирования и выполнения. Планы представлены на языке Graph Modeling Language [20]. SimDB выстраивает задачи на плоскости, чтобы избежать нагромождения, назначает цвета и аннотирует их обширной статистикой: например, назначение задач, ввод-вывод по сети, информация о наборе данных. Можно масштабировать/панорамировать PSP, создавать статистику, фильтровать, выделять задачи и их зависимости в Gephi [20], рис. 4.9a.

4.3.2 Интерактивная визуализация и анимация

Визуализация очень важна для понимания данных. SimDB предоставляет массивы через открытый, популярный, стандартный протокол WMTS [75] и отображает их в своем интерактивном графическом интерфейсе, рис. 4.10. Можно добавлять/уда-

лять массивы (например, `tca.lane`) на/из карты, панорамировать, масштабировать и настраивать их цветовую палитру (слои карты на рис. 4.10 прозрачны, поэтому цвета немного сливаются).

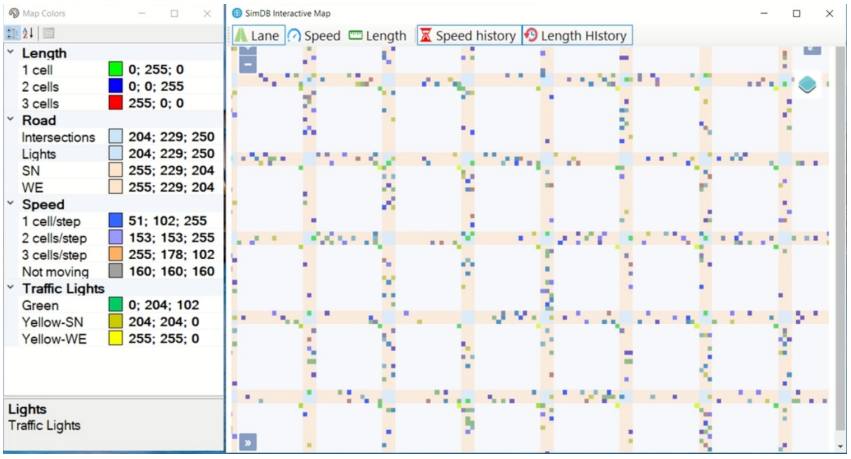


Рис. 4.10: Графический интерфейс SimDB: настройка цветов, интерактивная визуализация и анимация [61]

SimDB анимирует массивы истории, так что пользователи могут наблюдать моделирование в работе: каким образом автомобили движутся по дорогам, стоят в очереди на светофорах, поворачивают на перекрестках, меняют скорость и обгоняют друг друга [61], рис. 4.10.

4.3.3 Интероперабельность

Слой хранения SimDB основывается на сырых файлах в стандартных форматах, например GeoTIFF. Таким образом, тензоры легко доступны для других программ в виде хорошо известных бинарных файлов или визуализированных изображений. Например, чтобы просмотреть тензор на интерактивной карте Quantum GIS, добавьте набор данных SimDB в качестве слоя WMTS, рис. 4.11.

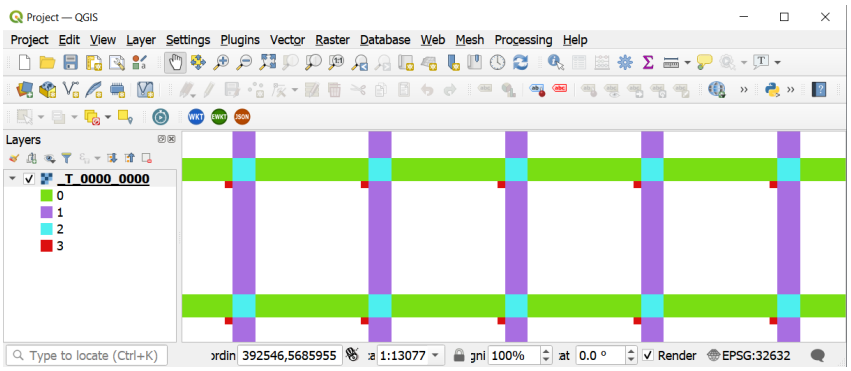


Рис. 4.11: Массив SimDB в Quantum GIS [61]

