KIMYO INTERNATIONAL UNIVERSITY IN

CENTRAL ASIAN JOURNAL OF STEM

ISSN 2181-2934 http://stem.kiut.uz/

УДК 004.65:004.75

https://doi.org/10.5281/zenodo.21883458

АДАПТИВНЫЙ МЕТОД ФИЗИЧЕСКОЙ ОРГАНИЗАЦИИ ДАННЫХ В ОЗЕРЕ ДАННЫХ ТЕЛЕКОММУНИКАЦИОННОГО ОПЕРАТОРА

Адилов Рустам

студент бакалавриата, кафедра компьютерных наук

Университет ИНХА в Ташкенте

ORCID: 0009-0004-7250-4627

Email: rstmadylov@gmail.com

Абдуллоев Солех

студент бакалавриата, кафедра компьютерных наук

Университет ИНХА в Ташкенте

ORCID: 0009-0002-8387-635X

Email: solekhabdulloev@gmail.com

Аннотация – Рост объёмов данных детализации соединений (CDR) у операторов связи привёл к переходу от классических хранилищ к озёрам данных на колоночных форматах с отсечением файлов по блочной статистике. Эффективность отсечения полностью определяется физической раскладкой данных — набором ключей секционирования, порядком записей внутри секции и целевым размером файла. На практике эти параметры задаются вручную и по отдельности, что приводит либо к недостаточной кластеризации, либо

к деградации из-за избыточного числа мелких файлов. В работе решается задача совместного выбора трёх параметров раскладки по наблюдаемому профилю аналитической нагрузки. Предложена аддитивная модель стоимости запроса, учитывающая, помимо объёма сканирования, поштучные расходы на открытие файлов и расходы на планирование по манифесту таблицы, не зависящие от селективности. Для многомерного упорядочивания по кривой Мортона выведена аналитическая оценка доли просматриваемых файлов, содержащая слагаемое граничного эффекта и объясняющая ухудшение отсечения при росте размерности порядка. Показано, что оценка систематически завышает объём сканирования для столбцов, функционально связанных с ключами упорядочивания; предложена поправка через индуцированную селективность с фильтрацией по силе функциональной зависимости. На основе модели построен алгоритм ADPC, включающий правило запуска компактизации по накопленному регрету с явной точкой безубыточности. Вычислительный эксперимент на смоделированном CDR-потоке ($3 \cdot 10^6$ записей, десять запросов с весами) показал: раскладка, выбранная алгоритмом, сокращает средневзвешенный объём сканирования в 2,73 раза и модельное время отклика в 2,55 раза относительно посуточного секционирования. Избыточное секционирование ухудшает время отклика на 7,8 % несмотря на меньший объём сканирования. Зависимость времени отклика от целевого размера файла немонотонна с минимумом при 16–32 МБ. Предложенная поправка снижает среднюю относительную ошибку модели с 70,4 % до 19,7 %.

Ключевые слова: большие данные, озеро данных, отсечение данных, кривая Мортона, секционирование, модель стоимости запроса, компактизация, .

ADAPTIVE PHYSICAL DATA LAYOUT METHOD FOR A TELECOM OPERATOR’S DATA LAKE

Abstract – The growth of Call Detail Record (CDR) volumes at telecom operators has driven a shift from classical warehouses to data lakes built on columnar formats with file-level data skipping. Skipping efficiency is entirely determined by the physical data layout: the set of partition keys, the record order within a partition, and the target file size. In practice these parameters are chosen manually and independently, which leads either to insufficient clustering or to degradation caused by an excessive number of small files. This paper addresses the joint selection of all three layout parameters from the observed analytical workload profile. An additive query cost model is proposed that accounts, beyond scan volume, for per-file open overhead and for manifest planning cost that does not depend on selectivity. For multidimensional Morton-curve ordering, an analytical estimate of the scanned file fraction is derived, containing a boundary-effect term that explains the degradation of skipping as the order dimensionality grows. The estimate is shown to systematically overstate scan volume for columns functionally dependent on the ordering keys; a correction based on induced selectivity with filtering by functional-dependency strength is proposed. Building on the model, the ADPC algorithm is developed, including a compaction trigger rule based on accumulated regret with an explicit break-even horizon. A computational experiment on a simulated CDR stream ($3 \cdot 10^6$ records, ten weighted queries) shows that the layout selected by the algorithm reduces the weighted scan volume by a factor of 2.73 and modelled response time by a factor of 2.55 relative to daily partitioning. Over-partitioning degrades response time by 7.8 % despite a smaller scan volume. Response time is non-monotonic in target file size, with a minimum at 16–32 MB. The proposed correction reduces the model’s mean relative error from 70.4 % to 19.7 %.

Keywords: big data, data lake, data skipping, Morton curve, partitioning, query cost model, compaction, call detail records.

