Сервис обработки потока сообщений.

Сервис предназначен для подсчета агрегатов входящих сообщений в реальном времени. У сервиса есть источник сообщений `Source` и потребитель результатов `Sink`.

Сервис постоянно/непрерывно, пока запущено приложение, потребляет сообщения `SourceMessage` из источника `Source`.

Результат обработки отправляется сообщениями `SinkMessage` в получатель `Sink`.

В сервисе должен быть реализован pipeline, включающий четыре фазы - дедупликация, фильтрация, группировка и агрегация по временным окнам. Обработка происходит в том порядке, как это указано ниже.

> Обработка данных (выполнение пайплайна) происходит в памяти java процесса без использования внешних хранилищ.

Для каждого этапа обработки должна быть предусмотрена возможность конфигурирования параметров во внешнем конфигурационом файле - выбор формата на ваше усмотрение.

Интерфейсы `Source`, `SourceMessage`, `Sink`, `SinkMessage` для сервиса приведены в конце документа. В качестве имплементации `Source` должен быть использован Kafka Consumer из топика `SOURCE`. В качестве имплементации `Sink` должен быть использован Kafka Producer в топик `SINK`.

Пайплайн.

Дедупликация

Во входящих сообщениях в явном виде отсутствует ключ дедупликации. Дедупликация производится по произвольному (конфигурируемому) набору ключей из геттера `labels()`. По сути дедупликация использует тот же принцип, что и группировка (см. ниже) с дополнительным параметром (конфигурируемо) - время окна дедупликации. **Пример**: для сконфигурированых полей 'A' и 'B' - считается, что все сообщения с одинаковыми значениями для ключей 'A' и 'B' являются одним и тем же сообщением и должны быть дедуплицированы.

Фильтрация

Фильтрация сообщений описывается правилами (конфигурируемо). Отфильтрованные сообщение не участвуют в дальнейшей обработке. Если правило не задано - этап фильтрации пропускается.

В фильтрации могут участвовать любые поля входящего сообщения. Существует большой разброс от самых простых случаев, до самых сложных. Формат правил фильтрации оставляется на ваше усмотрение. Здесь большая вариативность, все ограничивается только вашей фантазией.

Группировка

Сообщения могут быть сгруппированы по набору полей `labels()` (по аналогии с SQL GROUP BY). Правило для группировки (конфигурируемо) - это список полей (ключи `labels()`). Если правило не задано - реализация на ваше усмотрение.

Агрегация по временным окнам.

Сгруппированные сообщения агрегируются с учетом групп и временных окон (внутри каждой группы!). Время окна (конфигурируемо) в секундах. В течение временного окна рассчитывается агрегат для каждой группы. Агрегат имплементирует интерфейс `SinkMessage`. По окончании временного окна агрегат отправляется в получатель `Sink`.

Ожидаемый результат

- Приложение компилируется/запускается на **Java 17** версии.
- Инструмент для сборки - на ваше усмотрение. Приветствуется **gradle**.
- Проект содержит конфигурационный файл для пайплайна и подключения к **Apache Kafka**.
- Использование фреймворков - на ваше усмотрение, НЕиспользование - приветствуется.
- Предполагается что вы реализуете сервис БЕЗ использования библиотек, реализующих пайплайн, по типу Kafka Streams.
- К результату должен быть приложен README с инструкцией по сборке запуску приложения.
- Исходный код размещен на github.com, история коммитов приветствуется.
- Если вы сомневаетесь в корректности задачи, нашли ошибку или противоречие - опишите свои сомнения в README.

Интерфейсы для имплементации

Источник сообщений

```
interface Source {
    Iterable<SourceMessage> source();
}
```

Входящее сообщение

```
interface SourceMessage {
    long timestamp();
    Map<String, String> labels();
    double value();
}
```

Получатель сообщений

```
interface Sink extends Consumer<SinkMessage> {
}
```

Обработанное сообщение

```
interface SinkMessage {
    long from();
    long to();
    Map<String, String> labels();
    double min();
    double max();
    double avg();
    int count();
}
```

Saymon 2025.03.17