ЕРВУ — поиск и аналитика

Текущие проблемы

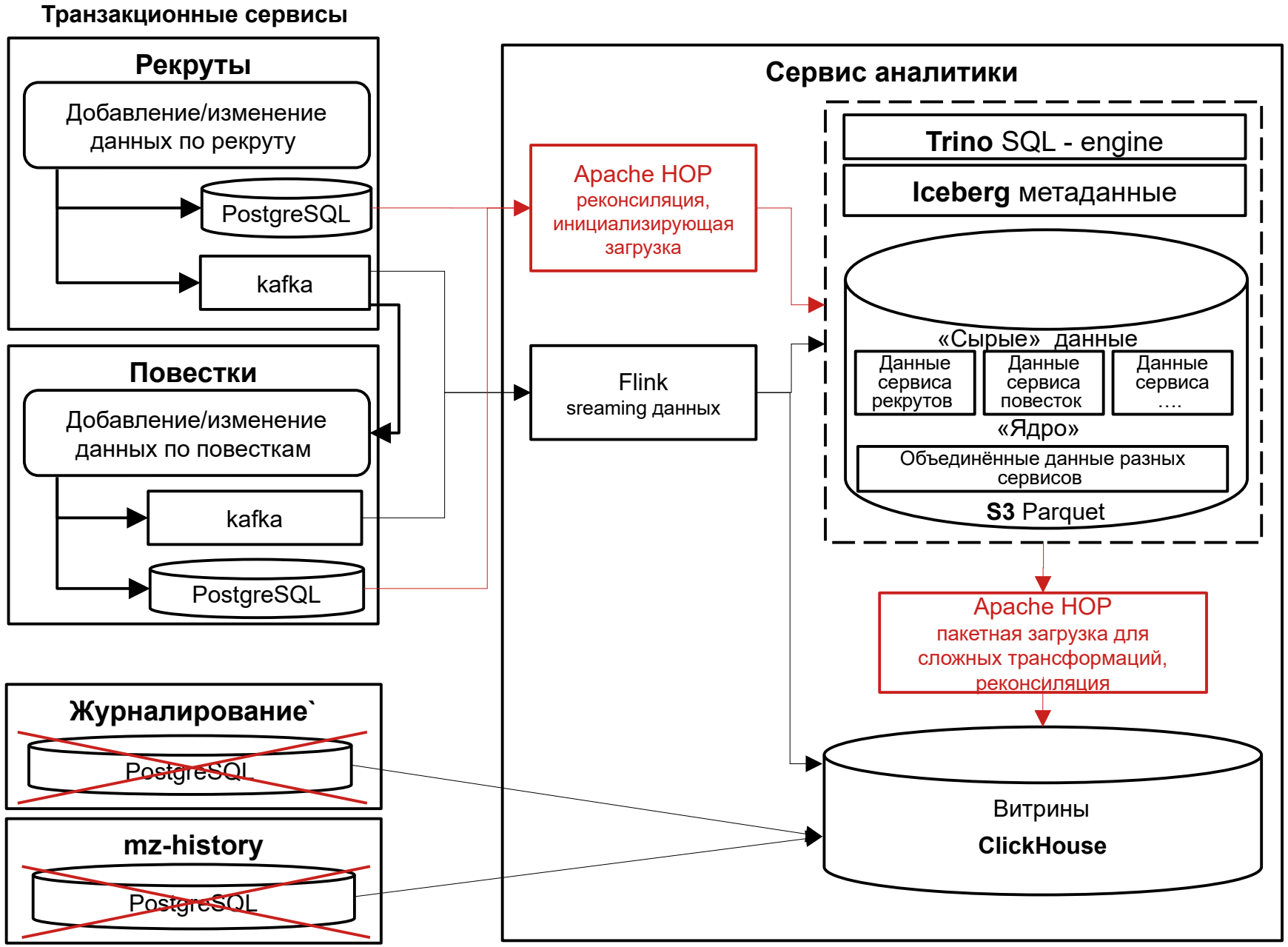
Медленная работа поисковых запросов на БД транзакционных сервисов
Поисковые запросы на БД транзакционных сервисов работают медленно, что связано с конкуренцией за ресурсы с запросами, выполняемыми в ходе решения прочих задач этих сервисов.