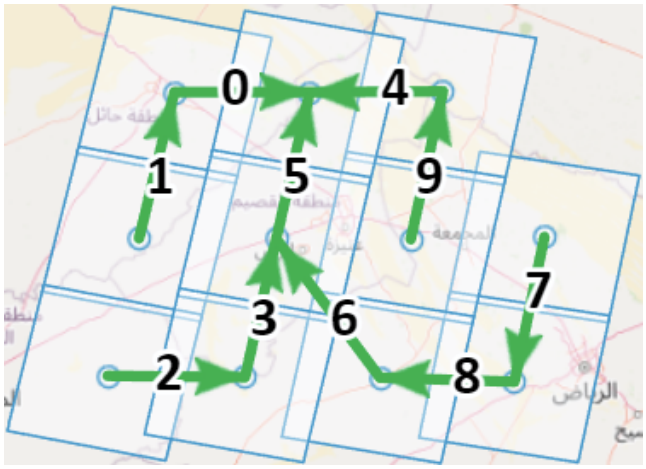
Больше информации в [61] и на домашних страницах [56, 61].

4.4 Быстрая и бесшовная мозаика: пошагово

FastMosaic может шаг за шагом сопровождать пользователей через весь рабочий процесс (разд. 3.5.1) построения бесшовной мозаики на сырых входных тензорах с помощью интерактивного и насыщенного графического интерфейса (разд. 3.5.2). Здесь мы кратко опишем основные шаги рабочего процесса мозаики. Мы рекомендуем читателю ознакомиться с домашней страницей FastMosaic[‖] для просмотра видео, демонстрирующего все этапы работы.

4.4.1 Создание плана мозаики

Сначала мы должны определить план мозаики с помощью инструмента Mosaic Plan Tool в интерфейсе. Пользователь в интерактивном режиме строит дерево (план выполнения мозаики), рисуя стрелки для соединения пар массивов и задавая порядок слияния. На шаге № i тензор в начале стрелки № i присоединяется к мозаике, построенной на данный момент, рис. 4.12a.



(a) План построения мозаики

```
{"type":"FeatureCollection",
   "features":[{
      "type":"Feature",
      "geometry": {
         "type":"LineString",
         "coordinates":
         [[510598.029, 3043050.697],
         [357855.877, 3042011.635]]
      },
      "properties": {"id": 0},
      "id":0
   }]
      }
```

(b) Крошечный план в GeoJSON

Рис. 4.12: Иллюстрации планов построения мозаики

Рисунок 4.12a показывает план выполнения для пакетного режима, который объединяет тензоры в заданном порядке. Тензоры (сцены со спутника Landsat 8) представлены в виде следов. Тензор-субъект находится в начале стрелки, поэтому FastMosaic вычисляет коэффициенты трансформации для тензоров-субъектов.

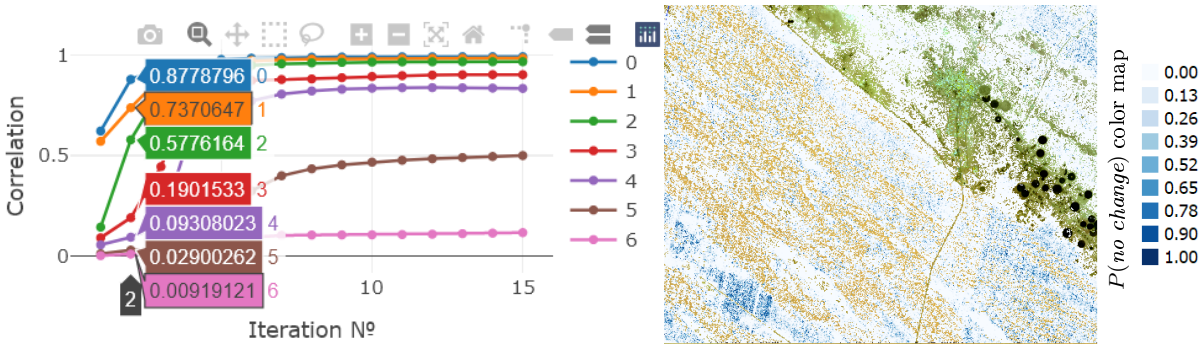
Планы посроения мозаик хранятся в формате GeoJSON. Рисунок 4.12b представляет план построения, который объединит два тензора в пошаговом режиме (стрелка № 4 на рис. 4.12a).

