JMS performance and pooling

CXF JMS should be very fast if configured correctly. There are two major settings that affect performance: pooling and synchronous receives on client side.

Pooling

On the client side CXF creates a new JMS Session and Producer for each message. This is necessary as these JMS objects are not thread safe. Especially creating a Producer is very time intensive though as a communication with the server is necessary.

Connection Factory pools help to improve perfomance by caching the Connection, Session and Producer.

For ActiveMQ configuring pooling is quite simple:

Pooling for ActiveMQ

```
import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.activemq.pool.PooledConnectionFactory;

ConnectionFactory cf = new ActiveMQConnectionFactory("tcp://localhost:61616");
PooledConnectionFactory pcf = new PooledConnectionFactory();
pcf.setExpiryTimeout(5000); // Make sure to set expiry timeout as the default of 0 prevents reconnect on
failures
pcf.setConnectionFactory(cf);

JMSConfiguration jmsConfig = new JMSConfiguration();
jmsConfig.setConnectionFactory(pcf);
```

Synchronous Receives on client side

For request / reply exchanges the JMS transport sends a request and waits for a reply. Whenever possible this is done using a JMS MessageListener. In some cases though CXF has to use a synchronous Consumer.receive() instead. The problem here is that CXF needs to be able to share queues between endpoints. So a MessageListener needs to use a MessageSelector to filter the messages. This selector needs to be known in advance so the listener can be opened once.

There are two cases where this is not possible:

1. MessageIdAsCorrelationId: If useConduitIdSelector is false and no static conduitIdSelectorPrefix is given then the client does not set a correlation id. The server then uses the message id of the request message as correlation id. As this id can not be known in advance the client has to use a synchronous Consumer.receive. This pattern is also necessary for communication with IBM JMS endpoints as they use this by default
2. The user can set the JMSMessageHeadersType in the request message and there do a SetJMSCorrelationID("customid"). In this case CXF also has to use a synchronous Consumer.receive()

So the general rule here is to avoid these settings that lead to the synchronous receives.

Expected perfomance

On a fast system (Intel Core i7) and with small messages CXF can achieve a performance of about 12,000 messages per second for one way and 3,500 transactions / second for request reply (both non persistent). If you measure much lower speeds then you might have an issue with your JMS configuration.