Route Throttling Example

Route Throttling Example

Available as of Camel 2.1

About

This example shows how to use the new feature RoutePolicy to dynamically at runtime to throttle routes based on metrics gathered by the current number of inflight exchanges.

What it means is that Camel will dynamic throttle the routes based on the flow of messages being processed at runtime.

The example have 3 routes where as two of the routes are input routes, and the last is the processing route
1. route1: from("jms") - input from a JMS queue
2. route2: from("file") - input from a file folder
3. route3: from("seda") - processing of the input messages, this one simulates CPU work by delaying the messages 100 mills.

How it works

When the example runs we have a dependency from the routes as follows:

- route1 -> route3
- route2 -> route3

What the example demonstrates is that Camel is capable of dynamic throttling route1 and route2 based on the total flow of messages. At runtime Camel gathers the current inflight exchanges in a registry which is used for metrics.

So when the flow is going *too fast* then the RoutePolicy kicks in and `suspends` route1 and/or route2. The current inflight exchanges will continue to be processed and when we are below a *threshold* then the RoutePolicy kicks in again and `resumes` route1 and/or route2.

Camel provides the throttling policy in the `org.apache.camel.impl.ThrottlingInflightRoutePolicy`.

How to run

The example has 3 maven goals to run the example

`mvn compile exec:java -PCamelServer` - starts the Camel Server which contains the 3 routes and where you should check its log output for how it goes.

`mvn compile exec:java -PCamelClient` - is a client that sends 10000 JMS messages to the JMS broker which is consumed by route1. The Server must be started beforehand.

`mvn compile exec:java -PCamelFileClient` - is a client that creates 5000 files that are consumed by route2. The server may be started beforehand, but its not required.

So at first you start the server. Then at any time you can run a client at will. For example you can run the JMS client and let it run to completion at the server. You can see at the server console logging that it reports the progress. And at sometime it will reach 10000 messages processed. You can then start the client again if you like.

You can also start the other client to create the files which then let the example be a bit more complicated as we have concurrent processing of JMS messages and files at the same time. And where as both of these should be dynamic throttled so we wont go *too fast*.

How to change

You can check the file `src/main/resources/META-INF/spring/camel-server.xml` file where you can see the configuration of the dynamic throttler. By default its configured as:

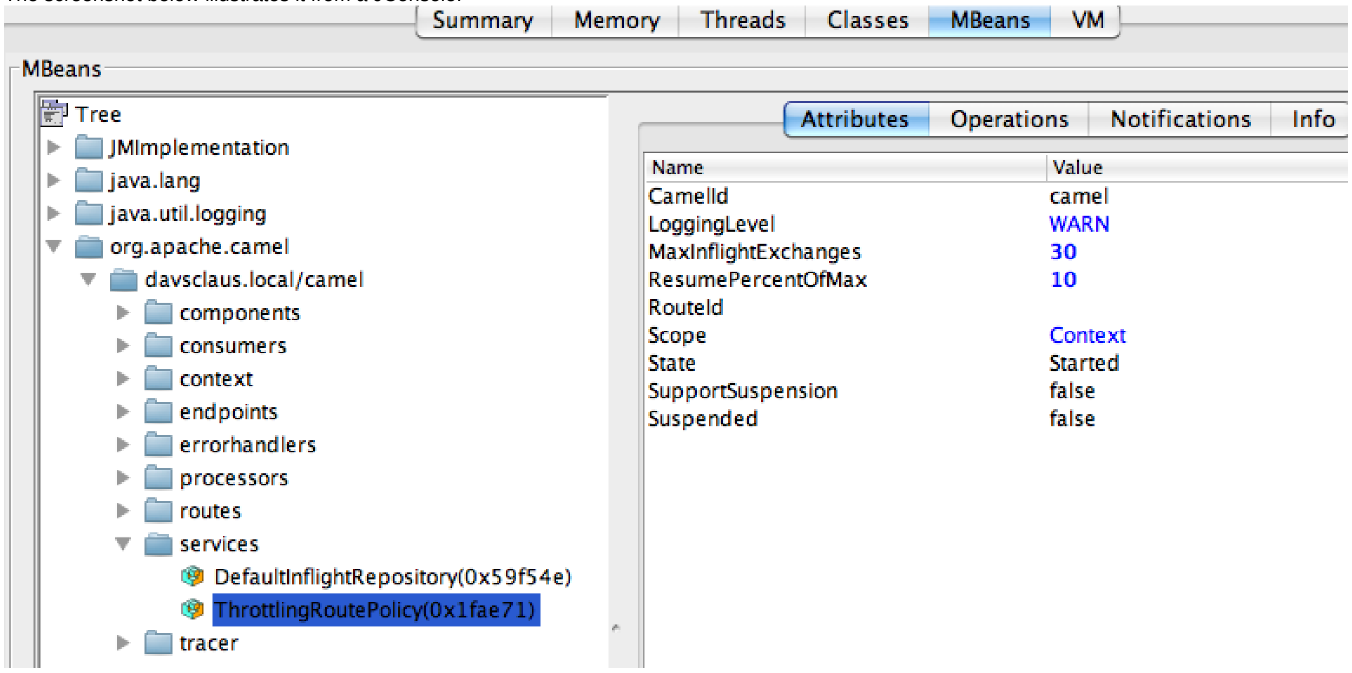
Error formatting macro: snippet: java.lang.NullPointerException

You can then change this and restart the server to see the changes in effect.

