УДК 004.65:004.75

Аббасов Э.М.,
аспирант
2 курс, факультет автоматики и вычислительной техники
Новосибирский Государственный Технический Университет
Россия, г. Новосибирск

АДАПТИВНОЕ РАЗМЕЩЕНИЕ АНАЛИТИЧЕСКИХ ДАННЫХ В LAKEHOUSE-ИНФРАСТРУКТУРЕ

Аннотация: *В статье рассматривается методика адаптивного размещения аналитических данных в lakehouse-инфраструктуре на базе ClickHouse, MinIO/S3 и Parquet. Предложен агент-оптимизатор, анализирующий SQL-нагрузку и автоматически создающий локальные агрегированные представления. Эксперимент на 100 млн строк показал ускорение hot-запросов до 7,56 раза по сравнению с прямым чтением из S3/Parquet.*

Ключевые слова: *большие данные; lakehouse; ClickHouse; MinIO; Parquet; материализованные представления; SQL-нагрузка.*

Annotation: *The article considers adaptive analytical data placement in a lakehouse infrastructure based on ClickHouse, MinIO/S3 and Parquet. A dynamic optimization agent is proposed; it analyzes SQL workload and automatically creates local aggregate views. An experiment on 100 million rows showed up to 7.56 times acceleration of hot queries compared with direct S3/Parquet reading.*

Key words: *big data; lakehouse; ClickHouse; MinIO; Parquet; materialized views; SQL workload.*

1. Введение

Рост объема аналитических данных усиливает противоречие между экономичным хранением детальных событий и необходимостью интерактивного выполнения BI-запросов. Объектные хранилища и колоночные форматы, такие как Parquet, позволяют дешево и воспроизводимо хранить большие наборы данных, однако прямое чтение из cold-слоя не всегда обеспечивает приемлемую задержку при повторяющихся агрегирующих запросах. Перенос всех данных в локальный вычислительный слой также не является универсальным решением, поскольку увеличивает стоимость хранения и усложняет эксплуатацию.

Lakehouse-подход объединяет свойства data lake и data warehouse: детальные данные размещаются в объектном хранилище, а вычислительный контур обеспечивает SQL-доступ и аналитическую обработку [1-4]. Эффективность такой архитектуры определяется не только форматом хранения, но и тем, какие агрегаты должны быть перенесены ближе к вычислениям. В статически спроектированных витринах этот выбор выполняется заранее, тогда как реальная нагрузка BI-систем может изменяться во времени.

Цель настоящей статьи - представить и экспериментально оценить методику адаптивного размещения аналитических данных, при которой решение о создании локальной агрегированной структуры принимается на основе фактической SQL-нагрузки. Объектом исследования является гибридная аналитическая инфраструктура ClickHouse + MinIO/S3 + Parquet + Superset. Предметом исследования выступает механизм динамического физического проектирования, основанный на анализе журнала запросов ClickHouse.

2. Материалы и методы

Экспериментальный стенд построен как воспроизводимая lakehouse-инфраструктура. В cold-слое используется S3-совместимое объектное хранилище MinIO, где размещается секционированный Parquet-набор

транзакционных данных. В качестве вычислительного и локального hot-слоя применяется ClickHouse, а для имитации BI-сценария - Apache Superset [5-8]. Программная часть включает генератор данных, скрипты загрузки и динамический агент-оптимизатор.

Таблица 1.

Компоненты экспериментального стенда

Компонент	Назначение	Используемые объекты
Python-генератор	Формирование синтетических транзакций и Parquet dataset	data_generator.py, CSV, Parquet partitions
MinIO	S3-совместимое объектное хранилище cold-слоя	bucket lake, transactions_parquet
ClickHouse	Колоночная обработка, локальные таблицы и системный журнал	transactions_s3, transactions_raw, transactions_aggregated, system.query_log
Superset	BI-слой и проверка доступности аналитического интерфейса	dataset на view ClickHouse
Динамический агент	Выявление slow patterns и синтез physical design объектов	autoagg_, mv_autoagg_, v_autoagg_*, registry table

Выбор ClickHouse как вычислительного слоя связан с особенностями колоночных СУБД, ориентированных на аналитические запросы, агрегацию и сокращение объема считываемых данных [9, 10]. Такая архитектура позволяет сопоставить прямое чтение из объектного слоя, чтение локальных агрегатов и гибридные маршруты доступа к данным.

