ЭТАП 2. АРХИТЕКТУРА ПРОГРАММНОГО РЕШЕНИЯ

Заказчик:	МТС.Тета, Демиденко Савва Александрович
Название проекта:	Аналитическое хранилище для KION
Исполнители	Сафонов Николай Юрьевич, Чертанов Денис Алексеевич

ОПИСАНИЕ И НАЗНАЧЕНИЕ ПРОГРАММЫ

Цель сервиса - сохранять и отдавать под высокой нагрузкой персональную историю просмотров пользователя.

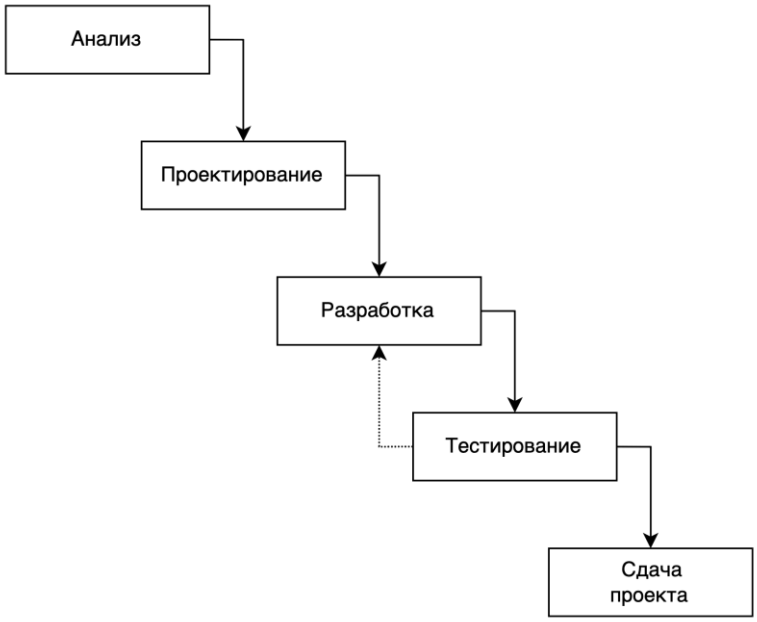
Ключевые требования:

- Сервис выдерживает нагрузку на запись новых событий - 8000 RPS;
- Сервис выдерживает нагрузку на чтение - 250 RPS;
- Доступность сервиса - 99,9%.

ОПИСАНИЕ МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА

Модель разработки - водопад. Разработка системы ведется итерациями. Модель водопад выбрана поскольку:

1. Заранее известны требования к системе;
2. Архитектура проектируется полностью перед началом разработки и не будет кардинально меняться;
3. Технологический стек заранее согласуется с заказчиком на этапе проектирования;
4. Целью проекта является создание прототипа (Proof of the concept).



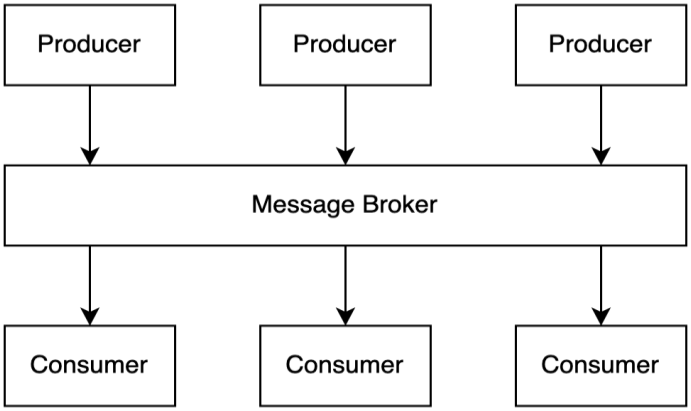
ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ

Producer-Consumer

Решаемые задачи:

1. Абстрагировать сервис-шлюз от хранилища событий;
2. Независимо масштабировать продюсера событий (сервис-шлюз) и консьюмера (хранилище событий);
3. Записывать события в хранилище асинхронно;
4. Гарантировать доставку событий.