ТЕЛЕКОММУНИКАЦИЯ ОПЕРАТОРИ МАЪЛУМОТЛАР КЎЛИДА МАЪЛУМОТЛАРНИ ФИЗИК ТАШКИЛ ЭТИШНИНГ АДАПТИВ УСУЛИ

Аннотация – Алоқа операторларида уланишлар тафсилоти маълумотлари (CDR) ҳажмининг ўсиши классик омборхоналардан устунли форматлар асосидаги ва блок статистикаси бўйича файлларни кесиб ташлаш имконини берувчи маълумотлар кўлига ўтишга олиб келди. Кесиб ташлаш самарадорлиги тўлиқ равишда маълумотларнинг физик жойлашуви — секцияловчи калитлар тўплами, секция ичидаги ёзувлар тартиби ва файлнинг мақсадли ҳажми билан белгиланади. Амалиётда бу параметрлар қўлда ва алоҳида-алоҳида танланади, бу эса ёки етарли бўлмаган кластерлашга, ёки майда файлларнинг ортиқча сони туфайли деградацияга олиб келади. Ишда жойлашувнинг учта параметрини таҳлилий юклама профили бўйича биргаликда танлаш масаласи ечилади. Сканерлаш ҳажмидан ташқари файлларни очишга кетадиган харажатларни ва селективликка боғлиқ бўлмаган манифест бўйича режалаштириш харажатларини ҳисобга олувчи аддитив сўров нархи модели таклиф этилган. Мортон эгри чизиғи бўйича кўп ўлчовли тартиблаш учун чегаравий эффект ҳадини ўз ичига олган таҳлилий баҳо келтириб чиқарилган. Кўрсатилганки, баҳо тартиблаш калитлари билан функционал боғланган устунлар учун сканерлаш ҳажмини тизимли равишда ошириб юборади; индуцирланган селективлик орқали функционал боғлиқлик кучи бўйича фильтрлаш билан тузатиш таклиф этилган. Модель асосида ADPC алгоритми қурилган. Ҳисоблаш эксперименти ($3 \cdot 10^6$ ёзув, вазнли ўнта сўров) кўрсатдики, алгоритм танлаган жойлашув сканерлаш ҳажмини 2,73 марта, моделли жавоб вақтини 2,55 марта қисқартиради. Ортиқча секциялаш кичикроқ сканерлаш ҳажмига қарамай жавоб вақтини 7,8 % га ёмонлаштиради. Таклиф этилган тузатиш моделнинг ўртача нисбий хатосини 70,4 % дан 19,7 % гача камайтиради.

Калит сўзлар: катта маълумотлар, маълумотлар кўли, маълумотларни кесиб ташлаш, Мортон эгри чизиғи, секциялаш, сўров нархи модели, компактлаштириш, уланишлар тафсилоти.

1. ВВЕДЕНИЕ

Телекоммуникационный оператор национального масштаба генерирует поток записей детализации соединений (Call Detail Records, CDR) порядка 10^9 строк в сутки. Эти данные служат основой для тарификации, антифрод-анализа, планирования радиосети и для задач управления ценностью клиента (Customer Value Management), где на их базе строятся признаки моделей оттока и оценки эффекта маркетинговых кампаний. Требования к хранилищу противоречивы: удержание данных за 12–24 месяца при приемлемой стоимости хранения — и отклик интерактивных запросов в единицы секунд.

Классические реляционные хранилища на этих объёмах экономически неприемлемы, поэтому индустрия перешла к архитектуре озера данных: колоночный формат файлов (Parquet, ORC) поверх распределённой или объектной файловой системы, слой табличного

формата (Apache Iceberg, Delta Lake, Apache Hudi), обеспечивающий транзакционность и хранение статистики, и отделённый вычислительный слой (Apache Spark, Trino).

Ключевой механизм производительности в такой архитектуре — отсечение данных (data skipping): для каждого файла хранится блочная статистика (минимум и максимум по каждому столбцу, zone map), и планировщик исключает файл из чтения, если диапазон его значений заведомо не пересекается с предикатом запроса. Эффективность отсечения не является свойством формата — она определяется тем, как записи распределены по файлам, то есть физической раскладкой данных.

Раскладка задаётся тремя параметрами: множеством ключей секционирования K, определяющим иерархию каталогов; множеством ключей упорядочивания Z, задающим порядок записей внутри секции; и целевым размером файла τ. На практике эти параметры выбираются администратором вручную, независимо друг от друга и, как правило, однократно при проектировании таблицы. Такой подход порождает два хорошо известных антипаттерна. Первый — недостаточная кластеризация: секционирование только по дате оставляет запросы с непространственно-временными предикатами (например, по когорте абонентов) без отсечения вовсе. Второй — избыточное секционирование: попытка включить в K все часто фильтруемые столбцы дробит таблицу на десятки тысяч мелких файлов, и выигрыш от отсечения перекрывается накладными расходами на планирование и открытие файлов.

Научная значимость задачи в том, что выбор раскладки — задача оптимизации на дискретно-непрерывном пространстве при заданном распределении нагрузки, и её корректная постановка требует модели стоимости, связывающей статистические характеристики данных и предикатов с наблюдаемым временем отклика.

Вклад работы состоит в следующем.

1. Аддитивная модель стоимости запроса в озере данных, учитывающая объём сканирования, поштучные расходы на открытие файлов и расходы на планирование по манифесту.
2. Аналитическая оценка доли просматриваемых файлов при упорядочивании по кривой Мортона с явным слагаемым граничного эффекта при чётко очерченных допущениях.
3. Поправка оценки для функционально связанных столбцов через индуцированную селективность с фильтрацией по силе функциональной зависимости; показано, что без фильтрации поправка срабатывает на случайных совпадениях значений и переоценивает отсечение.
4. Алгоритм ADPC совместного выбора трёх параметров раскладки и правило запуска компактизации по накопленному регрету с точкой безубыточности.
5. Вычислительный эксперимент, количественно подтверждающий выигрыш и демонстрирующий немонотонность стоимости по целевому размеру файла.

2. ОБЗОР ЛИТЕРАТУРЫ

Идея хранения агрегированной статистики блоков для отсечения восходит к работе Моеркотте о малых материализованных агрегатах [1], где показано, что лёгкая поблочная статистика даёт основную часть выигрыша полноценных индексов при пренебрежимой стоимости сопровождения. Колоночное хранение как основа аналитических систем

исследовано в [2, 3]; обзор конструктивных компромиссов между чтением, обновлением и памятью систематизирован в гипотезе RUM [4].

Промышленные реализации отсечения по zone map описаны в работах о Snowflake [5], Dremel [6] и Delta Lake [7]. В [7] отмечается, что многомерная кластеризация по кривым, сохраняющим локальность, существенно повышает долю отсекаемых файлов, однако выбор набора ключей оставляется пользователю. Кривая Мортона предложена в [8], сравнение с кривой Гильберта для многомерных запросов проведено в [9].

