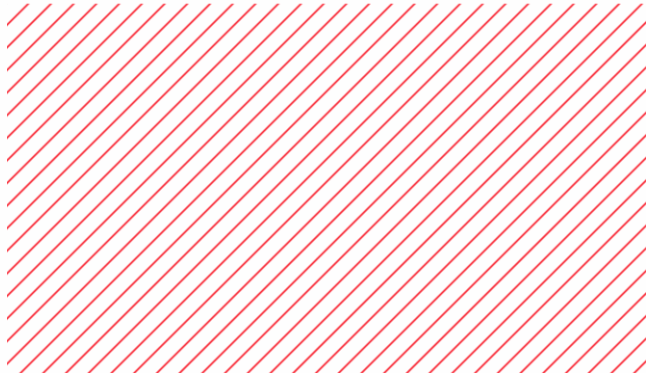
академия больших данных

mail.ru group

Потоковая обработка данных

Дмитрий Киселёв



План

- Мотивация
- Очереди сообщений и Apache Kafka
- Spark streaming
- Apache Flink

Мотивация

- Поток – естественный вид происхождения данных (логи, клики, транзакции и тд)
- Во многих ситуациях взаимодействие запрос-ответ не нужно (иногда невозможно)
    - Система не требует ответа (транзакции)
    - Фоновые процессы (антифрод, алерты)
    - Очень большая нагрузка
    - Боимся потерять данные
- Тем не менее нужно уметь обрабатывать такие данные с низкой задержкой
    - Пример: чем быстрее найдем мошенника, тем меньше потерь

Примеры

- Обработка транзакций
  - Процессинг карт
  - Поиск мошенников
- Мониторинг и алерты
- Аналитика
  - Ad-hoc анализ в реальном времени
- Подготовка данных
  - Подготовка real-time признаков для моделей
  - Обучение RL моделей в проде

Очередь сообщений

- Очередь сообщений реализует структуру данных First-in, First-out
- Обеспечивают независимость систем в точках сбора и обработки данных
- Гарантируют доставку сообщения от источника в обработчик
- Гарантируют порядок доставки

Apache Kafka

Apache Kafka – распределенная система потоковой обработки данных

Свойста

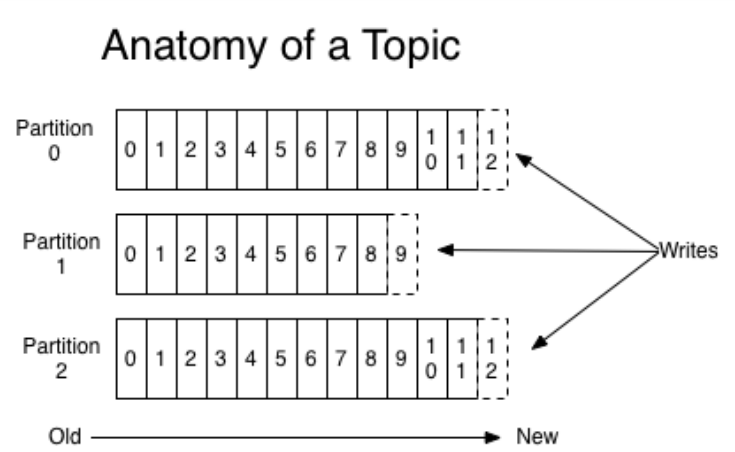
- Высокая доступность
- Отказоустойчивость
- Горизонтальная масштабируемость

Apache Kafka

Основные термины

- Генератор (Producer) – создает сообщения и публикует их в конкретную **тему**
- Тема (Topic) – упрощенно аналог папки в системе или таблицы в SQL
- Потребитель (Consumer) – может подписаться на тему и вычитывать из нее новые сообщения

Apache Kafka



- Данные в топике хранятся в партициях
- В каждой партиции сообщение лежит в виде журнала коммитов
- Каждому сообщению присваивается смещение (offset)
- Потребитель периодически опрашивает Kafka о появлении данных в топиках после определенного оффсета и смещает оффсет при получении
- Потребитель обычно привязан к конкретным партициям, так как обычно чтение происходит распределенно в пределах группы (consumer-group)
- Механизм репликации и выбора «ведущих» брокеров для партиций регулируется через ZooKeeper

Виды потоковой обработки данных

Real-time processing

- Низкие задержки
- Низкая пропускная способность
- Поэлементная обработка
- Нет потерь данных и дубликатов

Micro-batch processing

- Высокие задержки
- Высокая пропускная способность
- Пакетная обработка
- Могут терять или дублировать данные

https://hazelcast.com/glossary/micro-batch-processing/

Spark Streaming

- Использует микробатч подход
- API практически идентично DataFrame API
    - Можно использовать обученные оффлайн на батче пайплайны
- Основные отличия – чтения (spark.readStream) и запись (spark.writeStream) потоков (очереди, файлы и тд)
- Использует StreamingDataFrame

