Using IBM MQ High Availability sample amqsphac to test the switchover of a multi-instance queue manager

https://www.ibm.com/support/pages/node/6985633

Date last updated: 23-Apr-2023

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Objective +++

You want to verify the behavior of IBM MQ client applications to use the Automatic Client Reconnection feature, when there is a manual graceful switchover from the Active instance to a Standby instance in a Multi-Instance queue manager.

One quick way to test the switchover of an IBM MQ multi-instance queue manager is by using the following:

- The MQSERVER environment variable and using a connectionNameList that lists the Active and Standby:

```
export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/hostname1(1414),hostname2(1414)
```

- The high availability sample "amqsphac" supplied with IBM MQ, which uses Automatic Client Reconnection to put a message into a queue.
  Note: The sample "amqsputc" does NOT uses Automatic Client Reconnection.

++ Related article

https://www.ibm.com/support/pages/node/6985635
How to specify the ConnectionNameList for an IBM MQ multi-instance queue manager in MQSERVER, CCDT, Explorer, JMS ConnectionFactory

++ Overall approach

The idea is to have the Active instance running in Host1 (host riggioni1 for this tutorial), and the Standby instance in Host2 (suvereto1).

The MQ Client application amqsphac is running in Host4 (volterra1) and uses MQSERVER which specifies the desired order for attempting the connections to the queue manager.
- The sample connects first to Host1.
- Due to the switchover, the MQ client shared libraries detect that the queue manager in Host1 is no longer working, and attempts to connect to Host2.

For more information on amqsphac see:
https://www.ibm.com/docs/en/ibm-mq/9.3?topic=multiplatforms-high-availability-sample-programs
IBM MQ / 9.3
High availability sample programs
"
The amqsghac, amqsphac, and amqsmhac high availability sample programs use automated client reconnection to demonstrate recovery following the failure of a queue manager.
... and can be used in combination to demonstrate reconnection after the failure of one instance of a multi-instance queue manager.
"
You need to install the fileset for the Samples, such as:
$ rpm -qf /opt/mqm/samp/bin/amqsphac
MQSeriesSamples-9.3.0-2.x86_64.rpm

++ Configuration

See: Chapter 2: Configuration and setup
https://www.ibm.com/support/pages/node/6985543
Configuring and using an IBM MQ 9.3 Multi-Instance in Linux

There are 4 Linux hosts used in this scenario.

Host1 (riggioni1): The Active instance is started first.
Host2 (suvereto1): Then the Standby instance is started. It will become the active one after the switchover.
Host3 (bilbao1): It is not involved in this scenario, but it is mentioned here for completeness. It has the physical file system for the queue manager data and recovery logs.
Host4 (volterra1): MQ client application (setting MQSERVER and running amqsphac)

Queue Manager Name: QMMI1 (port 1420)

++ Procedure

+ In Host1 (riggioni1), start the Active instance, then run dspmq

```
mqm@Host1: /home/mqm
$ strmqm -x QMMI1

$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                   STATUS(Running)
   INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
   INSTANCE(suvereto1.fyre.ibm.com) MODE(Standby)
      master(riggioni1.fyre.ibm.com,7225039692376204173)
      active(riggioni1.fyre.ibm.com,7225039692376204173)
```

+ In Host2 (suvereto1), start the Standby instance, then run dspmq

```
mqm@Host2: /home/mqm
$ strmqm -x QMMI1

mqm@Host2: /home/mqm
$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                   STATUS(Running as standby)
   INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
   INSTANCE(suvereto1.fyre.ibm.com) MODE(Standby)
      master(riggioni1.fyre.ibm.com,7225039692376204173)
      active(riggioni1.fyre.ibm.com,7225039692376204173)
      standby(suvereto1.fyre.ibm.com,7225039726733750287)
```

+ In Host4 (volterra1), set MQSERVER and run sample

As MQ client user, specify the following MQSERVER variable that points to both hosts (the first one is Host1, the second is Host2)

```
mqm@Host4: /home/mqm
$ export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/riggioni1(1420),suvereto1(1420)'
```

The MQSERVER uses the concept of "connectionNameList", in which the host(port) are separated by commas.
The ORDER of the items in the list is extremely important, because the MQ client library code will attempt the reconnection/new-connection strictly in the order of the list:

1) starting with the first item: riggioni1(1420)

2) if first item is not available, then will try the 2nd: suvereto1(1420)