Автоматизированный выбор физического дизайна имеет длительную историю в реляционных СУБД: инструменты выбора индексов и материализованных представлений по журналу нагрузки [10, 11] строятся на модели стоимости оптимизатора и переборе с отсечением. Эти работы ориентированы на структуры, внешние по отношению к данным, тогда как в озере данных основной рычаг — сам порядок размещения записей.

Наиболее близкое направление — оптимизация раскладки под нагрузку. В [12] предложено мелкогранулярное разбиение блоков по предикатам журнала запросов; метод даёт агрессивное отсечение, но требует явного перечисления предикатов и плохо обобщается на ранее не встречавшиеся запросы. Обучаемые многомерные индексы Flood [13] и Tsunami [14] подстраивают разбиение пространства под нагрузку и перекос данных, продолжая линию обучаемых структур доступа [15]; они предполагают контроль над структурой индекса, которого в озере данных с иммутабельными файлами нет. Динамическая адаптация порядка данных к нагрузке рассматривалась в парадигме «взлома базы данных» [16], где сортировка выполняется как побочный эффект запросов.

Отдельный пласт составляют инженерные работы по экосистеме Hadoop [17, 18], потоковому приёму данных [19] и организации LSM-хранилищ [20], где задача компактизации ставится как компромисс между стоимостью записи и чтения. Общие принципы распределённой обработки больших данных изложены в [21, 22], векторизованное исполнение запросов — в [23].

Перечисленные подходы оптимизируют один параметр раскладки: либо разбиение [12–14], либо порядок [16], либо политику слияния файлов [20]. Совместная оптимизация тройки (K, Z, τ) с явным учётом накладных расходов на число файлов в известных нам работах не формализована, хотя именно взаимодействие этих параметров порождает описанные антипаттерны: увеличение мощности K улучшает точное отсечение, но сокращает число файлов в секции и тем самым ухудшает отсечение по Z, одновременно увеличивая общее число файлов.

3. ПОСТАНОВКА ЗАДАЧИ

Пусть таблица D содержит множество записей над схемой атрибутов A, объём таблицы равен S. Раскладка

$$L = (K,\ Z,\ \tau), \qquad K \subseteq A, \quad Z \subseteq A \backslash K, \quad \tau > 0 \qquad (1)$$

определяет разбиение D на множество иммутабельных файлов $F(L)$: записи группируются по значению ключей K (секции), внутри секции упорядочиваются по ключу Мортона, построенному над Z, и последовательно нарезаются на файлы целевого размера τ. Для каждого файла f и каждого столбца c хранится пара минимума и максимума.

Рабочая нагрузка задана распределением на множестве запросов:

$$W = \{(q_i,\ w_i)\}, \qquad \sum_i w_i = 1 \qquad (2)$$

Каждый запрос q характеризуется множеством конъюнктивных диапазонных предикатов $P(q)$ и селективностями $\sigma(q,c)$ — долей строк таблицы, удовлетворяющих предикату по столбцу c.

Файл f просматривается при выполнении q, если его блочная статистика не позволяет доказать пустоту пересечения:

$$\pi_q(f) = \prod_{(c,\,lo,\,hi)\,\in\,P(q)} \mathbf{1}\left[\max_c(f) \geq lo\ \wedge\ \min_c(f) \leq hi\right] \qquad (3)$$

Объём сканирования и число просмотренных файлов:

$$C_{\text{scan}}(q,L) = \sum_{f\in F(L)} s\,(f)\,\pi_q(f), \qquad n_f(q,L) = \sum_{f\in F(L)} \pi_q\,(f) \qquad (4)$$

Задача состоит в отыскании раскладки, минимизирующей математическое ожидание стоимости обслуживания нагрузки:

$$L^* = \operatorname*{argmin}_L J(L), \qquad J(L) = \sum_i w_i\ T(q_i,L) \qquad (5)$$

при ограничении на допустимое число файлов и на бюджет перестроения. Задача (5) дискретно-непрерывна и переборно неразрешима: число кандидатов на пару (K, Z) экспоненциально по мощности A. Дальнейшее изложение строит вычислимую аппроксимацию T и эвристику направленного поиска.

Модель стоимости

Время отклика запроса складывается из трёх различных по природе слагаемых: пропорционального объёму чтения, пропорционального числу открываемых файлов и пропорционального числу файлов в манифесте таблицы — стоимости планирования, оплачиваемой независимо от отсечения:

$$T(q,L) = \frac{C_{\text{scan}}(q,L)}{B} + c_0\left\lceil\frac{n_f(q,L)}{P}\right\rceil + c_{\text{plan}}\,N(L) \qquad (6)$$

где B — агрегированная пропускная способность сканирования кластера, P — число параллельных задач, c_0 — стоимость открытия файла (сетевой обмен, чтение подвала Parquet, планирование задачи), $c$plan — стоимость обработки одной записи манифеста при построении плана, $N(L)$ — общее число файлов таблицы.

Именно третье слагаемое, обычно игнорируемое при настройке, объясняет патологию избыточного секционирования: оно не зависит от селективности запроса и растёт линейно с числом файлов.

Отсечение по секционированию

Существенно, что секционирование даёт точное отсечение с гранулярностью секции, а не строки. Пусть V — множество секций, $S(v)$ — объём секции v, а $V(q)$ — множество секций, диапазон значений ключей K в которых пересекается с предикатами запроса. Тогда

$$\varphi_P(q) = \frac{1}{S} \sum_{v \in V(q)} S(v) \;\geq \prod_{c \in K \cap \mathrm{dom}\, P(q)} \sigma(q, c) \qquad (7)$$

Равенство в (7) достигается только тогда, когда границы предиката совпадают с границами секций. Например, при секционировании по суткам предикат на метку времени, покрывающий половину суток, читает секцию целиком: селективность равна 0,5, а доля сканирования — единице. Приближение через произведение селективностей, встречающееся в инженерной практике, допустимо лишь для выровненных по границам предикатов; в общем случае оно занижает стоимость.