Задержки при загрузке данных в аналитическую подсистему
Данные в аналитической БД появляются с запаздыванием. Хотелось бы сократить лаг между появлением информации в БД транзакционных сервисов и в аналитической БД

Требуется решение по «глобальному поиску» для обеспечения возможности выполнения произвольных запросов к сводным данным разных сервисов
При традиционном подходе с использованием PostgreSQL в качестве аналитической БД такие запросы выполняются медленно при отсутствии индексов. Большое кол-во индексов замедляет вставку данных.

Быстрый рост объемов журналов в скором времени не позволит их эффективно обрабатывать
Данные по журналированию растут очень быстро, особенно в части журнала пользовательских операций, журнала истории объектов и журналов межведомственного обмена. Удалять данные из этих журналов можно только после достаточно длительного интервала хранения(~ 5 лет и более). Быстрый рост объемов данных в этих журналах создаёт риск того, что они не смогут эффективно (или хоть как-то) обрабатываться средствами PostgreSQL. Для работы с журналами требуются иные технологии.

Общая схема решения



Общее описание компонентов решения

ClickHouse — аналитическая колоночная распределенная БД, обеспечивающая хранение и очень быстрый поиск для данных больших объёмов. Поиск эффективен даже без индексов, поскольку для фильтрации колоночная БД читает и сканирует только те колонки, на которые наложены критерии фильтрации.
Недостатком БД являются ограничения по работе с join'ами. Данные, в общем случае должны быть денормализованными и размещаться в одной таблице. Для имитации join'ов есть определённые подходы типа nested атрибутов (поля связанной таблицы вкладываются в структуру основной таблицы в виде массива)

S3+Parquet+Iceberg+Trino — аналитическая колоночная, распределенная БД. Работающая несколько более медленно чем ClickHouse, но позволяющая относительно быстро делать join'ы.
Используется для хранения:

- сводных данных разных сервисов (ядра), обеспечивающих глобальный поиск.
- сырых данных из источников

Flink — инструмент для потоковой обработки данных (streaming). Имеет встроенные адаптеры для подключения к различным источникам и приёмникам. Позволяет конструировать цепочки по обработке данных, упрощая разработку
Реализует отказоустойчивое кластерное решение

HOP — инструмент для пакетной загрузки и обработки данных. Используется для инициализирующей загрузки, реконсиляции данных источников и приёмников, а так же для сложных преобразований, которые требуют обработки больших объёмов данных и не могут быть эффективно выполнены при потоковой обработке.

Предложения по решению

Проблема	Способ решения
Медленная работа поисковых запросов на БД транзакционных сервисов	Перевод поисковых запросов на аналитическую колоночную массивно-параллельную (mpp) БД. • Данные транзакционных сервисов переносятся в специализированные витрины аналитической БД • В ходе переноса выполняется денормализация данных и парсинг JSON полей • Денормализация данных, преобразование JSON объектов в «регулярные» реляционные структуры, колоночное хранение, параллельная обработка и пр. особенности аналит. БД увеличивают скорость обработки запросов
Задержки при загрузке данных в аналитическую подсистему	Загрузка данных в аналит. подсистему переводится на поточный режим (по-возможности) • Данные не «вытягиваются» из источников через запросы по расписанию, а загружаются в аналит. подсистему по событиям, генерируемым транзакционными сервисами при появлении или изменении данных, что минимизирует лаг между появлением данных в источнике и поступлением их в аналит. сервис. • Для тех данных, которые будут загружаться в пакетном режиме (не все данные могут быть загружены в поточном режиме), предусматривается приоритизация данных с точки зрения их актуальности и приоритизацию процессов, отвечающих за пакетную загрузку. Данные, в высокими требованиями к их актуальности загружаются часто и в первую очередь, прочие данные загружаются реже.
Требуется решение по глобальному поиску для запросов к сводным данным разных сервисов	В аналитической подсистеме реализуется слой сводных данных разных сервисов («ядро»), к которому и адресуются запросы глобального поиска. • Реализация данного слоя предполагает выполнение соответствующих оптимизаций для повышения скорости запросов - «разумную» денормализацию, парсинг JSON’ов и пр.
Быстрый рост объемов журналов не позволит их эффективно обрабатывать	Перевод сервисов журналирования на непосредственную работу с аналитической БД (вместо PostgreSQL). Для сервисов журналирования работа с аналитической БД вполне возможна — эти сервисы записывают данные в аналитическую БД без их дальнейшего изменения, что вполне соответствуют профилю работы с аналитической БД (запись данных без дальнейшего редактирования с последующим поиском. Горизонтальное масштабирование и массивно-параллельная обработка данных позволит сохранять и обрабатывать большие объемы данных, наращивая ресурсы по мере необходимости.