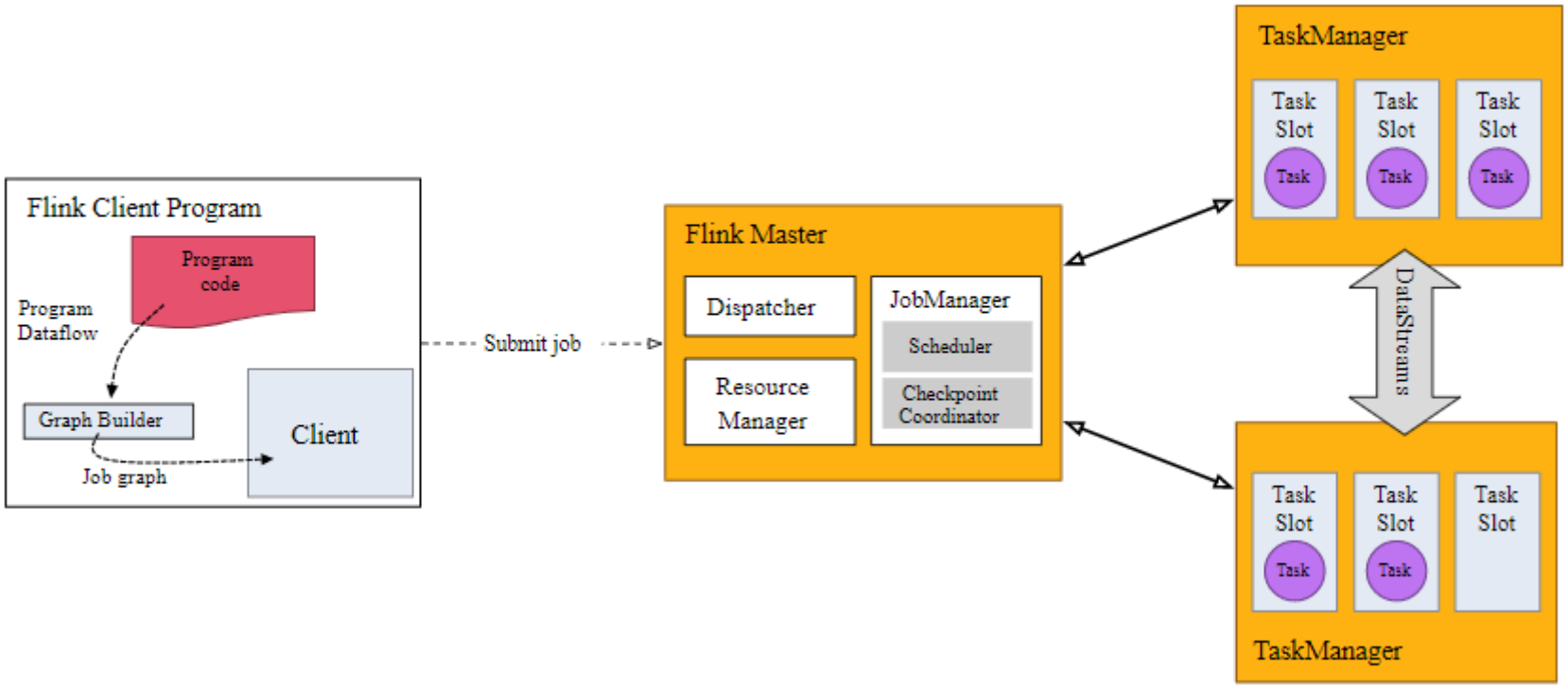
Поэтому мы не будем его разбирать)

Apache Flink

Еще одна система для работы с большим потоком данных

- Обрабатывает события поэлементно в реальном времени
- Stateful
- Распределенная
- Отказоустойчивая
- Высокодоступная

Flink: архитектура

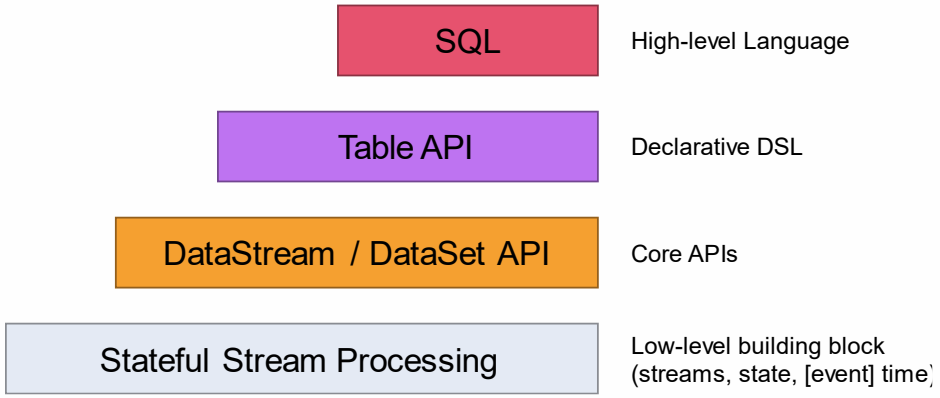


https://ci.apache.org/projects/flink/flink-docs-release-1.11/concepts/flink-architecture.html

