**Потоки данных в DKB под управлением Apache Kafka**

*версия от 02.02.2017 (Голосова М.)*

Оглавление

[Программные узлы в потоке данных 1](#_Toc473818815)

[Source 1](#_Toc473818816)

[Processor 2](#_Toc473818817)

[Sink 2](#_Toc473818818)

[Требования к внешним программам 2](#_Toc473818819)

[Общие требования 2](#_Toc473818820)

[Source 2](#_Toc473818821)

[Processor 4](#_Toc473818822)

[Sink 5](#_Toc473818823)

[Запуск узлов 5](#_Toc473818824)

[Существующий узел 5](#_Toc473818825)

[Новые узлы 5](#_Toc473818826)

[Source 5](#_Toc473818827)

[Processor 6](#_Toc473818828)

[Sink 6](#_Toc473818829)

Программные узлы в потоке данных

Source

Тип узла: источник данных

Назначение: получение данных из внешнего источника

На стороне Kafka: \*SourceConnector, runPipeConnector

Доступные коннекторы:

* External (запускает внешнюю программу и отвечает за получание из неё данных, запись данных в Kafka)
* JDBC (получает данные из RDB)
* ~~Pipe (читающий данные из pipe’а)~~  (устарело)

Processor

Тип узла: потоковый процессор / обработчик

Назначение: преобразование данных

На стороне Kafka: Stream Application

Доступные приложения:

* runExternalProcessor (запускает внешнюю программу и отвечает за направление данных в неё / из неё)

Sink

Тип узла: сток данных

Назначение: вывод данных их Kafka, запись их во внешние хранилища

На стороне Kafka: \*SinkConnector

Доступные коннекторы:

* External (запускает внешнюю программу и отвечает за направление данных из Kafka в неё)
* ~~JDBC~~ (ещё не протестировано)
* ~~HDFS~~ (ещё не протестировано)

Требования к внешним программам

Общие требования

1. Работа со стандартными потоками данных:
   1. STDIN: поток входных данных для обработки
   2. STDOUT: поток выходных данных (результатов работы программы)
   3. STDERR: информационные сообщения и сообщения об ошибках
2. Код возврата:
   1. =0 – успешное (штатное) завершение программы
   2. ≠0 – завершение с ошибкой

Source

Общие требования:

1. Входные данные: нет
2. Сообщения об ошибках и информационные сообщения: выводятся на STDERR

При работе с ExternalSourceConnector:

1. Выходные данные: выводятся на STDOUT
2. Известные типы данных:
   1. string: передаются в Kafka-топик как есть
   2. json: проверяются на соответствие синтаксису; передаются в Kafka-топик строкой
   3. jsonArray: проверяются на соответствие синтаксису; передаются в Kafka-топик набором строк, соответствующих элементам массива
3. Рабочий цикл:
   1. программа стартует
   2. получает данные из внешнего источника
   3. выводит полученные данные на STDOUT
   4. завершается (код возврата: 0)
4. В случае возникновения ошибок:
   1. отправляет информационное сообщение на STDERR
   2. завершается с кодом возврата, отличным от 0

При работе с runPipeConnector:

1. Выходные данные: выводятся в файл FILE
2. Параметры командной строки:
   1. (опционально, для запуска из runPipeConnector) -p|--pipe PIPE
      1. создать (или проверить наличие) соответствующий именованный канал (named pipe);
      2. использовать PIPE в качестве файла для вывода выходных данных
3. Рабочий цикл:
   1. программа стартует:
      1. по cron’у
      2. из runPipeConnector
      3. запускается вручную (manual)
   2. получает данные из внешнего источника
   3. записывает данные в файл/пайп (FILE / PIPE)
   4. далее:
      1. (cron) завершается
      2. (runPipeConnector, manual) переходит в режим ожидания; по внутренним условиям (таймер etc) переходит к п. b).

Processor

При работе с runExternalProcessor:

1. Входные данные: 1 строка (= 1 запись из Kafka-топика)
2. Выходные данные:
   1. 1 и более строк, соответствующих записям в выходном Kafka-топике;
   2. EOPMarker (см. plain-topology.properties) (в начале строки) -- контрольный символ, означающий, что процессор закончил обработку полученной строки и готов принимать новые данные. После этого символа не должно следовать никаких данных: они не будут приняты до тех пор, пока не начнётся обработка следующего сообщения, и могут быть потеряны.
3. Сообщения об ошибках и информационные сообщения: выводятся на STDERR
4. Рабочий цикл:
   1. программа стартует
   2. переходит в режим ожидания входных данных
   3. при получении новой строки данных производит обработку данных
   4. в процессе и / или по завершении обработки на STDOUT отправляются результаты обработки данных для передачи далее в поток
   5. по завершении передачи выходных данных в STDOUT отправляется служебный символ EOPMarker
   6. переходит к п. b).