Альтернативные варианты для аналитической БД

На предыдущих слайдах описано решение, использующее две аналитические БД, каждая из которых имеет свои достоинства и недостатки.

- *ClickHouse* — очень быстрая обработка данных в одной таблице, но ограниченная работа с join'ами, что затрудняет реализацию процессов, связанных с трансформацией данных и требует дополнительных внешних решений. В итоге имеем набор специализированных витрин, ориентированных на конкретные сценарии использования, для подготовки которых требуются внешние инструменты.
- S3+Parquet+Iceberg+Trino — приемлемые возможности по реализацию join'ов, но более медленная скорость исполнения запросов по отношению к ClickHouse. Это приводит к тому, что в ряде случаев приходится делать специализированные денормализованные витрины и перегружать их в ClickHouse

Комбинация этих БД повышает эффективность итогового решения, но ведёт к его существенному усложнению. При этом частая вставка или обновление данных создают определённые проблемы и требуют доп. затрат на их решение

В настоящее время стали появляться аналитические БД, которые обеспечивают:

- скорость выборки данных, сравнимую с ClickHouse
- эффективно работают join'ами (данные могут объединяться в запросе — не требуется заранее создавать плоские таблицы)
- обеспечивают эффективную модификацию данных на уровне отдельных полей

К таким БД можно отнести **StarRocks** (https://www.starrocks.io) и **Doris** (https://doris.apache.org)

Особенности работы аналитических БД

Аналитические колоночные БД, имеют особенности по работе с данными

S3+Iceberg+Trino

- Создаёт новый Parquet файл после выполнения операции Insert. Большое кол-во отдельных insert’ов приводит к большому кол-ву используемых Parquet файлов, что приводит к замедлению операций select. Исходя из этого, даже при поточной обработке данных вставку следует производить в режиме «микробатчей», накапливая поступающую информацию в течении хотя бы нескольких секунд, а не для каждой записи
- Обновление записей ведётся в режиме «Copy-On-Write» (COW) или «Merge-On-Read» (MOR).
  - Режим COW подразумевает перезапись всего Parquet файла, в одну или несколько записей которого внесены изменения. Медленно на запись, но быстро на чтение
  - Режим MOR подразумевает создание дополнительных Parquet файлов, в которых содержатся сведения об изменённых или удалённых записях. Слияние основного файла и файлов, содержащих сведения об изменениях происходит на этапе чтения данных. Быстро на вставку, но медленнее на чтение.
  -

Для ускорения работы системы и предотвращения появления слишком большого числа файлов должна периодически запускаться операция «Compaction». В ходе этой операции множество небольших файлов сливаются в один. Следует правильно подбирать размеры Parquet файлов, которые обеспечат максимальную скорость поисковых запросов. Частота операции «Compaction» так же должна быть оптимальной (слишком частое ее выполнение потребует слишком много ресурсов, а слишком редкое приведёт к наличию большого числа файлов, что не оптимально для работы системы)

ClickHouse

- Операции, связанные с модификацией или удалением данных, в общем случае не модифицируют текущую запись, а создают её новую версию. Это приводит к росту объёма занимаемых данных и к снижению скорости исполнения поисковых запросов. Для удаления старых версий записей и оптимизации структуры хранения в ClickHouse должна выполняться операция «Merge» (по назначению аналогичная операции «Compaction»)
- ClickHouse содержит множество процессоров (engine), посредством которых ведётся обработка данных. Важно выбрать engine, максимально соответствующий решаемой задаче

StarRocks

Особенности StarRocks

Open source решение

Есть Российское представительство - https://starrockspro.ru/

Быстрая аналитика на больших данных

Благодаря встроенной многомерной агрегации и векторизации запросов, анализ проходит очень быстро, обеспечивая результаты, сравнимые с ClickHouse.