Environment

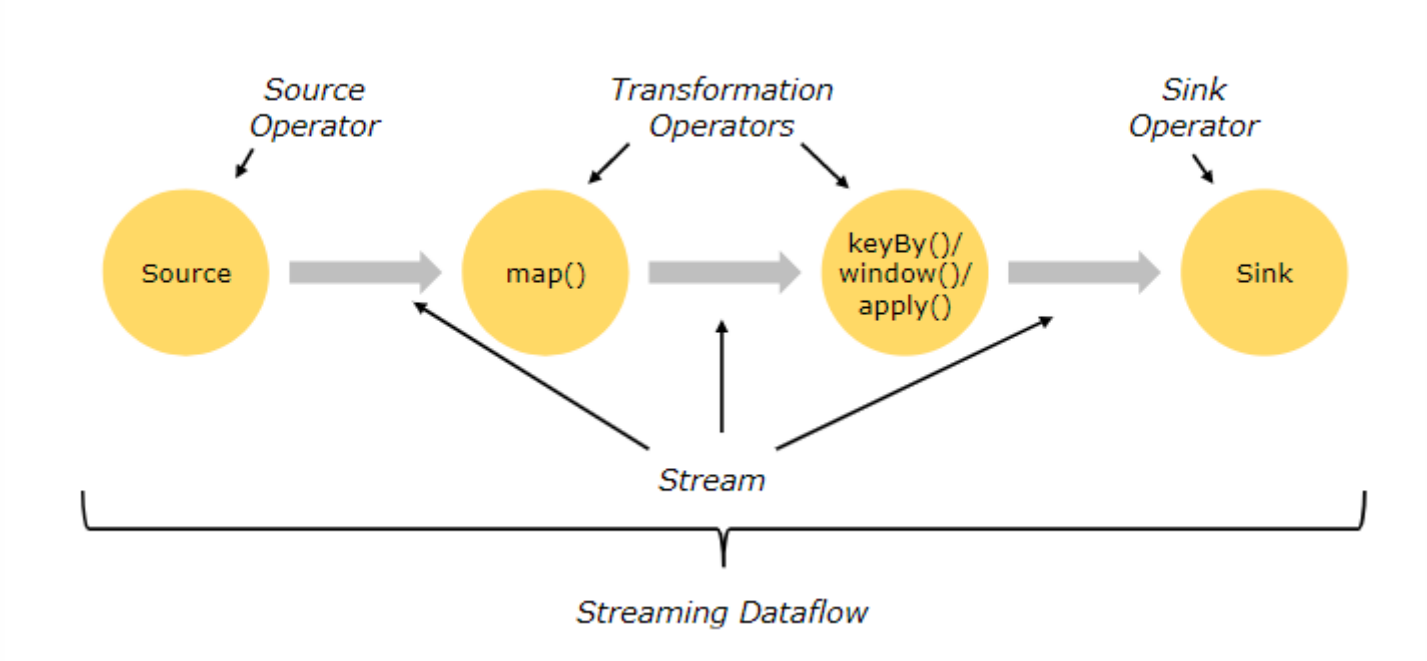
Для начала работы с Flink нужно создать среду построение графа вычислений

- **StreamExecutionEnvironment**
- ExecutionEnvironment
- BatchTableEnvironment / StreamTableEnvironment (есть версии для python)



https://ci.apache.org/projects/flink/flink-docs-release-1.11/concepts/index.html

Структура джобы



https://ci.apache.org/projects/flink/flink-docs-release-1.11/learn-flink/

Чтение (source) и запись (sink) данных

- Apache Kafka (source/sink)
- Amazon Kinesis Streams (source/sink)
- RabbitMQ (source/sink)
- Apache NiFi (source/sink)
- Google PubSub (source/sink)
- Twitter Streaming API (source)
- Apache Cassandra (sink)
- Elasticsearch (sink)
- Hadoop FileSystem (sink)
- JDBC (sink)

Трансформации

- Управление потоком (группировка по ключу, объединение)
- Изменение потока (операции над событиями: map, flatMap, …)
- Агрегация потока (операции над массивами событий: aggregate, sum, …)

Управление потоком

- keyBy – перекидывает элементы по ключу на одну машину. Из DataStream создает KeyedDataStream
- Union – создает новый поток из нескольких. DataStream -> DataStream
- Connect – создает «склеенный» поток из нескольких. DataStream -> ConnectedStreams
- Window – создает новый поток батчей из одного потока. DataStream -> WindowedStream
- Broadcast – создает новый поток, который будет одинаковый на всех партициях, BroadcastedStream