Когда план определен, FastMosaic готов к выборке данных для оценки вероятности отсутствия изменений и переходу к следующим шагам.

[‖] https://wikience.github.io/fastmosaic2023

4.4.2 Выборка, выполнение и тепловые карты

После определения массивов (тензоров), опорного и субъекта, мы готовы к созданию массивов X и Y (рис. 3.10) с помощью инструмента Sampling Tool в графическом интерфейсе для использования их в каноническом корреляционном анализе. Инструмент создает NetCDF файл с выбранными значениями ячеек из перекрывающихся ячеек входных массивов (тензоров).



(a) Корреляции канонических переменных (b) Тепловая карта $P(\text{no change})$

Рис. 4.13: Интерактивный график и тепловая карта

Далее мы можем использовать X и Y для выполнения CCA, MAD, IR-MAD, генерации карты вероятностей отсутствия изменений и коэффициентов регрессии. Напомним, что X и Y – это $M \times K$ массивы, где M – количество перекрывающихся ячеек, а K – размерность входных массивов (тензоров), исключая пространственные размеры (количество каналов в случае сцен Landsat 8).

Во время выполнения, пользователь может визуально отслеживать сходимость алгоритма с помощью интерактивной диаграммы, обновляемой на каждом шаге итерации, рис. 4.13a. Кроме того, коэффициенты также отображаются в консоли. Строка № i показывает корреляцию пары № i канонических переменных.

Параметры алгоритмов описаны в статье [59] и показаны в видео, разд. 4.4: (1) реализация CCA: предложенная или Python scikit-learn, и условие остановки: (2) порог корреляции (значимость изменения корреляций канонических переменных) или (3) максимальное количество итераций CCA.

Пользователь может видеть, что наша реализация CCA работает значительно быстрее по сравнению с Python scikit-learn и на синтетических и на реальных данных: она может работать примерно в 30 раз быстрее даже для двух перекрывающихся сцен Landsat 8 ($M \approx 6 \times 10^6$, $K = 7$), рис. 4.15, и может быть более 30 раз быстрее для выборок из нормального распределения, рис. 4.14.

Мы генерировали случайные входные переменные, используя нормальное распределение. Например, пусть X – случайная переменная, полученная из стандартного нормального распределения, где среднее равно 0, а дисперсия равна 1: $X \sim N(0, 1)$. Другую переменную мы можем построить следующим образом: $Y = X + Z$, где $Z \sim N(0, 0.5)$ (параметры можно варьировать без существенного влияния на скорость работы). Мы также варьировали M, количество входных векторов размером K (размер выборки) и установили $K = 8$ (напомним, что мы используем M и K вместо N и k, принятых в [59], поскольку N обозначает количество размерностей тен-

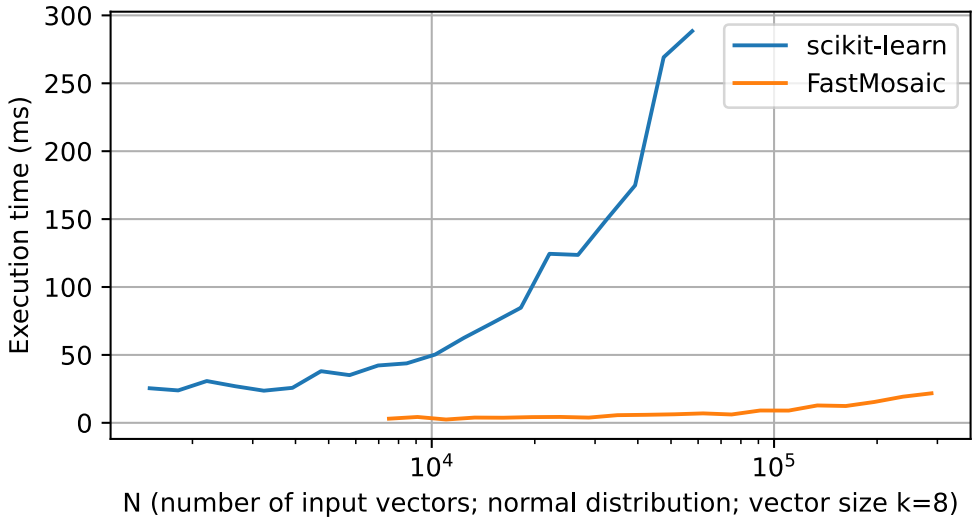


Рис. 4.14: CCA: FastMosaic VS. Python's "scikit-learn-[59]