*Гибкая интеграция и **потоковая** загрузка данных*

Благодаря **поддержке потоковой загрузки** данных и широкому набору коннекторов, StarRocks легко интегрируется с Kafka, Flink и другими источниками.

***Автоматическое** горизонтальное масштабирование*

Платформа масштабируется, добавляя узлы и перераспределяя данные без потери производительности.

Оптимизированное хранение и высокопроизводительные индексы

StarRocks использует колоночное хранение, адаптивное индексирование и сжатие данных, что снижает нагрузку на диски и ускоряет аналитические операции.

Низкая задержка и высокая доступность

Благодаря архитектуре MPP и эффективному кешированию, StarRocks обеспечивает обработку сложных аналитических запросов с минимальной задержкой

Основные шаги решения

Решение на базе S3+Iceberg+Trino, ClickHouse

1) Развернуть инфраструктуру для SQL доступа данных в S3 (S3 + Iceberg + Trino).
2) Развернуть ClickHouse для использования в целях аналитики и отчётности
3) Развернуть инфраструктуру Flink для обеспечения потоковой обработки данных
4) Обеспечить со стороны сервисов системы передачу в очереди kafka сведений о создании, изменении, удалении экземпляров информационных сущностей, используемых в отчётности и аналитике.
5) Разработать модель данных ClickHouse для обеспечения решения задач оперативной отчётности (в рамках отдельных сервисов).
6) Разработать ETL процессы для первичной загрузки данных в витрины ClickHouse для задач оперативной отчётности
7) Разработать процессы потоковой загрузки данных в витрины ClickHouse для задач оперативной отчётности с целью поддержания их в актуальном состоянии
8) Разработать процедуры реконсиляции для сверки состава данных в витринах оперативной отчётности и в источниках
9) Разработать модель данных сырого слоя на S3+Iceberg+Trino
10) Разработать ETL процедуры для первичной загрузки данных в сырой слой
11)Разработать процессы потоковой загрузки данных в сырой слой с целью поддержания его в актуальном состоянии
12)Разработать модель ядра (слоя консолидированных данных) для обеспечения глобального поиска
13)Разработать ETL процедуры для первичной загрузки ядра (слоя консолидированных данных)
14)Разработать процессы для поддержания ядра в актуальном состоянии (загрузка из слоя сырых данных в ядро)

Основные шаги решения

Решение на базе StarRocks

1) Развернуть StarRocks.
2) Развернуть инфраструктуру Flink для обеспечения потоковой обработки данных
3) Обеспечить со стороны сервисов системы передачу в очереди kafka сведений о создании, изменении, удалении экземпляров информационных сущностей, используемых в отчётности и аналитике.
4) Разработать модель данных StarRocks для обеспечения решения задач оперативной отчётности (в рамках отдельных сервисов).
5) Разработать ETL процессы для первичной загрузки данных в витрины StarRocks для задач оперативной отчётности
6) Разработать процессы потоковой загрузки данных в витрины StarRocks для задач оперативной отчётности с целью поддержания их в актуальном состоянии
7) Разработать процедуры реконсиляции для сверки состава данных в витринах оперативной отчётности и в источниках
8) Разработать модель данных сырого слоя на StarRiocks
9) Разработать ETL процедуры для первичной загрузки данных в сырой слой
10) Разработать процессы потоковой загрузки данных в сырой слой с целью поддержания его в актуальном состоянии
11) Разработать модель ядра (слоя консолидированных данных) для обеспечения глобального поиска
12) Разработать ETL процедуры для первичной загрузки ядра (слоя консолидированных данных)
13) Разработать процессы для поддержания ядра в актуальном состоянии (загрузка из слоя сырых данных в ядро)