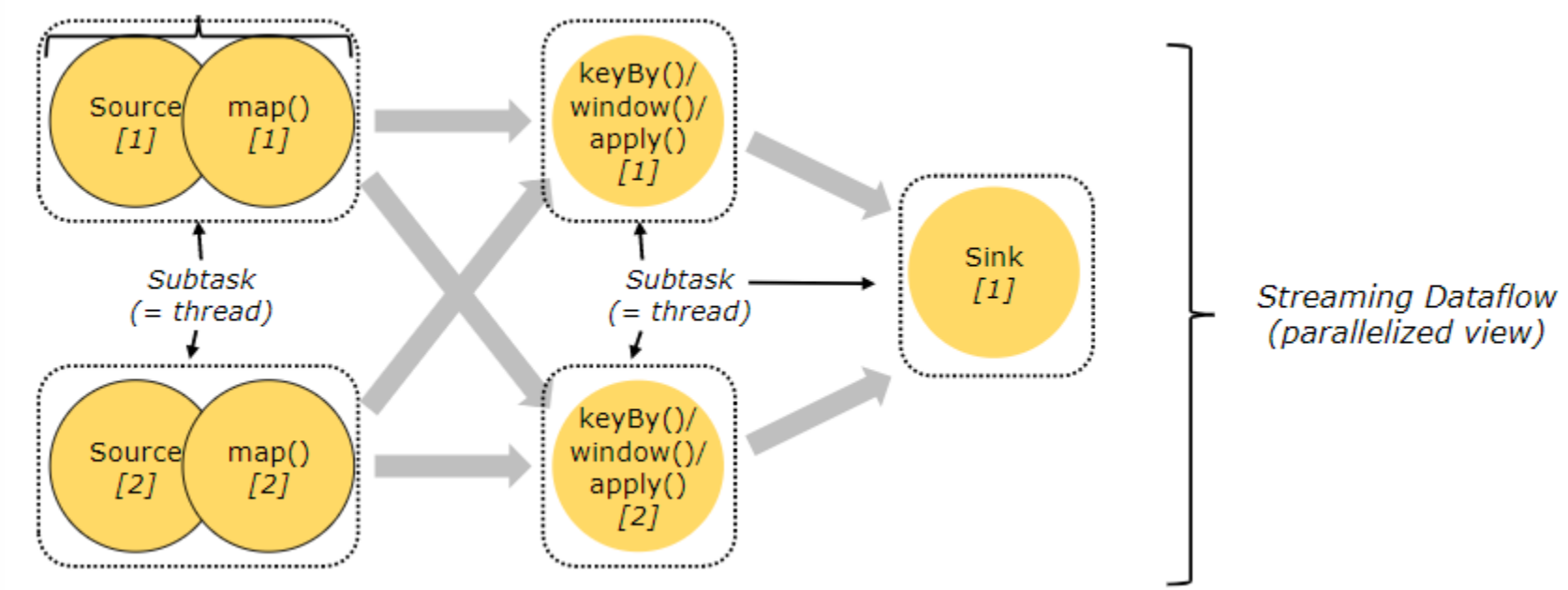
Изменение потока

- Map – получает и возвращает одно событие
- FlatMap – получает одно событие, возвращает несколько
- Filter – фильтрует поток по условию
- ProcessFunction – абстрактный класс для создания UDF

Также над разными типами потоков определены дополнительные функции

- KeyedProcessFunction – работает с группированными по ключу событиями. Rich функция, то есть обладает стейтом
- (Keyed)CoProcessFunction – для работы со склеенными потоками
- WindowFunction – работает над окнами

Управление потоком



https://ci.apache.org/projects/flink/flink-docs-release-1.11/concepts/flink-architecture.html

Агрегации

Для агрегаций важно уметь сохранять состояние. Например, если мы считаем среднее, то нам нужно знать количество сообщений и их сумму.

Если мы создаем окно, то нам нужно уметь сохранять несколько сообщений

Состояния (State)

Самое важное, что нужно знать про состояния – их лучше **использовать только на KeyedDataStream**

KeyedState хранится локально на дисках

Для состояния можно задать время жизни

Flink периодически сохраняет чекпоинты в общедоступную всем нодам директорию (S3, hdfs) для восстановления в случае падения.