зора в данной диссертации), рис. 4.14**. Для $M = 10^6$, Python's CCA и FastMosaic работают 5056.7 мс и 127.8 мс, соответственно; отношение составляет 39.57.

Параметр толерантности по умолчанию в реализации Python's CCA значительно замедляет ее работу, поэтому в наших экспериментах мы установили для толерантности значение $0,01$ (такое же, что и для FastMosaic). Последующее увеличение толерантности вряд ли приемлемо. Эти же параметры использовались для сравнения производительности Python's CCA и FastMosaic на сценах Landsat 8. Пусть U_P, V_P и U_F, V_F – пары канонических переменных, сгенерированные Python's CCA и FastMosaic соответственно. Мы обнаружили, что $|corr(U_P, U_P) - corr(U_F, U_F)| < 10^{-u}$ для $u = 2$ и выше, где $u \in \mathbb{Z}$ и для синтетических, и для реальных данных. Это подтверждает, что оба метода имеют сопоставимое качество выходных данных (канонических переменных).

Корреляции переменных можно проследить с помощью интерактивной диаграммы, рис. 4.13a. Значения корреляций и коэффициенты, сгенерированные пошагово, также можно найти в видеоролике FastMosaic, см. статью № 8.

Поскольку мы вычислили массив $P(\text{no change})$, мы можем сгенерировать соответствующую тепловую карту, чтобы тщательно исследовать ячейки в области перекрытия, которые, вероятно, являются инвариантными, рис. 4.13b.

FastMosaic сохраняет тепловую карту в формате GeoTIFF, что позволяет интерактивно исследовать ее в ГИС (географической информационной системе), например, в популярной Quantum GIS.

**Intel Core i7 2.6 GHz, 64 GB RAM, 512 GB NVMe, Windows 10, Sklearn (scikit-learn) version 1.4.0.

4.4.3 Трансформация (нормализация)

Наконец, мы можем построить мозаику. Поскольку у нас есть 2 входных массива (тензора) для пошагового режима, нам нужно всего K пар коэффициентов $\beta_k, \epsilon_k \in \mathbb{R}$ для преобразования массива (тензора)-субъекта $B\langle lat, lon, K\rangle$ (справа, рис. 4.15) $\beta_z B[x, y, z] + \epsilon_z$ ($z, k \in [1, K]$, разд. 2.5.2) для создания бесшовной мозаики.