Синтетический набор имитирует транзакционный поток. Каждая запись содержит event_id, event_time, event_date, client_id, amount, channel и region. Для моделирования неравномерности пользовательской активности применяется распределение Ципфа, а для временной сезонности - смесь суточных пиков. Вероятность выбора клиента ранга k задается выражением (1):

$$P(k) = \frac{k^{-\theta}}{\sum_{i=1}^{M} i^{-\theta}} \quad (1)$$

где M - количество уникальных клиентов, θ - параметр концентрации активности. Такой способ генерации позволяет получить приближение к типовой ситуации, когда небольшая доля клиентов формирует значительную часть транзакций.

Динамический агент-оптимизатор реализован как внешний Python-процесс. Он подключается к ClickHouse через native-протокол или резервный механизм docker exec, инициирует сброс журналов, читает system.query_log и выделяет повторяющиеся агрегирующие SELECT-запросы к таблице transactions_s3. Для каждого найденного шаблона используется normalized_query_hash, а также фактические показатели частоты появления и длительности выполнения. Подход относится к классу автоматизированной настройки физического проектирования баз данных, так как структура создается не по экспертному предположению, а на основе измеряемых параметров нагрузки [11, 12].

Множество запросов-кандидатов на материализацию определяется правилом (2):

$$Q_\tau = \{\, q \mid \text{freq}(q,\tau) \geq f_{\min} \ \wedge\ \text{duration}(q,\tau) \geq t_{\min} \,\} \qquad (2)$$

где τ - окно наблюдения, $\text{freq}(q,\tau)$ - частота появления шаблона, duration(q, τ) - длительность выполнения, $f_{\min}$ и $t_{\min}$ - заданные пороги. После отбора агент извлекает поля GROUP BY и агрегатные функции count, sum, avg, min, max, uniqExact. Для обеспечения идемпотентности вычисляется короткая сигнатура (3):

$$sig = \text{hash}(\text{dimensions} \parallel \text{aggregates} \parallel \text{raw_table}) \qquad (3)$$

на основе которой создаются имена целевой таблицы AggregatingMergeTree, materialized view и итогового представления. Созданные объекты фиксируются в agent_optimization_registry. Это обеспечивает воспроизводимость, аудит примененных оптимизаций и возможность связать результаты измерений с конкретной сигнатурой SQL-паттерна.

Benchmark-эксперименты выполнялись в трех группах сценариев. Hot-сценарии проверяют последние 30 дней и демонстрируют эффект локальных агрегатов. Cold-сценарии обращаются к исторической части Parquet-набора, которая остается в S3/Parquet. Mixed-сценарий пересекает границу hot/cold и позволяет оценить работу гибридного представления. Подобная схема измерений соответствует общей практике экспериментальной оценки аналитических платформ больших данных [13-15].

Таблица 2.

Benchmark-сценарии эксперимента

Сценарий	Источник	Назначение
s3_hot_daily	transactions_s3	Базовое прямое чтение hot-диапазона из S3/Parquet
local_hot_mv	transactions_aggregated	Оценка предельного эффекта локального агрегата
hybrid_hot_daily	v_transactions_hybrid_daily	Проверка hot-маршрута гибридной витрины
s3_cold_daily	transactions_s3	Базовое прямое чтение cold-диапазона
hybrid_cold_daily	v_transactions_hybrid_daily	Проверка cold-маршрута гибридной витрины
hybrid_mixed_daily	v_transactions_hybrid_daily	Проверка запроса, пересекающего hot и cold диапазоны
generated_autoagg_hot	v_autoagg_	Оценка динамически сформированного итогового представления

Для интегральной количественной оценки маршрутов оптимизации может использоваться взвешенная метрика системной эффективности (4):

$$E_j = \alpha \frac{T_0}{T_j} + \beta \frac{B_0}{B_j} + \gamma \frac{M_0}{M_j} - \delta \frac{S_j}{S_0} \quad (4)$$

где E_j - интегральная эффективность j-го сценария; T, B, M и S - соответственно время выполнения, объем чтения, потребление памяти и объем хранения; индекс 0 обозначает базовый сценарий прямого чтения S3/Parquet; α, β, γ и δ - весовые коэффициенты.

3. Результаты исследования и обсуждение

Эксперимент на 100 млн строк проводился с одним warm-up запуском и 30 измеряемыми повторами на каждый сценарий. Порядок сценариев перемешивался, что снижало влияние последовательности выполнения. Для оценки использовались две группы метрик: client-side показатели, отражающие пользовательское время ожидания результата, и server-side показатели из system.query_log, отражающие длительность выполнения запроса, объем чтения и использование памяти на стороне ClickHouse.

Таблица 3.