Число секций и среднее число файлов в секции:

$$N_{\mathrm{part}} = |V|, \qquad \bar{n}_p = \max\left(1, \frac{S}{N_{\mathrm{part}}\, \tau}\right) \qquad (8)$$

Существенны две оговорки. Во-первых, Npart — число фактически встретившихся комбинаций значений ключей, которое может быть на порядки меньше произведения кардинальностей; последнее даёт лишь верхнюю границу. Во-вторых, при перекошенном распределении ключей размеры секций различаются на порядки, и корректной величиной является распределение числа файлов по секциям, а не его среднее; формулы следующего подраздела применяются посекционно, а свёртка ведётся с весами $S(v)/S$. Замена на среднее правомерна лишь при умеренном перекосе.

Из (8) следует условие сохранения целевого размера файла:

$$N_{\mathrm{part}} \leq \frac{S}{\tau} \qquad (9)$$

При нарушении (9) секции содержат менее одного полного файла: фактический размер файлов падает ниже τ, а общее число файлов стремится к числу секций. Это и есть формальное условие возникновения проблемы мелких файлов.

Отсечение по упорядочиванию Мортона

Пусть m — мощность Z, и внутри секции записи упорядочены по ключу Мортона над квантильно-нормированными значениями Z. Оценка строится при следующих допущениях: (а) координаты нормированы по рангам, поэтому селективность предиката измеряется в квантильной шкале и совпадает с долей строк; (б) множество файлов секции идеализируется регулярной сеткой ячеек в нормированном m-мерном кубе — точное соответствие кривой Мортона регулярной сетке имеет место при числе файлов, равном степени 2^m; (в) предикаты диапазонные и по разным измерениям независимы.

При этих допущениях ячейка сетки имеет сторону, равную числу файлов секции в степени минус один на m. Запрос задаёт гиперпрямоугольник со сторонами $\sigma(j)$; число пересекаемых им ячеек вдоль измерения j даёт долю $\sigma(j)$ плюс сторона ячейки. Для измерений, не ограниченных предикатом, доля равна единице. Отсюда

$$\varphi_Z(q) = \prod_{j \in Z \cap \mathrm{dom}\, P(q)} \min\left(1,\ \sigma_j + n_p^{-1/m}\right) \qquad (10)$$

Слагаемое в степени минус один на m — граничный эффект: даже при бесконечно селективном предикате доля просматриваемых файлов не опускается ниже этой величины. Из (10) следуют два практически важных вывода. Во-первых, при $m = 1$ формула вырождается в сумму селективности и обратного числа файлов, то есть одномерная сортировка почти идеальна для «своего» столбца. Во-вторых, добавление каждого нового измерения в Z увеличивает граничный член для всех измерений, то есть улучшает отсечение для новых предикатов ценой ухудшения для прежних. Это объясняет наблюдаемую на практике деградацию при включении в порядок более трёх-четырёх столбцов.

Итоговая оценка доли сканирования есть произведение (7) и (10).

Поправка на функциональные связи столбцов

Оценка (10) учитывает только предикаты, столбцы которых явно входят в Z. Реальные схемы CDR содержат сильные функциональные зависимости: идентификатор соты однозначно определяет регион. Если столбец c, не входящий ни в K, ни в Z, функционально определяется столбцом z из Z, то предикат по c косвенно ограничивает и z, а значит отсечение происходит и по нему. Базовая оценка этого не учитывает и систематически завышает объём сканирования.

Введём индуцированную селективность предиката относительно ключа z:

$$\tilde{\sigma}_z(c, lo, hi) = \frac{|\{r \in D:\ r.z \in Z_c\}|}{|D|}, \qquad Z_c = \{r.z:\ lo \le r.c \le hi\} \qquad (11)$$

то есть доля записей, значение ключа z которых встречается хотя бы у одной записи, удовлетворяющей предикату.

Применять (11) без ограничений нельзя. Если z имеет высокую кардинальность, множество $Z(c)$ оказывается малым просто в силу разреженности выборки, а не из-за реальной зависимости, и поправка ошибочно предсказывает отсечение. Поэтому введём меру силы функциональной зависимости:

$$\delta(z \to c) = \frac{|\{\text{distinct } z\}|}{|\{\text{distinct } (z, c)\}|} \in (0,\ 1] \qquad (12)$$

Величина δ равна единице тогда и только тогда, когда z функционально определяет c, и убывает по мере ослабления зависимости. Поправка применяется только к ключам, для которых δ не ниже порога δmin (в экспериментах 0,95):

$$\tilde{\varphi}_Z(q) = \prod_{j \in Z \cap \mathrm{dom}\, P(q)} \min\left(1, \sigma_j + n_p^{-1/m}\right) \cdot \prod_{c \in C^{fd}} \min\left(1,\ \min_{z \in Z_c^{\delta}} \tilde{\sigma}_z(c) + n_p^{-1/m}\right) \qquad (13)$$

где верхний индекс δ обозначает подмножество ключей Z, прошедших порог, а Cfd — множество столбцов предиката вне K и Z, для которых это подмножество непусто. Величины δ и индуцированные селективности вычисляются однократно по выборке данных и хранятся как часть профиля таблицы. Раздел результатов показывает, что фильтр (12) существенно повышает точность: без него поправка срабатывает на случайных совпадениях значений.