JMX management

At runtime you can manage the `ThrottlingInflightRoutePolicy` at runtime as its listed under services in the JConsole.
For example you can change the option `maxInflightExchanges` while its running to find a more suitable value.

The screenshot below illustrates it from a JConsole.



Name	Value
CamelId	camel
LoggingLevel	WARN
MaxInflightExchanges	30
ResumePercentOfMax	10
RouteId	
Scope	Context
State	Started
SupportSuspension	false
Suspended	false

See more at using JMX with Camel.

Sample output from running example

When running the example you should see a lot of activity logged to the console.
Below is a snippet when the example runs to an end when the 10000 messages have been processed. What you should notice is that the throttling ensures that the JMS consumer is not taking in more messages than we can process. That means that ++JMS++ and ++SEDA++ are completing at nearly the same time.

```
[ultMessageListenerContainer-42] +++JMS +++                     INFO  Received: 9800 messages so far. Last
group took: 673 millis which is: 148.588 messages per second. average: 159.734
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 25 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9800 messages so far. Last
group took: 733 millis which is: 136.426 messages per second. average: 159.789
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 25 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 25 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[ultMessageListenerContainer-55] +++JMS +++                     INFO  Received: 9900 messages so far. Last
group took: 758 millis which is: 131.926 messages per second. average: 159.395
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 25 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9900 messages so far. Last
group took: 598 millis which is: 167.224 messages per second. average: 159.86
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 24 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 25 > 20 inflight
exchange by suspending consumer.
[currentConsumers=25&size=25000] ThrottlingInflightRoutePolicy  WARN  Throttling consumer: 2 <= 2 inflight
exchange by resuming consumer.
[ultMessageListenerContainer-76] +++JMS +++                     INFO  Received: 10000 messages so far. Last
group took: 732 millis which is: 136.612 messages per second. average: 159.129
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 10000 messages so far. Last
group took: 732 millis which is: 136.612 messages per second. average: 159.589
```

However if you run the example **without** the throttling you will notice that the JMS consumer runs faster than we can process the messages. Towards the end we have more than 2000 messages pending on the internal SEDA queue, when the JMS consumer finishes.

```
[ultMessageListenerContainer-41] +++JMS +++                     INFO  Received: 9800 messages so far. Last
group took: 304 millis which is: 328.947 messages per second. average: 225.272
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 7800 messages so far. Last
group took: 645 millis which is: 155.039 messages per second. average: 178.461
[ultMessageListenerContainer-65] +++JMS +++                     INFO  Received: 9900 messages so far. Last
group took: 543 millis which is: 184.162 messages per second. average: 224.765
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 7900 messages so far. Last
group took: 438 millis which is: 228.311 messages per second. average: 178.956
[ultMessageListenerContainer-65] +++JMS +++                     INFO  Received: 10000 messages so far. Last
group took: 395 millis which is: 253.165 messages per second. average: 225.017
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8000 messages so far. Last
group took: 408 millis which is: 245.098 messages per second. average: 179.561
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8100 messages so far. Last
group took: 410 millis which is: 243.902 messages per second. average: 180.148
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8200 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 180.744
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8300 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 181.334
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8400 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 181.913
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8500 messages so far. Last
group took: 666 millis which is: 150.15 messages per second. average: 181.461
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8600 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 182.022
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8700 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 182.577
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8800 messages so far. Last
group took: 407 millis which is: 245.7 messages per second. average: 183.112
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 8900 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 183.649
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9000 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 184.177
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9100 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 184.693
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9200 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 185.2
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9300 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 185.703
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9400 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 186.194
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9500 messages so far. Last
group took: 405 millis which is: 246.914 messages per second. average: 186.677
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9600 messages so far. Last
group took: 408 millis which is: 245.098 messages per second. average: 187.142
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9700 messages so far. Last
group took: 413 millis which is: 242.131 messages per second. average: 187.581
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9800 messages so far. Last
group took: 410 millis which is: 243.902 messages per second. average: 188.024
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 9900 messages so far. Last
group took: 579 millis which is: 172.712 messages per second. average: 187.856
[currentConsumers=25&size=25000] +++SEDA+++                     INFO  Received: 10000 messages so far. Last
group took: 404 millis which is: 247.525 messages per second. average: 188.31
```

So by using the `ThrottlingInflightRoutePolicy` we can throttle the intake of messages to be on a pair with the speed we can process messages. And since its pluggable you can implement your own custom logic to throttle according to your needs.

See Also

- Examples
- RoutePolicy