Hvrfailover

Ingres Unix & Linux

Contents

Command **hvrfailover** applies only to Ingres locations.

Name

hvrfailover - Failover between Business Continuity nodes using replication

Synopsis

hvrfailover [**-r**] [**-v**] **start**

hvrfailover [**-r**] [**-v**] **stop**

hvrfailover boot

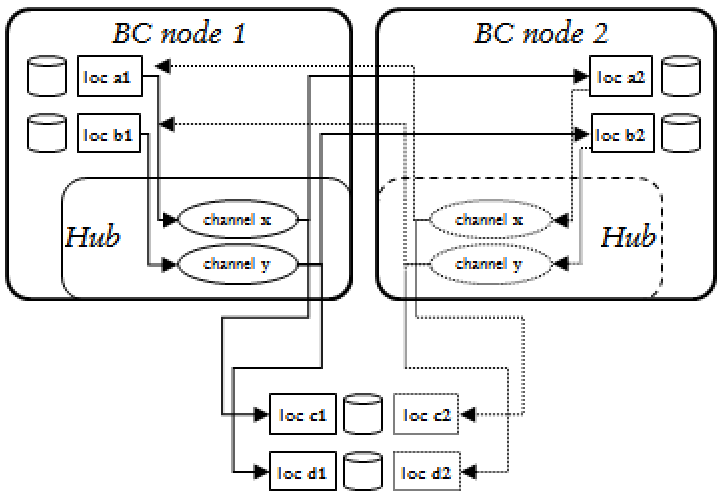
hvrfailover check

Description

hvrfailover manages the 'failover' of a database service between two nodes. These are called 'Business Continuity (BC) nodes'. They do not need to share a disk. At any moment only one of these BC nodes is active, and HVR is replicating database changes across to the other node, so that if an error occurs on the active node then **hvrfailover** can switch-over to a 'virtual IP address'. At this point existing database connections are broken (an error message will be returned immediately or they may have to timeout first) and new database connections are sent to the other node instead. This ensures 'high availability' for the database service, and means a replicated system has no 'single point of failure'. Each BC node contains one or more replicated databases plus a HVR hub with channels to replicate changes from them to the other BC node.

It is important to distinguish between 'graceful failover' and 'abrupt failover'. Graceful failover is when a proper 'stop' was performed on the first node (so that all replicated changes are flushed to the other node) before switching to the virtual IP address and starting on the other node. In this case, the first node is still consistent and a 'failback' is possible without causing consistency problems. Abrupt failover is when a proper stop was not performed (machine crashed?) or did not fully complete (network timeout?). In this case there could be unreplicated changes left in the first node (called 'phantom changes'), so a 'failback' to it is impossible because it could give database inconsistency. Resetting a database after an abrupt failover typically requires an **Hvrrefresh** and a new **Hvrinit**. The unreplicated changes are lost.

The replication inside the **hvrfailover** nodes must conform to the diagram on the right. The arrows show the direction of replication; the dotted arrows are replication after failover has occurred. The names of the hub database and channels are identical in both nodes. All locations in one BC node must end with suffix **1** and in the other with suffix **2**. Channels in one node must capture changes from one location inside that node and integrate them to one on the other node.



So for example if a channel named **mychannel** in the hub database on the first BC node capture location is **xxx1** and an integrate location **xxx2**, then channel **mychannel** also exists in other BC node with capture location **xxx2** and integrate location **xxx1**.

Channels may also integrate changes to extra databases outside either BC nodes. This can be used to add an off-site 'disaster recovery' database to the channel for a 'one-way' switch-over after a real emergency, with the two BC nodes providing bi-directional switch-over to maintain business continuity for smaller failures or during batchwork. These databases must be added with a different location name in each hub; with suffix **1** in the first node's hub and with suffix **2** in the other hub.

Command **hvrfailover start** first checks that the other BC node is not active by doing a **ping** of virtual IP address (unless option **-r** is supplied) and **Hvrtestlistener** to the other hub's scheduler (unless option **-v** is supplied). It then begins replication to the other node by starting the HVR Scheduler. Finally (if option **-r** is not supplied), it activates the virtual IP address, so new database connections will be redirected to databases on this node. Option **-r** therefore means that replication should be started, but database connections via the virtual IP address should not be affected. Option **-v** means that only the virtual IP address is started or stopped. Command **hvrfailover start** should never be done on one node while either replication or the virtual IP address is active on the other node.

Command **hvrfailover stop** breaks all connections into the local databases by stopping and starting the DBMS communication server (e.g. **ingstop/start -iigcc**, not if **-v** is supplied), and deactivating the virtual IP address (not if **-r** is supplied). It also attempts to flush replication so that both databases are identical.

Configuration Options

File **$HVR_CONFIG/files/hvrfailover.opt** contains the following options:

Parameter	Description
-env=*NAME*=*VALUE*	Environment variables. Set e.g. **$II_SYSTEM** and **$HVR_PUBLIC_PORT**.
-hubdb=*dbname*	Hub database name. Should be the same name on both BC nodes. Mandatory.
-virtual_ip=*ipaddress*	Virtual IP address.

Configuring Hvrfailover

1. Configure the DBMS (Ingres) so it is restarted at boot time on both nodes.
2. Configure the **hvrfailover.opt** file in **$HVR_CONFIG/files**. This file should be identical on both BC nodes.
   Example:

```
-hubdb=hvrhub
-virtual_ip=192.168.10.228
-env=II_SYSTEM=/opt/Ingres/IngresII
-env=HVR_PUBLIC_PORT=50010
```

3. Create a new interface for the virtual IP address on both BC nodes. This can be done, as **root** or by copying an interface file into directory **/etc/sysconfig/network-scripts**.
   Example:

   ```
   $ cp ifcfg-eth0 ifcfg-eth0:1
   ```

   Then edit this file and change the directives:

   ```
   DEVICE=eth0:1                        #(for this example)
   IPADDR=<virtual ip>
   ONBOOT=no
   USERCTL=yes
   ```

4. Configure **Hvrremotelistener** and **Hvrfailover** so they are restarted at boot time by adding a line to **/etc/hvrtab** on both nodes.
   Example:

   ```
   root 4343 /opt/hvr/hvr_home /opt/hvr/hvr_config -EHVR_TMP=/tmp
   ingres hvrfailover /opt/hvr/hvr_home /opt/hvr/hvr_config -EHVR_TMP=/tmp
   ```

5. Configure **Hvrmaint** to run frequently, for example by adding a line to **crontab**.
6. Optionally a crontab may be added to run **Hvrlogrelease** on both BC nodes. **Hvrlogrelease** option (**-logrelease_expire**) should be used to ignore logrelease files older than a certain period. This is necessary to ensure **Hvrlogrelease** cleans up redo/journal files on the BC nodes.

Files

HVR_HOME	
bin	
hvrfailover	Perl script.
HVR_CONFIG	
files	
hvrfailover.opt	Option file.
hvrfailover.state	File created automatically by **hvrfailover**. It contains string **START** or **STC**
hvrfailover.stop_done	File created automatically by **hvrfailover**.
log	
hvrfailover	
hvrfailover.log	Log file.