Алгоритм ADPC

Алгоритм 1. Adaptive Data Placement and Compaction

Вход: журнал запросов Q, выборка данных D_s, бюджет N_max
Выход: раскладка L* = (K*, Z*, tau*)

1. Извлечь из Q множество предикатов; оценить веса w_i и селективности sigma(q,c) по D_s.
2. Для каждого столбца c вычислить оценку полезности (14).
3. Сформировать K: включить столбцы с высоким b_c, низкой кардинальностью и монотонной динамикой (время события), соблюдая ограничение (9).
4. Сформировать Z_cand: все подмножества мощности 2..4 из top-5 столбцов по b_c, не вошедших в K.
5. Для каждой пары (Z, tau) из Z_cand x T_grid оценить J(L) по (5)-(7), (13); выбрать минимум.
6. Вычислить delta и sigma с тильдой, записать профиль таблицы.

Оценка полезности столбца на шаге 2 учитывает граничный эффект: селективность ниже граничного члена не даёт дополнительного отсечения, поэтому она обрезается снизу:

$$b_c = \sum_i w_i \; \mathbf{1}[c \in \operatorname{dom} P(q_i)] \ln \frac{1}{\max\left(\sigma(q_i, c),\; n_p^{-1/m}\right)} \qquad (14)$$

Логарифм возникает естественно: вклад столбцов в отсечение мультипликативен по (7) и (10), поэтому в аддитивной свёртке фигурирует логарифм селективности. Обрезка снизу даёт верхнюю границу вклада одного столбца — натуральный логарифм числа файлов секции, делённый на m, — то есть сверхселективный столбец не может доминировать в ранжировании произвольно сильно. Без обрезки точечные предикаты (например, поиск по идентификатору абонента с селективностью порядка 10^{-5}) получают завышенную оценку.

Величины числа файлов секции и m в (14) зависят от искомой раскладки, что формально делает оценку рекурсивной. На практике достаточно одной итерации: (14) вычисляется при начальном приближении, и результат используется только для сокращения пространства поиска на шаге 4. Окончательный выбор делается по полной модели стоимости на шаге 5. Эксперимент показывает, что жадный выбор по убыванию оценки полезности даёт субоптимальную раскладку, то есть шаг 5 не может быть заменён ранжированием.

Компактизация по регрету

Поток приёма данных непрерывно добавляет мелкие файлы, и фактическая раскладка отклоняется от оптимальной. Чтобы сравнивать выигрыш и затраты, обе величины выражаются в одних единицах — кластеро-секундах. Накопленный регрет за горизонт H при интенсивности запросов λ:

$$R(H) = \lambda H\big(J(L_t) - J(L^*)\big) = \lambda H\, \Delta J \qquad (15)$$

Стоимость перестроения объёма загрязнённых данных складывается из чтения и записи:

$$C_{\text{comp}} = S_{\text{dirty}} \left(\frac{1}{B_r} + \frac{1}{B_w}\right) \qquad (16)$$

Компактизацию рационально запускать при превышении регретом стоимости перестроения, откуда точка безубыточности — минимальный горизонт, оправдывающий перестроение:

$$H^* = \frac{S_{\text{dirty}}(1/B_r + 1/B_w)}{\lambda\, \Delta J} \qquad (17)$$

Правило (15)–(17) заменяет распространённые пороговые эвристики экономически обоснованным критерием: перестроение выполняется тогда и только тогда, когда ожидаемая экономия на запросах до следующего перестроения превышает стоимость перезаписи

4. АНАЛИЗ И РЕЗУЛЬТАТЫ

Постановка эксперимента

Проверялись три гипотезы: H1 — совместный выбор трёх параметров раскладки по модели стоимости даёт выигрыш относительно типовых раскладок; H2 — зависимость стоимости от целевого размера файла немонотонна и имеет внутренний минимум; H3 — поправка (13) с фильтром (12) существенно повышает точность модели.

Эксперимент выполнен на программной модели, воспроизводящей механику отсечения по блочной статистике. Моделируется таблица CDR объёмом 320 ГБ; для вычислительной осуществимости используется выборка $3 \cdot 10^6$ записей при сохранении числа файлов и профиля распределений. Доля просканированных данных инвариантна относительно такого масштабирования при фиксированном числе файлов, поскольку зависит только от селективностей и числа файлов.

Таблица 1. Профиль атрибутов

Атрибут	Кардинальность	Распределение
Метка времени	896 790	14 суток, суточная неравномерность с пиком 10:00–21:00
Сутки	14	равномерно
Регион	14	неравномерно, доля столичного региона 28 %
Сота	4 998	функционально вложена в регион
Абонент	1 170 530	степенное распределение активности
Тип услуги	8	неравномерно
Тарифный план	30	равномерно

Константы модели (6): B = 6 ГБ/с, c_0 = 12 мс, P = 200, $c$plan = 0,04 мс. Значения соответствуют типовому кластеру Spark поверх объектного хранилища и являются параметрами модели.

Таблица 2. Рабочая нагрузка

Запрос	Вес	Содержание
Q1	0,18	Суточные агрегаты за последние сутки
Q2	0,14	Сутки и регион

Запрос	Вес	Содержание
Q3	0,12	7 суток, один абонент (профиль 360°), селективность $9{,}0 \cdot 10^{-6}$
Q4	0,10	Сутки и 10 сот
Q5	0,08	3 суток, тип услуги, два региона
Q6	0,09	14 суток, тарифный план
Q7	0,07	Сутки, регион, тип услуги
Q8	0,08	7 суток, когорта абонентов (CVM)
Q9	0,05	Полное сканирование (обучение модели)
Q10	0,09	2 суток, диапазон сот

Сравниваемые раскладки: L0 — секционирование по суткам, порядок поступления (типовая базовая конфигурация); L1 — сутки и сортировка по региону; L2 — секционирование по суткам, региону и соте (типичный антипаттерн избыточного секционирования); L3 — сутки и Z-порядок по региону, соте, абоненту (экспертный выбор); L4 — раскладка, выбранная алгоритмом ADPC.