Client-side summary расширенного benchmark на 100 млн строк

Сценарий	Тип маршрута	Повторов	Median, мс	P95, мс
S3 hot	S3/Parquet, hot	30	3871,44	5464,14
Local MV	Локальный агрегат, hot	30	512,12	1196,26
Hybrid hot	Гибридная витрина, hot	30	2223,86	3705,75
S3 cold	S3/Parquet, cold	30	11294,84	16860,94
Hybrid cold	Гибридная витрина, cold	30	10547,29	14779,44
Hybrid mixed	Гибридная витрина, mixed	30	5286,87	7029,34
Autoagg hot	Представление агента	30	560,06	821,89

По данным табл. 3 медиана прямого чтения hot-диапазона из S3/Parquet составила 3871,44 мс, тогда как локальный агрегированный слой показал медиану 512,12 мс. Локальный hot-слой оказался быстрее прямого чтения примерно в 7,56 раза. Гибридная hot-витрина также сократила задержку по сравнению с S3/Parquet, но ее результат оказался менее выраженным, чем у полностью локального агрегата.

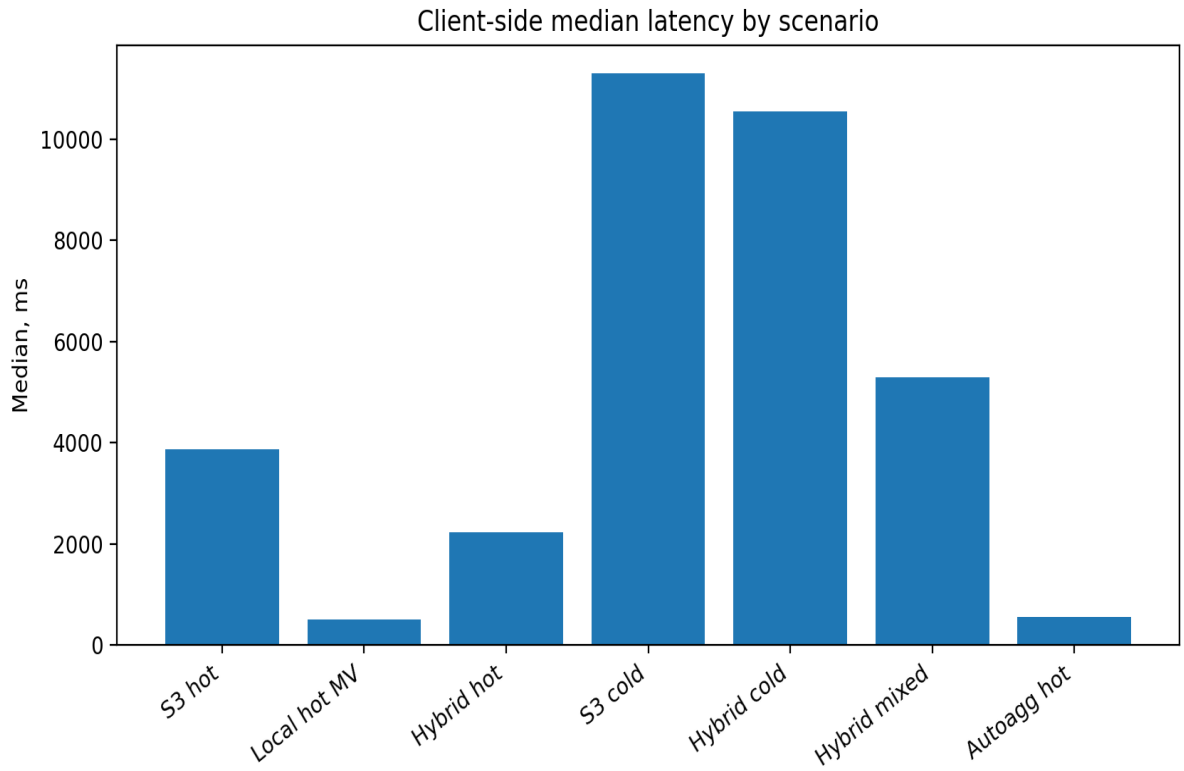


Рисунок 1. Сравнение медианной client-side задержки по сценариям

Таблица 4.

Server-side summary по system.query_log на 100 млн строк

Сценарий	Запросов	Median duration, мс	Median read, МиБ	Median memory, МиБ
S3 hot	233	28	35,86	42,44
Local MV	234	7	2,20	7,68
Hybrid hot	234	23	2,20	36,80
S3 cold	234	50	92,96	67,37
Hybrid cold	233	57	93,00	66,85
Hybrid mixed	233	42	54,19	60,17
Autoagg hot	124	26	8,91	14,20

Server-side метрики отделяют работу ClickHouse от накладных расходов Python-клиента, Docker и сетевого взаимодействия. Для локального hot-агрегата медианный объем чтения составил 2,20 МиБ против 35,86 МиБ у прямого hot-чтения из S3/Parquet. Это показывает, что снижение задержки

связано не только с локальностью размещения данных, но и с уменьшением фактического объема чтения.