Чекпоинт – оффсет для чтения потока и стейт, соответствующий этому оффсету

Агрегации (окна)

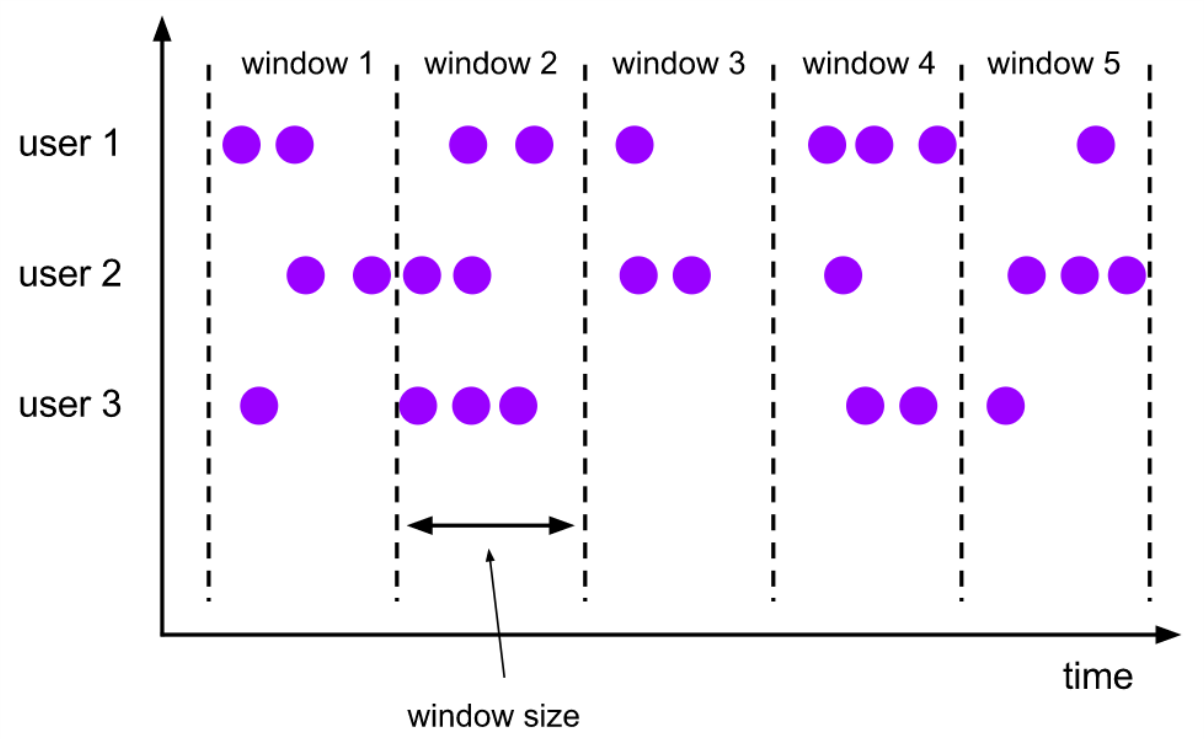
Непересекающиеся (Tumbling) окна

Скользящие (Sliding) окна

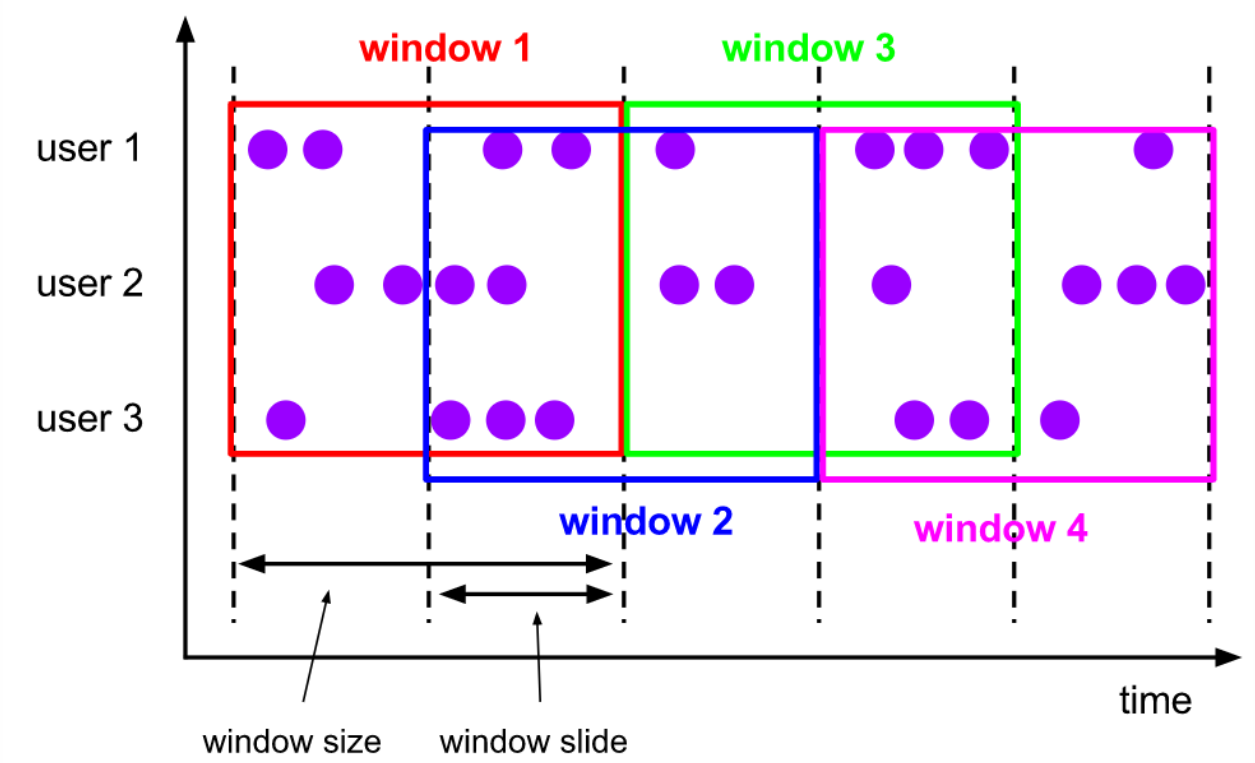
Сессионные (Session) окна

Окна позволяют использовать различные операторы агрегаций.