3) if 2nd is not available, then will try the 3rd (if specified),

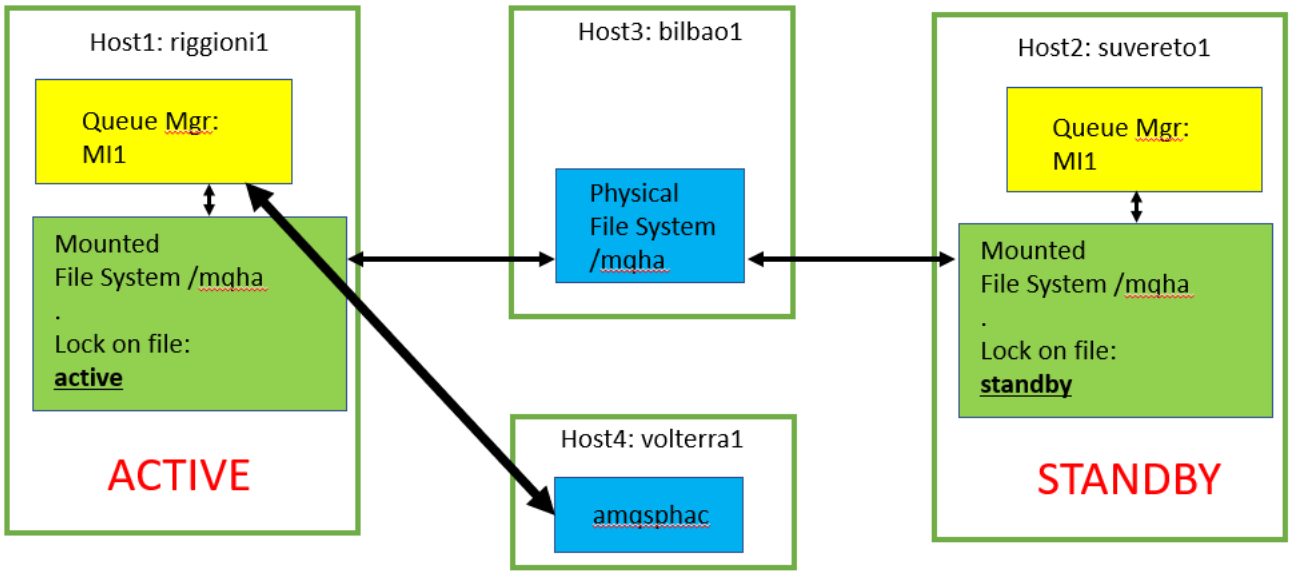
4) and so on.

Run the following MQ sample "high availability" that puts messages into a queue, and which is able to handle gracefully a switchover.

The MQ client library code attempts to connect to the FIRST item in the MQSERVER, in this case, Host1 riggioni1. Because it is able to connect, then it proceeds to put messages into Host1.

```
mqm@Host4:  /home/mqm
$ amqsphac Q1 QMMI1
Sample AMQSPHAC start
target queue is Q1
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
message <Message 6>
```

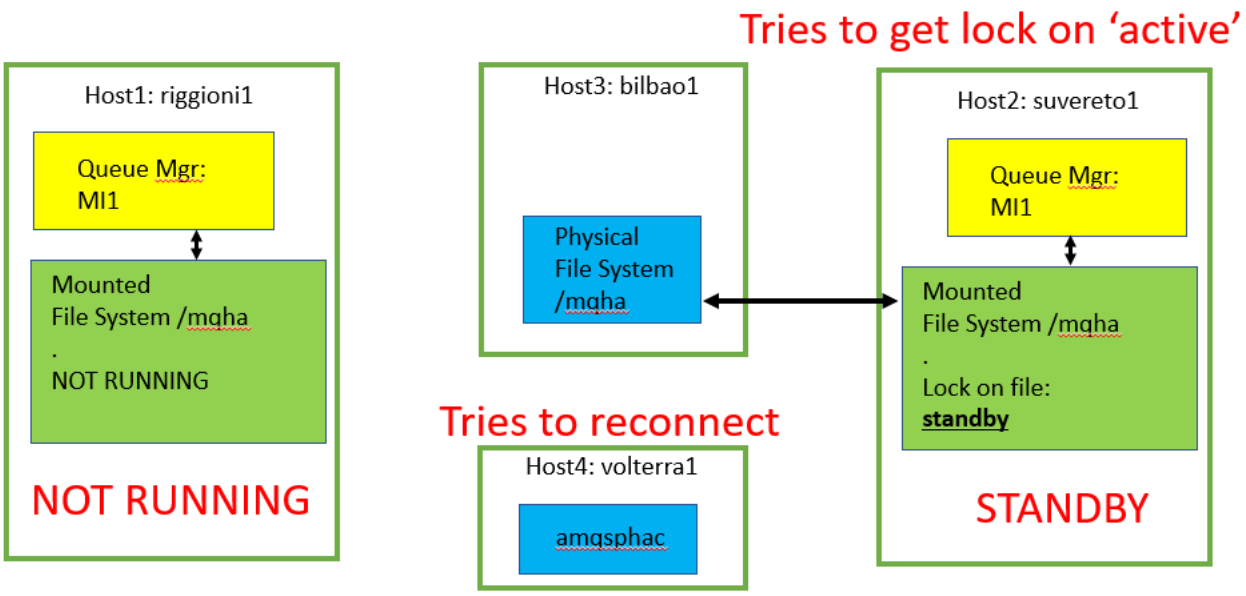
This is the picture at this moment:



+ In Host1 (riggioni1), end the queue manager and specify a switchover:

```
mqm@Host1: /home/mqm
$ endmqm -is QMMI1
Waiting for queue manager 'QMMI1' to end.
IBM MQ queue manager 'QMMI1' ending.
IBM MQ queue manager 'QMMI1' ended, permitting switchover to a standby
instance.
```

This is the picture at this moment:



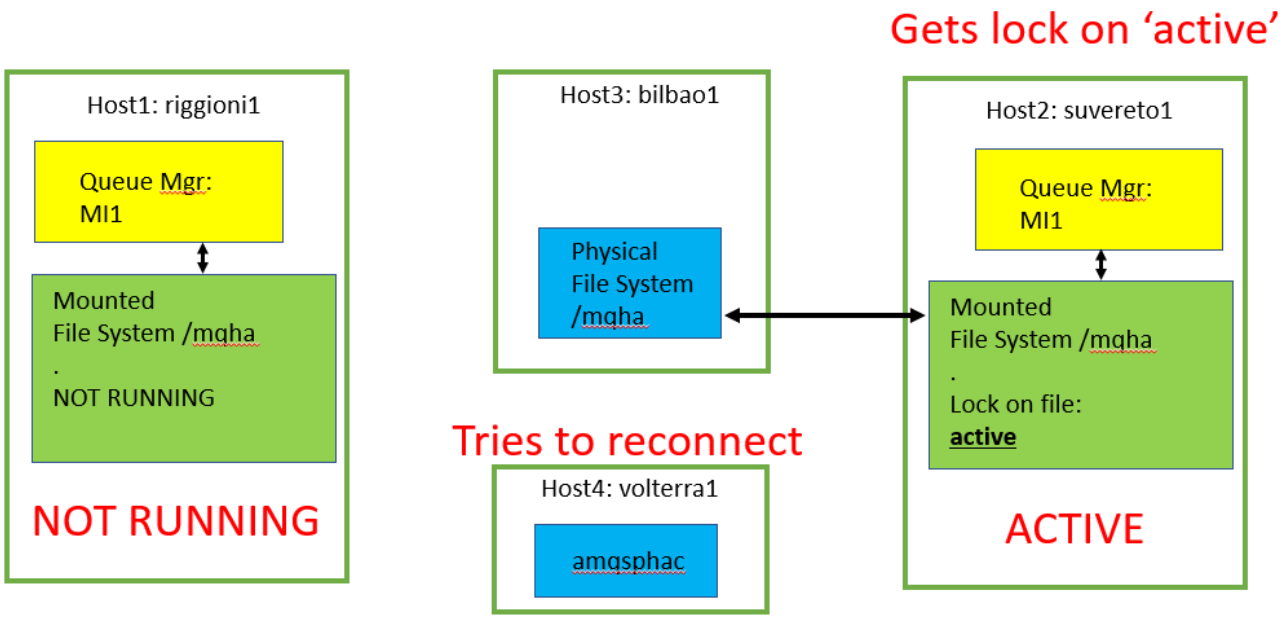
The Standby queue manager is asking every 2 seconds to the Operating System:
Can I get the lock of the file "active"?
The OS checks that the file lock for "active" is owned by host1, and denies the request to the Standby.

However, now that the Active in host1 has ended, then the file "active" is NO longer locked! Thus, at the next time that the Standby asks the OS for the file lock, the OS will assign the lock to the instance in host2 and the instance will become the new Active.