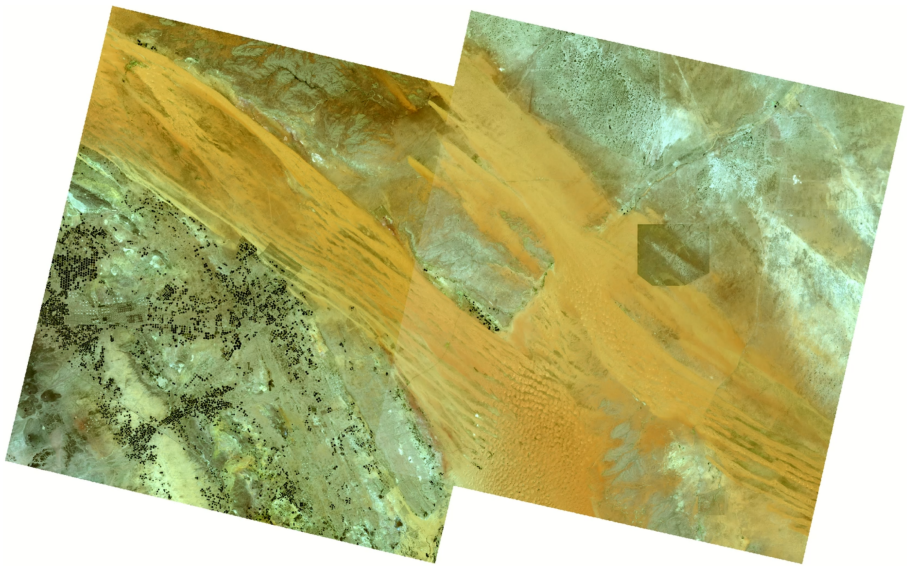


Рис. 4.15: Входные спутниковые сцены (пути 167 и 168, ряд 41): RGB

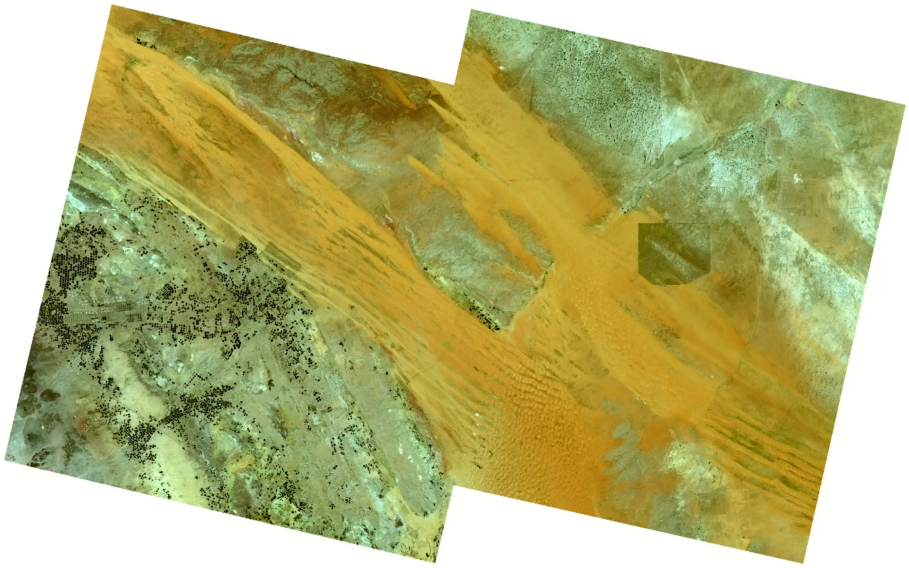


Рис. 4.16: Бесшовная мозаика массивов (тензоров) для входных данных на рис. 4.15

Очевидно, что визуальный переход между сценами является резким и напоминает шов, рис. 4.15. Построить качественную мозаику довольно сложно: простой настройки контраста недостаточно, чтобы это исправить. Очевидно, что FastMosaic обладает масштабируемыми CCA, MAD, IR-MAD для быстрого построения бесшовных мозаик, которые выглядят единым непрерывным изображением в естественных цветах, рис. 4.16.

Глава 5

Заключение

В заключении излагаются итоги выполненного исследования, рекомендации, перспективы дальнейшей разработки темы.

В области ТСУБД мы заложили новые теоретические основы, представили новые архитектурные и реализационные аспекты, а также продемонстрировали значимость нашего вклада на реальных данных и важных практических приложениях. Результаты, включенные в данную диссертацию, представлены на ведущих международных конференциях по компьютерным наукам: VLDB и SIGMOD.

Подробный список результатов диссертации изложен в разд. 1.3. Основные положения, выносимые на защиту, также находятся в разд. 1.3.

Мы уже отмечали, что область ТСУБД по праву является молодой, поэтому работа в этой области только началась. ТСУБД могли бы учитывать другие типы данных, например, пространственные полигоны, реляционные таблицы и графы, или работать в polystore-системах. Большего внимания требует использование новых аппаратных средств, например, NVM и GPU. Одним из наиболее перспективных направлений R&D является изучение новых приложений ТСУБД, подобных моделированию. Приложения ставят особые задачи перед ТСУБД и помогают стать им более робастными системами в целом.

Data Science и Machine Learning только прокладывают себе путь к ТСУБД, одним из главных преимуществ которых является нативная поддержка тензоров. Привлекательно запускать DS/ML внутри ТСУБД, чтобы избежать дорогостоящего обмена данными с системами DS/ML.