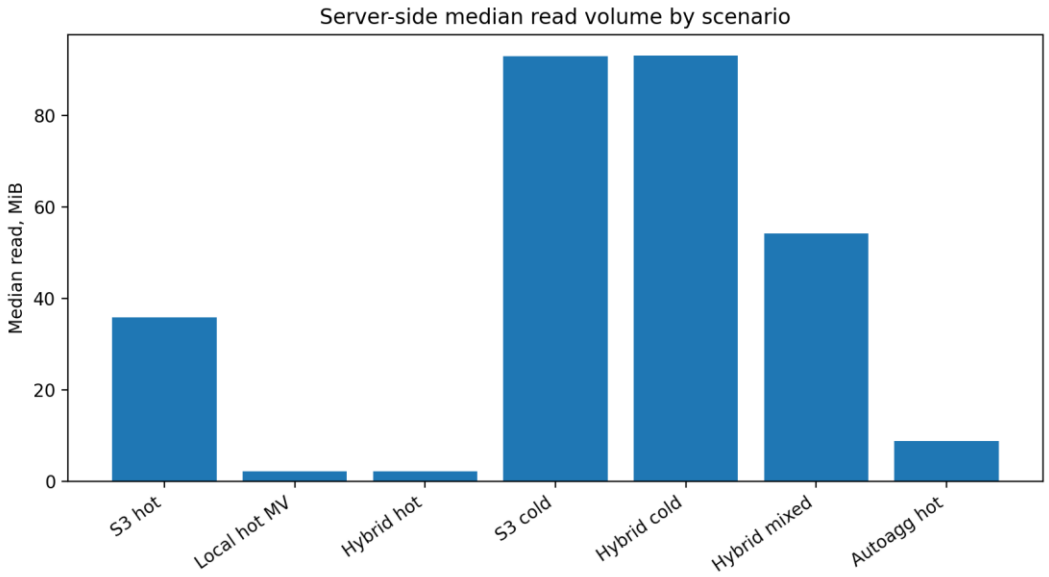


Рисунок 2. Медианный объем чтения по server-side метрикам

Отдельно была проверена динамически созданная витрина v_autoagg_87ec8a1fb3d2. В изолированном benchmark ее медианное время выполнения составило 206,49 мс, а P95 - 368,81 мс. Этот результат показывает, что автоматически сформированное итоговое представление способно работать быстрее заранее заданной гибридной витрины при условии, что оно соответствует конкретному устойчивому SQL-паттерну.

Таблица 5.

Проверка эквивалентности агрегатов на 100 млн строк

Группа	Сценарий	Fingerprint: groups / sum_amount / event_count	Статус
hot	S3 hot	144988 / 2407745766 / 16046243	ok
hot	Local MV	144988 / 2407745766 / 16046243	ok
hot	Hybrid hot	144988 / 2407745766 / 16046243	ok
hot	Autoagg hot	144988 / 2407745766 / 16046243	ok
cold	S3 cold	454947 / 7550326709 / 50331676	ok
cold	Hybrid cold	454947 / 7550326709 / 50331676	ok

Совпадение fingerprint для hot- и cold-групп подтверждает, что ускорение не достигалось за счет изменения семантики запроса. Для

сравниваемых маршрутов совпали количество групп, сумма amount и суммарное число событий, что позволяет считать результаты агрегирования эквивалентными в пределах выбранных сценариев.

Полученные результаты подтверждают, что для BI-нагрузки с повторяющимися агрегирующими запросами критически важен не только выбор колоночного формата, но и адаптивное размещение часто используемых агрегатов. Прямое чтение из S3/Parquet остается удобным для cold-данных и воспроизводимого хранения исходного набора, однако при регулярных hot-запросах локальная материализация существенно сокращает пользовательскую задержку.

Научная новизна работы заключается в реализации полного цикла адаптации физического проектирования: наблюдение за фактической нагрузкой, выделение устойчивого SQL-паттерна, синтез физической структуры, backfill, регистрация результата и экспериментальная проверка. В отличие от заранее подготовленных статических витрин, агент ориентируется на реальные запросы, что повышает применимость подхода в условиях изменяющегося профиля BI-аналитики.

Практическая значимость состоит в том, что предложенный агент может быть использован как прототип инструмента сопровождения аналитических витрин в открытом lakehouse-стеке. Исходные детальные данные остаются в объектном хранилище, а локальный слой ClickHouse используется только для тех агрегатов, которые подтверждены фактической нагрузкой. Это снижает риск избыточной материализации и делает решение проверяемым.