Sink

При работе с ExternalSinkProcessor:

1. Входные данные:
   1. 1 строка (= 1 запись из Kafka-топика)
   2. (опционально) ‘\0’ -- служебный символ, сигнализирующий, что завершена передача набора строк; может использоваться для разбивки входных сообщений на наборы, передаваемые в хранилище в рамках одной транзакции / одного запроса / …
2. Выходные данные: нет
3. Сообщения об ошибках и информационные сообщения: выводятся на STDERR
4. Рабочий цикл:
   1. программа стартует
   2. переходит в режим ожидания входных данных
   3. при получении новой строки (или набора строк, завершающихся служебным символом) производит отправку данных во внешнее хранилище
   4. переходит к п. b).
5. В случае возникновения ошибок завершается с кодом возврата, отличным от 0.

Запуск узлов

Существующий узел

Dataflow/000\_kafka/run.sh MyStageName {start|stop|restart}

Новые узлы

Source

1. Создать файл с параметрами процесса KafkaConnect  
   Пример: Dataflow/000\_kafka/config/connect-sources-standalone.properties  
   Основные параметры:
   1. rest.port (уникальный для каждого процесса KafkaConnect)

Опциональные параметры:

* 1. {key|value}.converter (в соответствии с форматом топика):
     1. org.apache.kafka.connect.storage.StringConverter -- pure String
  2. org.apache.kafka.connect.json.JsonConverter -- Json data
  3. {key|value}.converter.schemas.enable (в соответствии с форматом топика)
  4. offset.storage.file.filename (на данный момент, ExternalSourceConnector работает только с программами, выполняющими загрузку всех данных за один раз и, как следствие, не нуждается в том, чтобы хранить смещение. Значением этого параметра может быть что угодно; главное – не удалять его)
  5. offset.flush.interval.ms (как часто сохранять текущее положение смещения. Поскольку ExternalSourceConnector не работает со смещением, интервал стоит поставить побольше)

1. Создать файл с параметрами SourceConnector’а.  
   Пример: Dataflow/000\_kafka/config/example-source.properties  
   Основные параметры:
   1. name (уникальный идентификатор задачи)
   2. topics (Kafka-топики, из которых данные будут направлены на вход внешней команды)
   3. external.command (строка запуска внешней команды)
   4. poll.interval.ms (как часто необходимо запускать внешнюю программу)
2. Запуск узла:  
   $KAFKA\_HOME/bin/connect-standalone.sh myConnect.properties\  
   mySource.properties

Processor

cd Dataflow/000\_kafka/

java -cp $CLASSPATH ru/kiae/dkb/kafka/streams/runExternalProcessor \  
 -O MY\_OUTPUT\_TOPIC -S MY\_INPUT\_TOPIC \  
 -c "../099\_myStage/myStage.py --opt1 VAL1 ..."

Sink

1. Создать файл с параметрами процесса KafkaConnect  
   Пример: Dataflow/000\_kafka/config/connect-sink-standalone.properties  
   Основные параметры:
2. rest.port (уникальный для каждого процесса KafkaConnect)

Опциональные параметры:

1. {key|value}.converter (в соответствии с форматом топика):
   * 1. org.apache.kafka.connect.storage.StringConverter -- pure String
     2. org.apache.kafka.connect.json.JsonConverter -- Json data
2. {key|value}.converter.schemas.enable (в соответствии с форматом топика)
3. offset.storage.file.filename (для SinkConnector’ов смещения хранятся в служебном топике \_\_consumer\_offsets, указанный здесь файл будет проигнорирован. Значением этого параметра может быть что угодно; главное – не удалять его).
4. offset.flush.interval.ms (как часто сохранять текущее положение смещения)
5. Создать файл с параметрами SinkConnector’а.  
   Пример: Dataflow/000\_kafka/config/example-sink.properties  
   Основные параметры:
   1. name (уникальный идентификатор задачи)
   2. topic (Kafka-топик, в который будут публиковаться данные)
   3. external.command (строка запуска внешней команды)

Опциональные параметры:

* 1. batch.size:
* N>0 : (максимальное) количество сообщений, которые должны быть объединены в “набор” сообщений при отправке внешнему процессу.  
  Сообщения отправляются одно за другим, и служебный символ '\0' отмечает конец набора сообщений;
* N=0 : автоматическое определение количества сообщений в наборе (количество сообщений в наборе будет равно количеству сообщений, прочитанных из исходных топиков за один раз);
* N<0 : отключить объединение сообщений в наборы (‘\0’ не отправляется)

1. Запуск узла:  
   $KAFKA\_HOME/bin/connect-standalone.sh myConnect.properties mySink.properties