Общее описание паттерна:



Последствия применения паттерна:

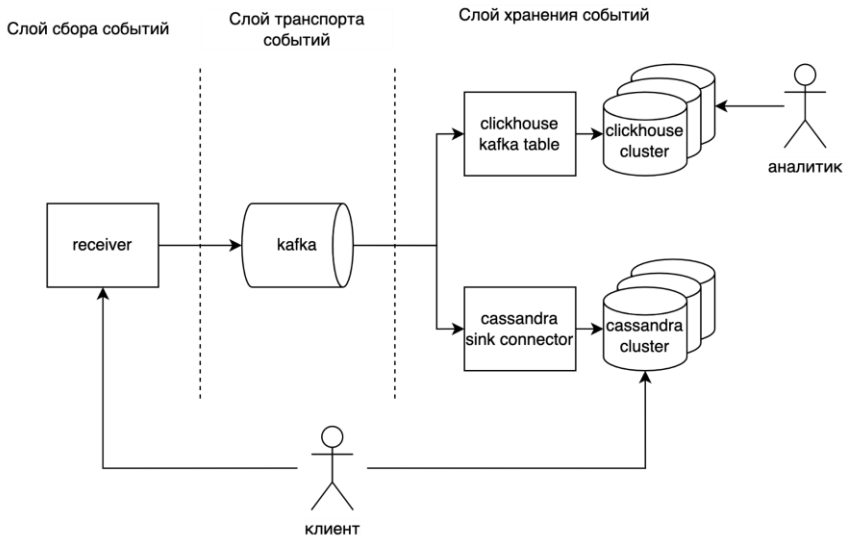
1. Слой сбора событий отделен от слоя хранения;
2. Сервисы сбора событий могут масштабироваться независимо от хранилища;
3. Хранилище может масштабироваться независимо от сервисов сбора событий;
4. К потоку событий могут подключаться другие подсистемы, а не только хранилище;
5. Появляется возможность обеспечивать работу сервиса под высокой нагрузкой на запись событий за счет применения асинхронной записи.

ОПИСАНИЕ ТИПА АРХИТЕКТУРЫ

Общее описание системы

Система разбита на три основных слоя:

1. Слой, отвечающий за сбор событий. Компонент состоит из группы одинаковых сервисов. Основная задача каждого сервиса принять запрос с описанием события, провалидировать формат и содержание запроса и быстро переложить запрос в персистентное хранилище. Источники событий отправляют события в один из сервисов этого компонента.
2. Шина событий. Этот компонент представляет из себя поток событий, из которого могут читать все подсистемы, которым требуется информация о поступающий событиях. Задачей этого компонента является временное хранение событий (порядка нескольких дней) и предоставление доступа к свежим событиям другим подсистемам.
3. Слой хранения событий. Задачей этого слоя является перманентное хранение всей истории событий и предоставление sql-доступа клиентам и аналитикам или системам, которым требуется аналитика по событиям.



Описание слоя сбора событий

Для сбора событий используется один или несколько сервисов, написанных на языке Java на фреймворке Spring с использованием сервера Embedded Jetty для обработки запросов.

Описание слоя транспортировки

Для временного хранения событий и асинхронной отправки в слой хранения используется Kafka. Kafka широко используется в микросервисных архитектурах. Разработано большое количество коннекторов для перенаправления сообщений из Kafka в различные хранилища, благодаря чему ее очень просто встроить в систему. Также для Kafka существует клиент на Java поддерживающий асинхронную запись. Поэтому было принято решение использовать Kafka для транспортировки событий в хранилища.

Описание слоя хранения

Слой хранения разбит на два компонента для двух сценариев работы:

1. Выполнение аналитических запросов;
2. Быстрое получение истории событий пользователей.

Для аналитических запросов используется Clickhouse. Clickhouse специально разрабатывался под подобные сценарии работы и заточен под высокую пропускную способность аналитических запросов на большом массиве данных.

Хранилища должны поддерживать высокую пропускную способность на запись. Для второго сценария работы также необходима высокая скорость чтения массива данных по ключу. Для удовлетворения этих двух критериев выбор делался среди хранилищ на основе LSM деревьев. Среди готовых решений таких как HBase, Cassandra, ScyllaDB, InfluxDB, RocksDB с учетом других ограничений было принято решение использовать Cassandra.

Заказчик

Личная подпись

Ответственный по проекту

Личная подпись

Дата: 26 марта 2022