Сейчас сложились наилучшие условия для того, чтобы начать вносить свой вклад в R&D Растровых (Тензорных) СУБД (ТСУБД).

Список литературы

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, et al. TensorFlow: a system for large-scale machine learning. In *OSDI*, pages 265–283, 2016.

[2] K. A. Al-Gaadi, A. A. Hassaballa, E. Tola, et al. Prediction of potato crop yield using precision agriculture techniques. *Plos One*, 11(9):e0162219, 2016.

[3] AnyLogic. anylogic.com/road-traffic, 2024.

[4] ArcGIS book. learn.arcgis.com/en/arcgis-imagery-book, 2024.

[5] Arkansas River. newworldencyclopedia.org/entry/Arkansas_River, 2024.

[6] V. Balaji, A. Adcroft, and Z. Liang. Gridspec: a standard for the description of grids used in Earth system models. *arXiv preprint arXiv:1911.08638*, 2019.

[7] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, pages 1363–1375, 2016.

[8] P. Baumann and S. Holsten. A comparative analysis of array models for databases. *Int. J. Database Theory Appl.*, 5(1):89–120, 2012.

[9] P. Baumann, D. Misev, V. Merticariu, and B. P. Huu. Array databases: concepts, standards, implementations. *Journal of Big Data*, 8(1):1–61, 2021.

[10] P. Baumann, D. Misev, V. Merticariu, B. P. Huu, and B. Bell. DataCubes: a technology survey. In *IGARSS*, pages 430–433. IEEE, 2018.

[11] S. Blanas, K. Wu, S. Byna, B. Dong, and A. Shoshani. Parallel data analysis directly on scientific file formats. In *SIGMOD*, pages 385–396, 2014.

[12] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, and J. Saltz. Titan: a high-performance remote-sensing database. In *ICDE*, pages 375–384, 1997.

[13] D. Choi, H. Yoon, and Y. D. Chung. Resky: efficient subarray skyline computation in array databases. *Distributed and Parallel Databases*, 40(2-3):261–298, 2022.

[14] D. Choi, H. Yoon, and Y. D. Chung. Subarray skyline query processing in array databases. In *SSDBM*, pages 37–48, 2021.

[15] P. Cudré-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, et al. A demonstration of SciDB: a science-oriented DBMS. *PVLDB*, 2(2):1534–1537, 2009.

[16] V. S. da Silva, G. Salami, M. I. O. da Silva, E. A. Silva, J. J. Monteiro Junior, and E. Alba. Methodological evaluation of vegetation indexes in land use and land cover (LULC) classification. *Geology, Ecology, and Landscapes*, 4(2):159–169, 2020.

[17] D. J. DeWitt et al. Client-server Paradise. In *VLDB*, pages 558–569, 1994.

[18] Earth on AWS. https://aws.amazon.com/earth/, 2024.

[19] ECWMF report. https://www.ecmwf.int/en/computing/our-facilities/data-handling-system, 2022.

[20] GML. https://gephi.org/users/supported-graph-formats/, 2024.

[21] A. T. Hammad and G. Falchetta. Probabilistic forecasting of remotely sensed cropland vegetation health and its relevance for food security. *Science of the Total Environment*, 838:156157, 2022.

[22] O. Horlova, A. Kaitoua, and S. Ceri. Array-based data management for genomics. In *ICDE*, pages 109–120, 2020.

[23] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[24] C. E. Kilsedar and M. A. Brovelli. Multidimensional visualization and processing of big open urban geospatial data on the web. *ISPRS International Journal of Geo-Information*, 9(7):434, 2020.

[25] B. Kim, K. Koo, U. Enkhbat, S. Kim, J. Kim, and B. Moon. M2bench: a database benchmark for multi-model analytic workloads. *PVLDB*, 16(4):747–759, 2022.

[26] S. Ladra, J. R. Paramá, and F. Silva-Coira. Scalable and queryable compressed storage structure for raster data. *Information Systems*, 72:179–204, 2017.

[27] Landsat missions. https://landsat.usgs.gov/, 2024.