Результаты

Таблица 3. Оценки полезности (14) при начальном приближении

Столбец	Абонент	Метка времени	Сота	Регион	Тип услуги	Тарифный план
Оценка	1,973	1,730	0,891	0,527	0,310	0,306

Метка времени выбрана ключом секционирования как монотонно растущий атрибут с приемлемой кардинальностью при агрегации до суток; ограничение (9) выполняется с большим запасом (14 секций против 10 240). Поиск на шаге 5 (таблица 4) выбрал в качестве ключей упорядочивания абонента, соту и тарифный план при целевом размере файла 32 МБ, что даёт 734,3 файла на секцию и граничный член 0,111.

Таблица 4. Сетка поиска ADPC (фрагмент)

Ключи упорядочивания	τ, МБ	Файлов	Скан, %	Время, мс
абонент, сота	32	10 280	16,93	9 549
абонент, сота	64	5 136	17,27	9 478
абонент, тариф	64	5 136	13,29	7 340
абонент, сота, тариф	**32**	**10 280**	**11,17**	**6 446**
абонент, сота, тариф	128	2 567	13,61	7 390
абонент, сота, регион	32	10 280	17,39	9 803

Жадный выбор по убыванию оценки полезности дал бы набор «абонент, сота, регион» — раскладку, уступающую выбранной по времени отклика в 1,52 раза. Причина в функциональной вложенности соты в регион: мера зависимости равна единице, то есть включение обоих столбцов расходует измерение, не добавляя отсекающей способности, тогда как тарифный план статистически независим от прочих ключей и «раскрывает» запрос Q6.

Таблица 5. Сводные результаты

Раскладка	Файлов	Скан, %	Время, мс	Выигрыш по объёму	Выигрыш по времени
L0 сутки, порядок поступления	2 567	30,5	16 417	1,00	1,00
L1 сутки и сортировка по региону	2 567	26,0	14 042	1,17	1,17
L2 сутки, регион, сота	69 972	25,9	17 702	1,18	**0,93**
L3 сутки, Z(регион, сота, абонент)	2 567	18,0	9 738	1,69	1,69
L4 ADPC: сутки, Z(абонент, сота, тариф), 32 МБ	10 280	**11,2**	**6 446**	**2,73**	**2,55**

Таблица 6. Доля просканированных данных по запросам, %

Запрос	L0	L1	L2	L3	L4
Q1	7,14	7,14	7,14	7,14	7,14
Q2	7,14	2,03	2,00	2,58	2,86
Q3	49,99	49,99	49,99	6,62	9,76
Q4	7,14	0,55	**0,01**	0,94	1,08
Q5	21,42	3,39	3,21	7,32	8,72
Q6	100,00	100,00	100,00	100,00	**14,47**
Q7	7,14	0,47	0,43	1,64	1,86
Q8	49,99	49,99	49,89	7,05	10,67
Q9	100,00	100,00	100,00	100,00	100,00
Q10	14,29	1,36	0,73	2,30	2,50

Таблица 7. Время отклика по модели (6), мс

Запрос	L0	L2	L3	L4	L0 / L4
Q1	3 923	6 908	3 923	4 268	0,92
Q2	3 923	3 892	1 488	1 961	2,00
Q3	26 849	31 561	3 647	5 687	4,72
Q4	3 923	2 818	614	999	3,93
Q5	11 563	4 644	4 020	5 120	2,26
Q6	53 592	60 332	53 592	8 227	6,51
Q7	3 923	3 053	989	1 414	2,77
Q8	26 849	31 509	3 876	6 175	4,35
Q9	53 592	60 332	53 592	54 369	0,99
Q10	7 747	3 249	1 340	1 768	4,38

Таблица 8. Влияние целевого размера файла

τ, МБ	8	16	32	64	128	256	512
Файлов	41 103	20 555	10 280	5 136	2 567	1 288	644
Скан, %	9,62	10,31	11,17	12,34	13,61	15,31	17,35
Время, мс	7 017	6 455	**6 446**	6 832	7 390	8 235	9 293

Таблица 9. Мера силы функциональной зависимости по формуле (12)

Ключ / столбец	Регион	Тип услуги	Тарифный план	Сота
Абонент	0,470	0,517	0,421	0,391
Сота	**1,000**	0,125	0,033	—
Тарифный план	0,071	0,125	—	0,000

Таблица 10. Верификация аналитической модели (раскладка L4)

Запрос	Базовая (10)	Поправка без фильтра	Поправка с фильтром (13)	Измерено
Q1	0,0714	0,0714	0,0714	0,0714
Q2	0,0714	0,0279	0,0279	0,0286
Q3	0,0554	0,0554	0,0554	0,0976
Q4	0,0081	0,0081	0,0081	0,0108
Q5	0,2142	0,0198	0,0559	0,0872
Q6	0,1442	0,1442	0,1442	0,1447
Q7	0,0714	0,0095	0,0122	0,0186
Q8	0,0558	0,0558	0,0558	0,1067
Q9	1,0000	1,0000	1,0000	1,0000
Q10	0,0231	0,0231	0,0231	0,0250
Средняя абсолютная ошибка	0,0321	0,0175	**0,0137**	—
Средняя относительная ошибка	70,4 %	25,3 %	**19,7 %**	—
Максимальная ошибка	0,1270	0,0673	**0,0509**	—

Обсуждение

Гипотеза H1 подтверждена. Раскладка, выбранная алгоритмом, сокращает средневзвешенный объём сканирования в 2,73 раза и модельное время отклика в 2,55 раза относительно базовой конфигурации. Для отдельных классов запросов выигрыш существенно выше: анализ по тарифному плану ускоряется в 6,51 раза, профиль абонента — в 4,72 раза, выборка когорты для оценки эффекта кампании — в 4,35 раза.