Notice that dspmq does not show riggioni1 anymore. Instead, suvereto1 is mentioned as the active:

```
mqm@Host1: /home/mqm
$  dspmq -xf -m QMMI1
QMNAME(QMMI1)                                   STATUS(Running elsewhere)
    INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
        master(suvereto1.fyre.ibm.com,7225039726733750287)
        active(suvereto1.fyre.ibm.com,7225039726733750287)
```

This is the picture now:



+ Host4: The sample detects the connectivity problem due to the ending of the active in Host1

The MQ client library code attempts to reconnect to the FIRST item in the list for MQSERVER: Host1
Because Host1 is not working, then the MQ client library code attempts to connect to the SECOND item in the list: Host2

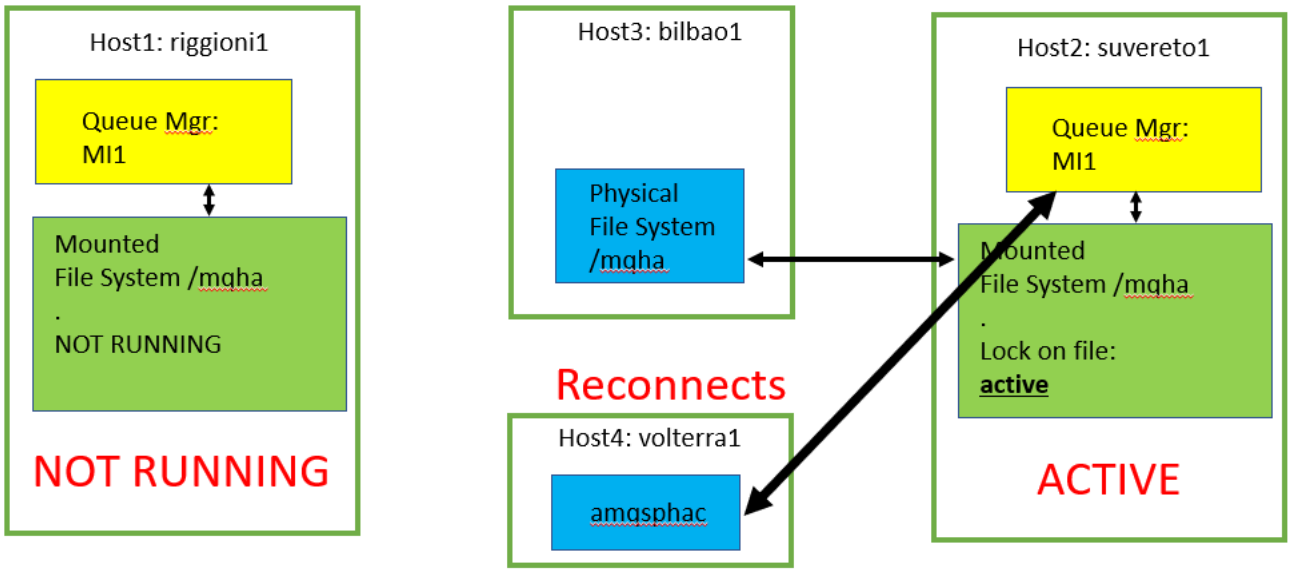
message <Message 6>
07:00:58 : EVENT : Connection Reconnecting (Reason: 2161, Delay: 34ms)
07:00:58 : EVENT : Connection Reconnecting (Reason: 2161, Delay: 1021ms)
07:00:59 : EVENT : Connection Reconnecting (Reason: 2161, Delay: 2179ms)
07:01:01 : EVENT : Connection Reconnecting (Reason: 2161, Delay: 4259ms)

+ Host2: the standby becomes the new Active
mqm@Host2: /home/mqm
$ dspmq -xf -m QMMI1
QMNAME(QMMI1) STATUS(Running)
INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
master(suvereto1.fyre.ibm.com,7225039726733750287)
active(suvereto1.fyre.ibm.com,7225039726733750287)

+ Host4:
Because Host2 has now the new Active, the MQ client library code proceeds to connect:

07:01:06 : EVENT : Connection Reconnected

This is the picture now:



+ Host4: After it is reconnected to Host2, the sample continues to put messages:

message <Message 7>
message <Message 8>
message <Message 9>

As you can see with this brief test, the MQSERVER variable and the sample amqsphac can be used to do a quick verification test of the switchover of an MQ multi-instance queue manager.

+++ end +++