[28] É. Leclercq et al. Polystore and tensor data model for logical data independence and impedance mismatch in big data analytics. In *LNCS*, pages 51–90. 2019.

[29] X. Li, R. Feng, X. Guan, et al. Remote sensing image mosaicking: achievements and challenges. *IEEE Geoscience and Remote Sensing Magazine*, 7(4):8–22, 2019.

[30] L. Libkin, R. Machlin, and L. Wong. A query language for multidimensional arrays: design, implementation, and optimization techniques. In *ACM SIGMOD Record*, volume 25 of number 2, pages 228–239, 1996.

[31] B. A. Lungisani, C. K. Lebekwe, A. M. Zungeru, and A. Yahya. The current state on usage of image mosaic algorithms. *Scientific African*:e01419, 2022.

[32] S. Maerivoet and B. De Moor. Cellular automata models of road traffic. *Physics reports*, 419(1):1–64, 2005.

[33] A. P. Marathe and K. Salem. Query processing techniques for arrays. *VLDBJ*, 11(1):68–91, 2002.

[34] Maxar AWS re:Invent, 80 TB/day. https://youtu.be/mkKkSRIxU8M, 2017.

[35] V. Mazzia, L. Comba, et al. UAV and machine learning based refinement of a satellite-driven vegetation index for precision agriculture. *Sensors*, 20(9):2530, 2020.

[36] P. Mehta, S. Dorkenwald, D. Zhao, et al. Comparative evaluation of big-data systems on scientific image analytics workloads. *PVLDB*, 10(11):1226–1237, 2017.

[37] Milliseconds make millions. https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Consulting/Milliseconds_Make_Millions_report.pdf, 2020.

[38] NASA EO. earthobservatory.nasa.gov/images/145108/floods-in-the-arkansas-river-watershed, 2019.

[39] S. Nativi, J. Caron, B. Domenico, and L. Bigagli. Unidata's common data model mapping to the ISO 19123 data model. *Earth Sci. Inform.*, 1:59–78, 2008.

[40] NCEP-DOE AMIP-II Reanalysis. http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis2.html, 2024.

[41] NCO. http://nco.sourceforge.net/, 2024.

[42] Oracle database release. https://docs.oracle.com/en/database/oracle/oracle-database/21/geors/image-processing-virtual-mosaic.html, 21c.

[43] Oracle SG. oracle.com/database/technologies/spatialandgraph.html, 2024.

[44] C. Ordonez, Y. Zhang, and S. L. Johnsson. Scalable machine learning computing a data summarization matrix with a parallel array DBMS. *Distributed and Parallel Databases*, 37(3):329–350, 2019.

[45] I. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[46] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson. The TileDB array data storage manager. *PVLDB*, 10(4):349–360, 2016.

[47] PostGIS. http://postgis.net/, 2024.

[48] RasDaMan home. http://rasdaman.org/, 2024.

[49] RasDaMan mosaic. https://doc.rasdaman.org/05_geo-services-guide.html#data-import-recipe-mosaic-map, 2024.

[50] J. A. Richards. *Remote Sensing Digital Image Analysis: An Introduction*. Springer-Verlag Berlin Heidelberg, 5th edition, 2013.

[51] R. A. Rodriges Zalipynis. Array DBMS in environmental science: satellite sea surface height data in the cloud. In *IDAACS*, pages 1062–1065. IEEE, 2017.

[52] R. A. Rodriges Zalipynis. Array DBMS: past, present, and (near) future. *PVLDB*, 14(12):3186–3189, 2021.

[53] R. A. Rodriges Zalipynis. BitFun: fast answers to queries with tunable functions in geospatial array DBMS. *PVLDB*, 13(12):2909–2912, 2020.

[54] R. A. Rodriges Zalipynis. ChronosDB: distributed, file based, geospatial array DBMS. *PVLDB*, 11(10):1247–1261, 2018.

[55] R. A. Rodriges Zalipynis. ChronosDB in action: manage, process, and visualize big geospatial arrays in the Cloud. In *SIGMOD*, pages 1985–1988, 2019.

[56] R. A. Rodriges Zalipynis. Convergence of array DBMS and cellular automata: a road traffic simulation case. In *SIGMOD*, pages 2399–2403, 2021.