Существенно, что выигрыш достигается не бесплатно. Из таблицы 7 видно, что запросы без селективных предикатов — суточный агрегат Q1 и полное сканирование Q9 — замедляются на 8,8 % и 1,4 % соответственно: они не выигрывают от отсечения, но оплачивают возросшее число файлов через второе и третье слагаемые в (6). Оптимизация раскладки, таким образом, перераспределяет стоимость от селективных запросов к сканирующим, и её целесообразность

определяется составом нагрузки. При доле тяжёлой пакетной аналитики выше некоторого порога приоритет смещается к материализованным агрегатам.

Патология избыточного секционирования воспроизведена количественно. Раскладка L2, дробящая таблицу на 69 972 файла, сканирует на 15,1 % меньше данных, чем базовая, но по времени отклика уступает ей на 7,8 %. Причина видна из (6): постоянное слагаемое стоимости планирования добавляет 2,80 с к каждому запросу, включая точечные, где оно многократно превышает время полезного чтения. Так, для Q4 объём сканирования падает в 714 раз, а время отклика — лишь в 1,39 раза. Это показывает, что оптимизация раскладки по объёму сканирования — распространённый на практике критерий — некорректна.

Гипотеза H2 подтверждена. Зависимость времени отклика от целевого размера файла (таблица 8) имеет выраженный внутренний минимум при 16–32 МБ. Слева от минимума растут поштучная составляющая и стоимость планирования; справа — объём сканирования, поскольку укрупнение файлов огрубляет блочную статистику (доля сканирования монотонно растёт с 9,62 % до 17,35 %). Отклонение от оптимума в 4 раза вниз обходится в 8,9 % времени отклика, в 16 раз вверх — в 44,2 %; принятое по умолчанию во многих системах значение 128 МБ даёт здесь проигрыш 14,6 %. Это оправдывает включение целевого размера файла в пространство оптимизации, а не задание его константой.

Гипотеза H3 подтверждена, причём роль фильтра (12) существенна. Поправка без фильтра снижает среднюю относительную ошибку с 70,4 % до 25,3 %, поправка с фильтром — до 19,7 % при снижении максимальной ошибки с 0,127 до 0,051. Механизм фильтра виден на запросе Q5 (тип услуги и два региона). Без фильтра формула берёт минимум индуцированной селективности по всем ключам упорядочивания, включая идентификатор абонента; поскольку кардинальность абонента ($1{,}17 \cdot 10^6$) сопоставима с числом строк, множество абонентов, встретившихся с данным типом услуги, оказывается «малым» чисто статистически, и модель предсказывает несуществующее отсечение — 0,0198 против измеренных 0,0872. Фильтр пропускает только пару «сота — регион» с мерой зависимости, равной единице (таблица 9), и оценка поднимается до 0,0559. Иными словами, индуцированная селективность без проверки на функциональную зависимость измеряет случайное совпадение множеств значений, а не реальную связь атрибутов.

Остаточная ошибка сосредоточена на запросах Q3 и Q8 — точечном и списочном предикатах по идентификатору абонента, где модель занижает сканирование примерно вдвое. Причина в допущении (в): формула (10) выведена для связного диапазонного предиката, тогда как множество идентификаторов когорты рассеяно по пространству ключа и пересекает существенно больше файлов, чем компактный гиперпрямоугольник той же меры. Уточнение оценки для этого класса предикатов — направление дальнейшей работы.

Практическая значимость и рекомендации по внедрению. Метод не требует изменения формата хранения или вычислительного движка и реализуется поверх существующих табличных форматов, поскольку опирается только на стандартные механизмы секционирования, сортировки при записи и блочной статистики. Предлагается следующий порядок внедрения: собрать журнал запросов за 2–4 недели из истории Spark или Trino и извлечь предикаты с частотами; оценить селективности, индуцированные селективности и меры зависимости по выборке 1–5 % таблицы с ежемесячным пересчётом профиля; выполнить поиск раскладки в офлайне, поскольку модель вычисляется аналитически, без пробных перестроений; применять выбранную раскладку к вновь записываемым секциям, а исторические секции перестраивать по правилу (17), начиная с наиболее часто читаемых;

константы модели калибровать однократно на кластере по серии контрольных запросов и проверять устойчивость выбранной раскладки к их вариации на ±30 %.

Ожидаемый экономический эффект складывается из сокращения потребления вычислительных ресурсов пропорционально снижению объёма сканирования и из сокращения времени отклика витрин, что напрямую влияет на длительность цикла подготовки маркетинговых кампаний.

Ограничения. Результаты получены на программной модели механики отсечения, а не на промышленном кластере; абсолютные значения времени отклика зависят от констант модели и должны трактоваться как модельные. Оценки выведены при допущениях (а)–(в) и теряют точность для списочных предикатов. Формулы применены с усреднённым числом файлов на секцию, что правомерно при умеренном перекосе; при сильном перекосе требуется посекционная свёртка согласно замечанию к (8). Модель не охватывает соединения таблиц, где на раскладку влияет также совместное размещение по ключу соединения. Распределение нагрузки предполагается стационарным на горизонте между перестроениями; при резкой смене профиля запросов правило компактизации реагирует с запаздыванием порядка точки безубыточности.

5. ЗАКЛЮЧЕНИЕ

1. Задача выбора физической раскладки данных в озере данных формализована как задача минимизации математического ожидания стоимости обслуживания рабочей нагрузки по трём параметрам: ключам секционирования, ключам упорядочивания и целевому размеру файла. Показано, что раздельная оптимизация этих параметров, принятая на практике, некорректна, поскольку они связаны через число файлов в секции (8) и через постоянную составляющую стоимости планирования (6).

2. Уточнена модель секционного отсечения: показано, что отсечение имеет гранулярность секции, получено неравенство (7) с условием достижения равенства и сформулировано условие сохранения целевого размера файла (9).

