Kepler - Bug #5378

add rety logic to SpanTodt, attempt to recover if RBNB goes down

04/13/2011 03:45 PM - Derik Barseghian

Status:	Resolved	**Start date:**	04/13/2011
Priority:	Normal	**Due date:**	
Assignee:	Derik Barseghian	**% Done:**	0%
Category:	sensor-view	**Estimated time:**	0.00 hour
Target version:	sensor-view-0.9.0	**Spent time:**	0.00 hour
Bugzilla-Id:	5378		

Description

We've found that requesting a large amount of data from RBNB on a gumstix, at least via rdv (1.9.0 and 2.2.1), can get the server into a messed up state, requiring a restart, and the RBNB files written to disk will become corrupt such that a restart of rbnb fails to load the old data (yikes - we'll work on a replication procedure and submit these bug(s) to the rbnb list). When this happen SpanTodt locks up. Even when rbnb is successfully restarted (e.g. by deleting rbnb files on disk and starting fresh), SpanTodt stays locked. We should add some retry logic so spanToDT can attempt to persevere through tough times.

History

#1 - 04/13/2011 03:48 PM - Derik Barseghian

This is the connection error I got from SpanTodt when RBNB became hosed:

```
ERROR flushing.
com.rbnb.sapi.SAPIException: This operation requires a connection.
at com.rbnb.sapi.Client.assertConnection(Client.java:634)
at com.rbnb.sapi.Source.Flush(Source.java:295)
at com.rbnb.sapi.Source.Flush(Source.java:258)
at util.SpanToDT$PeriodicFlushAndDetachThread.run(SpanToDT.java:602)
Error writing to DT.
com.rbnb.sapi.SAPIException: Nesting java.lang.IllegalStateException
at com.rbnb.sapi.Client.OpenRBNBConnection(Client.java:312)
at com.rbnb.sapi.Client.OpenRBNBConnection(Client.java:250)
at util.SpanToDT$WriteToDataTurbineThread._createChannelMap(SpanToDT.java:512)
at util.SpanToDT$WriteToDataTurbineThread._getChannelIndex(SpanToDT.java:558)
at util.SpanToDT$WriteToDataTurbineThread.run(SpanToDT.java:408)

Nested exception:
java.lang.IllegalStateException: Cannot reconnect to existing client handler /dataturbine/gpp-data.
at java.lang.reflect.Constructor.constructNative(Native Method)
at java.lang.reflect.Constructor.newInstance(Constructor.java:328)
at com.rbnb.api.ExceptionMessage.toException(ExceptionMessage.java:441)
at com.rbnb.api.Language.throwException(Language.java:1679)
at com.rbnb.api.ACO.receive(ACO.java:1531)
at com.rbnb.api.ACO.receive(ACO.java:1415)
at com.rbnb.api.ACO.login(ACO.java:1188)
at com.rbnb.api.SerializingACO.login(SerializingACO.java:315)
at com.rbnb.api.ACO.start(ACO.java:1959)
at com.rbnb.api.ClientHandle.start(ClientHandle.java:459)
at com.rbnb.sapi.Source.doOpen(Source.java:231)
at com.rbnb.sapi.Client.OpenRBNBConnection(Client.java:307)
at com.rbnb.sapi.Client.OpenRBNBConnection(Client.java:250)
at util.SpanToDT$WriteToDataTurbineThread._createChannelMap(SpanToDT.java:512)
at util.SpanToDT$WriteToDataTurbineThread._getChannelIndex(SpanToDT.java:558)
at util.SpanToDT$WriteToDataTurbineThread.run(SpanToDT.java:408)
ERROR flushing.
com.rbnb.sapi.SAPIException: This operation requires a connection.
at com.rbnb.sapi.Client.assertConnection(Client.java:634)
at com.rbnb.sapi.Source.Flush(Source.java:295)
at com.rbnb.sapi.Source.Flush(Source.java:258)
at util.SpanToDT$PeriodicFlushAndDetachThread.run(SpanToDT.java:602)
```

#2 - 04/13/2011 07:19 PM - Derik Barseghian

in r27514 added code for sleeping and retrying the connection to DT during startup.
This doesn't fix the issue of spanToDT jamming when a flush fails due to DT going down later.
This also doesn't address spanToDT jamming if span is down.

I'm debating if it's cleaner to add sleep and retry code in all the necessary places, or if we should simply die more frequently, and let spanToDT (and maybe also span and rbnb) be managed by something like daemontools (http://cr.yp.to/daemontools.html).

#3 - 06/14/2012 06:30 PM - Derik Barseghian

Should be fixed with r29937, r29939, r29950, 29951, with the caveat I've only tested using the simulator as SPAN. Leaving open for testing against real SPAN.

SpanToDT can now perserve when DataTurbine stops and then restarts, when SPAN stops and then restarts, or when both go down and restart during the same interval of time, in either order. It can also be started before DT or SPAN are started.

If the connection to DataTurbine is determined to be down, SpanToDT attempts to repair this first. When the connection is restored, all data gathered in the interim is inserted.

If the connection to DT is up, but the connection to SPAN is determined to be down, all threads are restarted. When the simulator is down there's of course no data to be lost here, were it really SPAN instead, SPAN buffers the data to disk.

#4 - 06/14/2012 08:45 PM - Derik Barseghian

Whoops, that last sentence doesn't make sense. If SPAN is down, it's down. Should SpanToDT be down, that's when SPAN would buffer to disk.

Did a bunch more testing against a real SPAN server, and checked in a few bug fixes at r29952.

Closing.

#5 - 03/27/2013 02:30 PM - Redmine Admin

Original Bugzilla ID was 5378