[57] R. A. Rodriges Zalipynis. Distributed in situ processing of big raster data in the Cloud. In volume 10742 of *LNCS*, pages 337–351. Springer, 2017.

[58] R. A. Rodriges Zalipynis. Evaluating array DBMS compression techniques for big environmental datasets. In *IDAACS*, volume 2, pages 859–863, 2019.

[59] R. A. Rodriges Zalipynis. FastMosaic in action: a new mosaic operator for Array DBMSs. *PVLDB*, 16(12):3938–3941, 2023.

[60] R. A. Rodriges Zalipynis. Generic distributed in situ aggregation for earth remote sensing imagery. In volume 11179 of *LNCS*, pages 331–342. Springer, 2018.

[61] R. A. Rodriges Zalipynis. SimDB in action: road trafic simulations completely inside Array DBMS. *PVLDB*, 15(12):3742–3745, 2022.

[62] R. A. Rodriges Zalipynis. Towards machine learning in distributed array DBMS: networking considerations. In volume 12629 of *LNCS*, pages 284–304. Springer, 2021.

[63] R. A. Rodriges Zalipynis and N. Terlych. WebArrayDB: A geospatial array DBMS in your web browser. *PVLDB*, 15(12):3622–3625, 2022.

[64] R. A. Rodriges Zalipynis et al. Array DBMS and satellite imagery: towards big raster data in the Cloud. In volume 10716 of *LNCS*, pages 267–279. Springer, 2018.

[65] R. A. Rodriges Zalipynis et al. Retrospective satellite data in the cloud: an array DBMS approach. In volume 793 of *CCIS*, pages 351–362. Springer, 2017.

[66] F. Rusu. Multidimensional array data management. *Foundations and Trends in Databases*, 12(2-3):69–220, 2023.

[67] Sentinel data access annual report. https://sentinels.copernicus.eu/web/sentinel/-/copernicus-sentinel-data-access-annual-report-2021, 2021.

[68] Sentinel Hub. https://www.sentinel-hub.com/, 2024.

[69] W. E. Splinter. Center-pivot irrigation. *Scientific American*, 234(6):90–99, 1976.

[70] D. C. Tomlin. *Geographic Information Systems and Cartographic Modeling*. Prentice-Hall, 1990.

[71] A. van Ballegooij. RAM: a multidimensional array DBMS. In *EDBT*, volume 3268, pages 154–165, 2004.

[72] S. Villarroya and P. Baumann. A survey on machine learning in array databases. *Applied Intelligence*, 53(9):9799–9822, 2023.

[73] S. Villarroya and P. Baumann. On the integration of machine learning and array databases. In *ICDE*, pages 1786–1789, 2020.

[74] W. Wen et al. A review of remote sensing challenges for food security with respect to salinity and drought threats. *Remote Sensing*, 13(1):6, 2020.

[75] WMTS. https://www.opengeospatial.org/standards/wmts, 2024.

[76] H. Xing and G. Agrawal. Accelerating array joining with integrated value-index. In *SSDBM*, pages 145–156, 2020.

[77] H. Xing and G. Agrawal. COMPASS: compact array storage with value index. In *SSDBM*, pages 1–12, 2018.

[78] J. Xue and B. Su. Significant remote sensing vegetation indices: a review of developments and applications. *Journal of Sensors*, 2017.

[79] M.-D. Yang, H.-H. Tseng, Y.-C. Hsu, and H. P. Tsai. Semantic segmentation using deep learning with vegetation indices for rice lodging identification in multi-date UAV visible images. *Remote Sensing*, 12(4):633, 2020.

[80] W. Zhao, F. Rusu, B. Dong, and K. Wu. Similarity join over array data. In *SIGMOD*, pages 2007–2022, 2016.

[81] W. Zhao, F. Rusu, B. Dong, K. Wu, A. Y. Ho, and P. Nugent. Distributed caching for processing raw arrays. In *SSDBM*, pages 1–12, 2018.

[82] W. Zhao, F. Rusu, B. Dong, K. Wu, and P. Nugent. Incremental view maintenance over array data. In *SIGMOD*, pages 139–154, 2017.