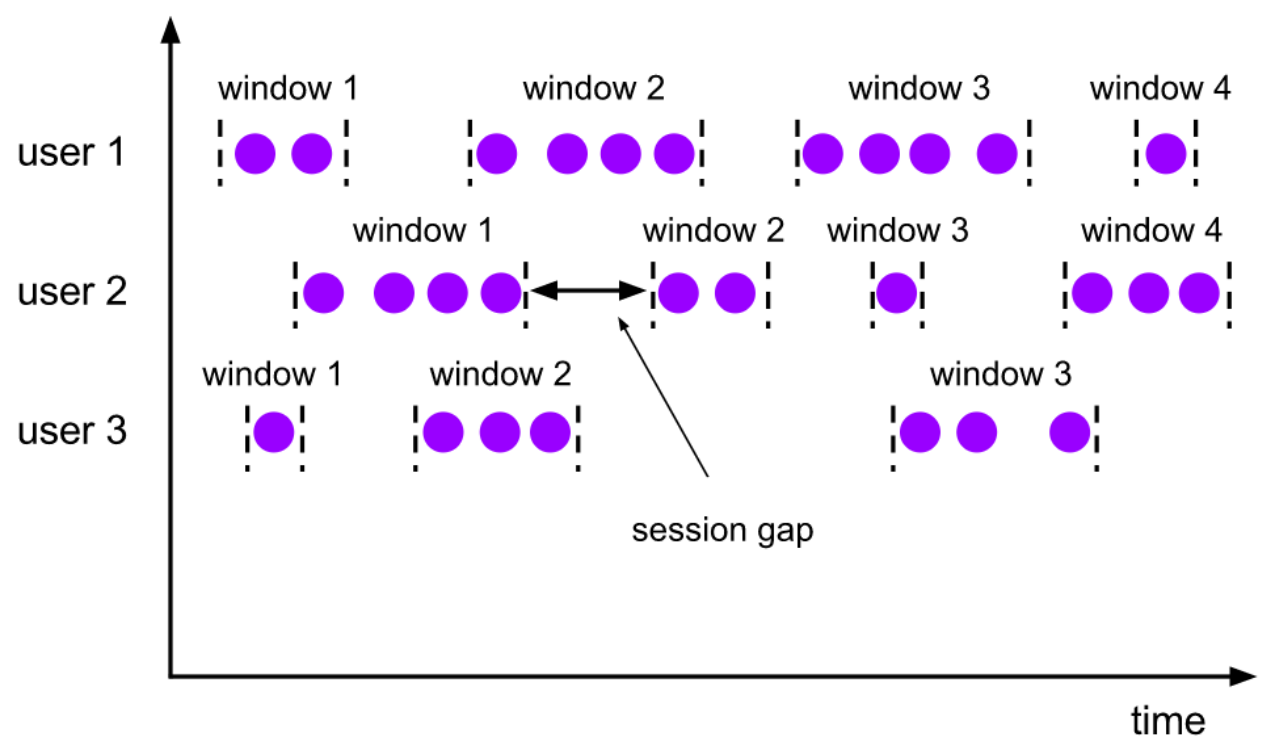
Непересекающиеся окна



Скользящие окна

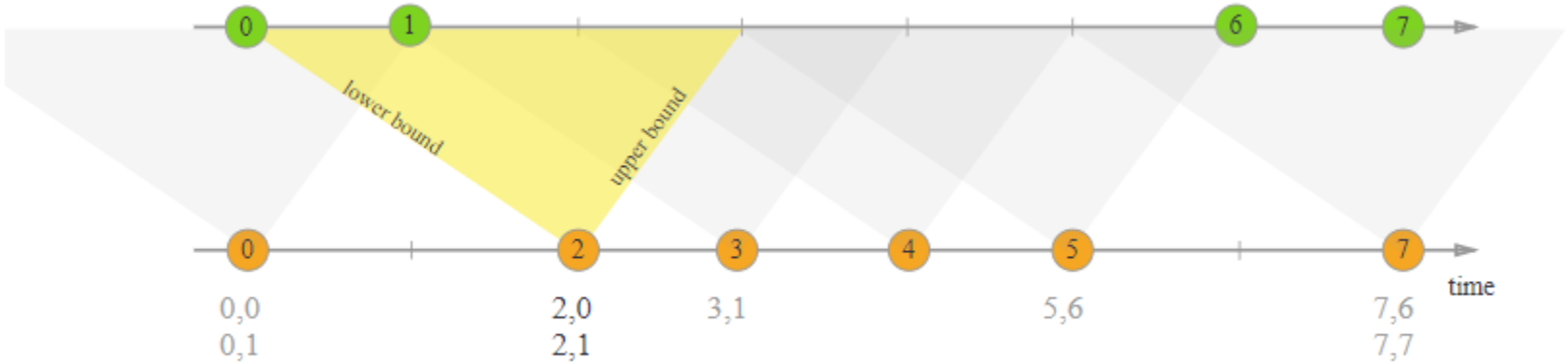


Сессионные окна



Join

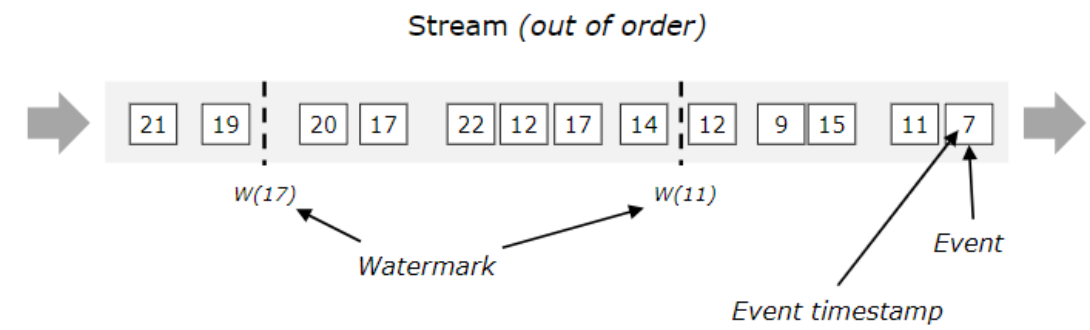
- WindowJoin – соединяет два события по ключу в окне
- IntervalJoin – соединяет потоки A и B так, чтобы временные метки событий из потока A лежали в интервале для потока B.



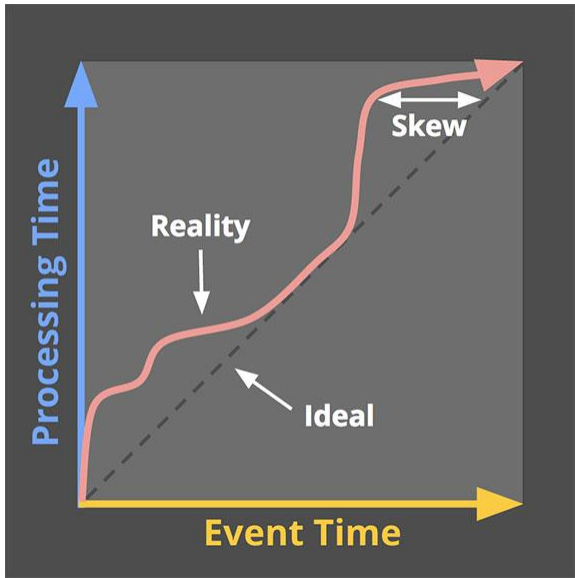
- Самое важное, что нужно знать – **НЕ ИСПОЛЬЗУЙТЕ НА БОЛЬШИХ ОКНАХ.** Джойны хранят все в оперативной памяти.

Время в потоковой обработке

- В потоковой обработке может быть важно использовать не текущее время обработки (ProcessingTime), а внутреннее время события (EventTime)
- Данные в потоке могут быть не упорядочены по времени
- Для работы с EventTime в потоковой обработке используются Watermarks
- Flink Source периодически вставляет Watermark(t) в поток. Это означает, что события со временем меньшим t больше не придут (не будут обработаны)