4. Ограничения исследования

Текущая реализация агента ориентирована на однотабличные агрегирующие SELECT-запросы к transactions_s3 с GROUP BY и типовыми агрегатными функциями count, sum, avg, min, max и uniqExact. JOIN, UNION, подзапросы, оконные функции и произвольное переписывание SQL-запросов в автоматическом режиме не рассматривались.

Агент создает итоговое представление, но не перенаправляет существующие BI-запросы автоматически. После создания view пользователь, SQL-слой или BI-dataset должны быть явно направлены на сформированный объект. Абсолютные значения времени выполнения зависят от аппаратной конфигурации, настроек Docker, состояния файлового кэша, версии ClickHouse и MinIO, а также от сетевых условий обращения к S3-совместимому хранилищу. Поэтому ключевое значение имеют не только абсолютные миллисекунды, но и относительные коэффициенты ускорения.

5. Заключение

В работе разработана и экспериментально проверена методика адаптивного размещения аналитических данных в lakehouse-инфраструктуре. Реализованный стенд ClickHouse + MinIO/S3 + Parquet + Superset позволил сопоставить прямое чтение из cold-слоя, локальный агрегированный слой, гибридную витрину и итоговое представление, сформированное динамическим агентом-оптимизатором.

Эксперимент на 100 млн строк показал, что для hot-диапазона локальный агрегированный слой снижает медианную client-side задержку примерно в 7,56 раза по сравнению с прямым S3/Parquet-чтением. Автоматически созданное итоговое представление в изолированном тесте достигло медианы 206,49 мс при P95 368,81 мс. Проверка fingerprint подтвердила эквивалентность агрегатов для сравниваемых маршрутов.

Полученные результаты позволяют рассматривать адаптивную материализацию как перспективное направление оптимизации BI-витрин в открытых lakehouse-системах. Дальнейшее развитие подхода целесообразно связать с поддержкой многотабличных запросов, автоматическим переписыванием SQL, политиками удаления устаревших агрегатов и оценкой экономической эффективности хранения.

Использованные источники:

1. Nambiar, A. An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management / A. Nambiar, D. Mundra // Big Data and Cognitive Computing. - 2022. - Vol. 6, No. 4. - P. 132.
2. Hai, R. Data Lake concept and systems: A survey / R. Hai, C. Quix, M. Jarke // arXiv preprint arXiv:2106.09592. - 2021.
3. Melnik, S. Dremel: Interactive Analysis of Web-Scale Datasets / S. Melnik [et al.] // Proceedings of the VLDB Endowment. - 2010. - Vol. 3, No. 1-2. - P. 330-339.
4. Armbrust, M. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics / M. Armbrust [et al.] // CIDR. - 2021.
5. Apache Parquet Documentation [Электронный ресурс]. URL: https://parquet.apache.org/docs/.
6. MinIO Documentation [Электронный ресурс]. URL: https://min.io/docs/minio/container/index.htmlClickHouse Documentation [Электронный ресурс]. URL: https://clickhouse.com/docs/.
7. Apache Superset Documentation [Электронный ресурс]. URL: https://superset.apache.org/docs/intro.
8. Abadi, D. J. Column-oriented database systems / D. J. Abadi, P. A. Boncz, S. Harizopoulos // Proceedings of the VLDB Endowment. - 2009. - Vol. 2, No. 2. - P. 1664-1665.
9. Stonebraker, M. C-Store: A column-oriented DBMS / M. Stonebraker [et al.] // Proceedings of the 31st International Conference on Very Large Data Bases. - 2005. - P. 553-564.
10. Bruno, N. Automatic physical database tuning: a relaxation-based approach / N. Bruno, S. Chaudhuri // Proceedings of the ACM SIGMOD International Conference on Management of Data. - 2005. - P. 227-238.

11. Agrawal, S. Database tuning advisor for Microsoft SQL Server 2005 / S. Agrawal [et al.] // Proceedings of the VLDB Endowment. - 2004. - P. 1110-1121.

12. Zaharia, M. Apache Spark: a unified engine for big data processing / M. Zaharia [et al.] // Communications of the ACM. - 2016. - Vol. 59, No. 11. - P. 56-65.

13. Carbone, P. Apache Flink: Stream and batch processing in a single engine / P. Carbone [et al.] // Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. - 2015. - Vol. 38, No. 4.

14. Singh, D. A survey on platforms for big data analytics / D. Singh, C. K. Reddy // Journal of Big Data. - 2015. - Vol. 2. - P. 1-20.