3. Для многомерного упорядочивания по кривой Мортона выведена аналитическая оценка доли просматриваемых файлов (10) со слагаемым граничного эффекта при явно очерченных допущениях. Оценка объясняет наблюдаемую на практике деградацию отсечения при включении в порядок более трёх-четырёх столбцов и даёт количественный критерий выбора размерности.

4. Установлено, что базовая оценка систематически завышает объём сканирования для столбцов, функционально связанных с ключами упорядочивания. Предложена поправка через индуцированную селективность (11) с фильтрацией по мере силы функциональной зависимости (12); показано, что без фильтрации поправка срабатывает на случайных совпадениях множеств значений. Поправка снижает среднюю относительную ошибку модели с 70,4 % до 19,7 %.

5. Предложен алгоритм ADPC с оценкой полезности (14), учитывающей граничный эффект через обрезку селективности снизу. Показано, что жадный выбор по этой оценке субоптимален и уступает выбору по полной модели стоимости в 1,52 раза по времени отклика, то есть ранжирование пригодно лишь для сокращения пространства поиска.

6. Сформулировано правило запуска компактизации по накопленному регрету (15)–(16) с явной точкой безубыточности (17) в согласованных единицах кластеро-секунд, заменяющее пороговые эвристики экономическим критерием.

7. Вычислительный эксперимент подтвердил: выбранная алгоритмом раскладка сокращает средневзвешенный объём сканирования в 2,73 раза и модельное время отклика в 2,55 раза; зависимость времени отклика от целевого размера файла немонотонна с минимумом при 16–32 МБ; избыточное секционирование ухудшает время отклика на 7,8 % несмотря на уменьшение объёма сканирования; при этом запросы без селективных предикатов замедляются на 1–9 %, что определяет границы применимости метода.

Направления дальнейшей работы: уточнение оценки для списочных предикатов и для конъюнкций слабо связанных условий; посекционная формулировка при сильном перекосе; расширение постановки на запросы с соединениями; верификация на промышленном кластере.

ЛИТЕРАТУРА

1. Moerkotte G. Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing // Proc. VLDB. — 1998. — P. 476–487.
2. Stonebraker M., Abadi D.J., Batkin A. et al. C-Store: A Column-oriented DBMS // Proc. VLDB. — 2005. — P. 553–564.
3. Abadi D., Boncz P., Harizopoulos S. et al. The Design and Implementation of Modern Column-Oriented Database Systems // Foundations and Trends in Databases. — 2013. — Vol. 5, No. 3. — P. 197–280.
4. Athanassoulis M., Kester M.S., Maas L.M. et al. Designing Access Methods: The RUM Conjecture // Proc. EDBT. — 2016. — P. 461–466.
5. Dageville B., Cruanes T., Zukowski M. et al. The Snowflake Elastic Data Warehouse // Proc. ACM SIGMOD. — 2016. — P. 215–226.
6. Melnik S., Gubarev A., Long J.J. et al. Dremel: Interactive Analysis of Web-Scale Datasets // PVLDB. — 2010. — Vol. 3, No. 1. — P. 330–339.
7. Armbrust M., Das T., Sun L. et al. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores // PVLDB. — 2020. — Vol. 13, No. 12. — P. 3411–3424.
8. Morton G.M. A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. — Ottawa: IBM Ltd., 1966. — 20 p.
9. Lawder J.K., King P.J.H. Querying Multi-dimensional Data Indexed Using the Hilbert Space-Filling Curve // ACM SIGMOD Record. — 2001. — Vol. 30, No. 1. — P. 19–24.
10. Chaudhuri S., Narasayya V. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server // Proc. VLDB. — 1997. — P. 146–155.
11. Agrawal S., Chaudhuri S., Narasayya V. Automated Selection of Materialized Views and Indexes for SQL Databases // Proc. VLDB. — 2000. — P. 496–505.
12. Sun L., Franklin M.J., Krishnan S., Xin R.S. Fine-grained Partitioning for Aggressive Data Skipping // Proc. ACM SIGMOD. — 2014. — P. 1115–1126.
13. Nathan V., Ding J., Alizadeh M., Kraska T. Learning Multi-dimensional Indexes // Proc. ACM SIGMOD. — 2020. — P. 985–1000.

14. Ding J., Nathan V., Alizadeh M., Kraska T. Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads // PVLDB. — 2020. — Vol. 14, No. 2. — P. 74–86.
15. Kraska T., Beutel A., Chi E.H. et al. The Case for Learned Index Structures // Proc. ACM SIGMOD. — 2018. — P. 489–504.
16. Idreos S., Kersten M.L., Manegold S. Database Cracking // Proc. CIDR. — 2007. — P. 68–78.
17. Thusoo A., Sarma J.S., Jain N. et al. Hive – A Petabyte Scale Data Warehouse Using Hadoop // Proc. IEEE ICDE. — 2010. — P. 996–1005.
18. Vavilapalli V.K., Murthy A.C., Douglas C. et al. Apache Hadoop YARN: Yet Another Resource Negotiator // Proc. ACM SoCC. — 2013. — Art. 5.
19. Kreps J., Narkhede N., Rao J. Kafka: A Distributed Messaging System for Log Processing // Proc. NetDB. — 2011. — P. 1–7.
20. O'Neil P., Cheng E., Gawlick D., O'Neil E. The Log-Structured Merge-Tree (LSM-Tree) // Acta Informatica. — 1996. — Vol. 33, No. 4. — P. 351–385.
21. Zaharia M., Chowdhury M., Das T. et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing // Proc. USENIX NSDI. — 2012. — P. 15–28.
22. Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters // Communications of the ACM. — 2008. — Vol. 51, No. 1. — P. 107–113.
23. Boncz P.A., Zukowski M., Nes N. MonetDB/X100: Hyper-Pipelining Query Execution // Proc. CIDR. — 2005. — P. 225–237.