https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/



Выводы

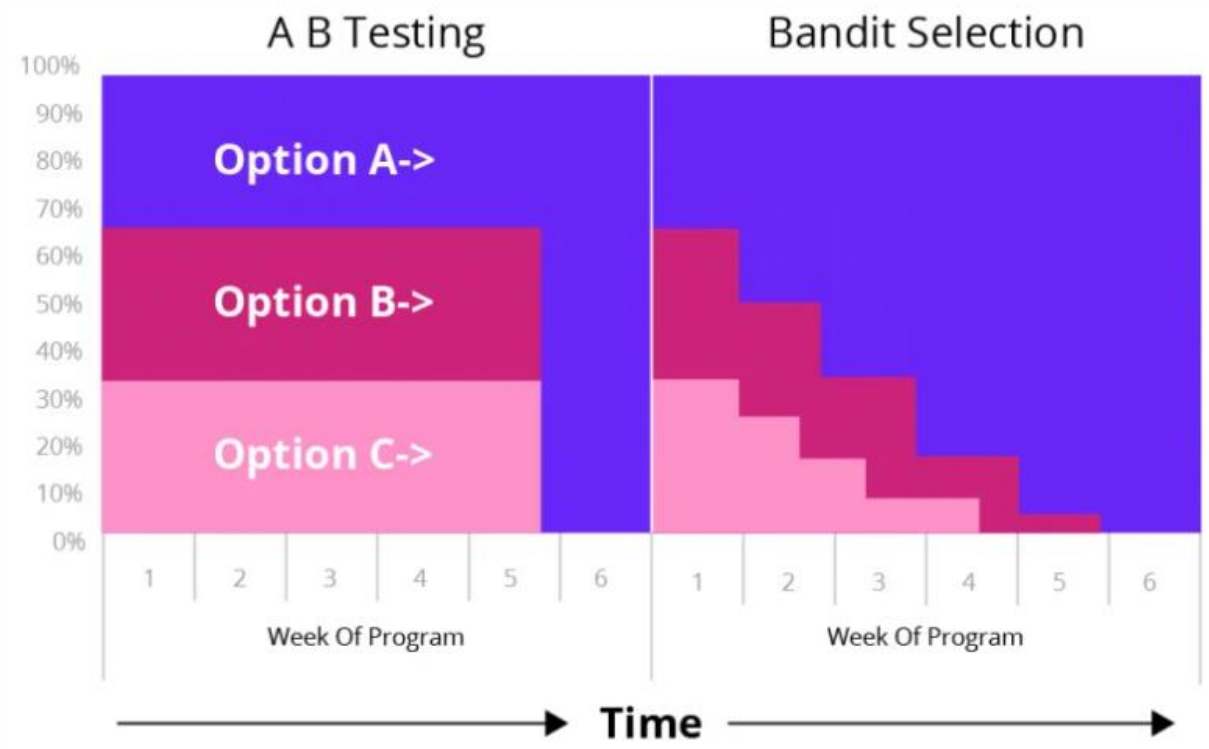
- Потоковая обработка много где полезна
- Бывает два вида потоковой обработки: микробатч и поэлементная
- Микробатч полезен в сервисах, где важна высокая пропускная способность, но не так критичны задержки, потеря или дублирование сообщений. SparkStreaming использует микробатч подход и апи аналогичное батчу
- Поэлементная обработка полезна, когда важны быстрая обработка, exactly-once consistency, но поток не такой объемный. Пример – Flink.
- Для передачи потоков можно использовать очереди сообщений, например, Apache Kafka
- Flink полезен в stateful поэлементной потоковой обработке данных

Семинар

На семинаре мы разработаем модель многорукого бандита UCB1

1. Сделаем простой генератор потока
2. Сделаем джоб для ранжирования
3. Сделаем генератор реакций пользователя
4. Сделаем расчет статистик для ранжирования

Многорукие бандиты



https://vwo.com/blog/multi-armed-bandit-algorithm/

UCB1

UCB1 – Upper Confidence Bound

Для ранжирования будем использовать формулу

$$Score_j = \overline{reward_j} + \sqrt{\frac{2\ln(Total\ number\ of\ shows)}{number\ of\ shows\ for\ j}}$$

То есть все, что нам нужно – хранить счетчики показов для каждой руки

Семинар

На семинаре мы разработаем модель многорукого бандита UCB1

1. Сделаем простой генератор потока
2. Сделаем джоб для ранжирования
3. Сделаем генератор реакций пользователя
4. Сделаем расчет статистик для ранжирования (обновления)

Генератор потока

- Создает поток запросов пользователей с конкретными предпочтениями
  - Будем семплировать из фиксированного количества пользователей
- Создает поток обратной связи на рекомендации
  - Случайно создаем ответ в зависимости от предпочтений
  - Предпочтения будем использовать одинаковые для всех – id руки делить на сумму id.

Ранжирование

- Принимает на вход поток запросов
- Принимает на вход статистики по разным рукам
- Создает поток с предсказаниями

Расчет статистик

- Агрегирует данные в сессионное окно по пользователю
- Считает количество действий по рукам

Семинар

Для начала работы создадим тестовый проект

- sbt new tillrohrmann/flink-project.g8
- Itellij -> project from existing source

- А дальше пишем вместе код)