Friday, January 22. 2010

Less known Solaris features - IP Multipathing (Part 10): Tips, Tricks and other comments

At the end of this tutorial i want to talk about some tricks, tips and comments in regard of new and classic IPMP. Reducing the address sprawl of probe based failure detectionThe probe based failure detection isn't without a disadvantage. You need a test IP adresses for every interface in an IPMP group. But: The IP addresses doesn't have to be routable ones, they doesn't even have to be in the same subnet as the data address. It's perfectly possible to use a routable IP address as the data address and to use for example the 192.168.1.0/24 address just for the IPMP test addresses . You have just to take care of one additional configuration item: You have to provide test targets in the same network. This can be done by configuring an an additional IP address on your router interfaces for example.

Explicitly configuring target systemsConfiguring explicit target systems is easy. You just have to configure host routes to them. Thus route add -host 192.168.178.2 192.168.178.2 -static would force in.mpathd to use this system as a target system.

A transient service to configure those target systemsThose routes aren't boot persistent. So you need a way to ensure that those routes are configured at boot up. You could use a simple legacy init.d script, but I want to implement it as a configurable transient SMF service. A transient service is a service that is executed once at the start and isn't monitored by the restarter daemon. A transient service is a useful mechanism to do such initial configuration tasks at system start.

The service consists out of a simple manifest. Store it in a file ipmptargets.xml:

system-wide configuration of IP routes for IPMP

The specific host routes are implemented as instances of this service. So it is possible to control the routes with a fine granularity.

Okay, obviously we need the script mentioned in the exec methods of the manifest. So put the following script into the the file /lib/svc/method/ipmptargets:

```
#!/bin/sh

. /lib/svc/share/smf_include.sh

getproparg() {
val=`svcprop -p $1 $SMF_FMRI`
[ -n "$val" ] && echo $val
}

if [ -z "$SMF_FMRI" ]; then
echo "SMF framework variables are not initialized."
exit $SMF_EXIT_ERR_CONFIG
fi

OPENVPNBIN='/usr/sbin/route'
IP=`getproparg config_params/ip`


if [ -z "$IP" ]; then
echo "config_params/ip property not set"
exit $SMF_EXIT_ERR_CONFIG
fi

case "$1" in
'start')
route add -host $IP $IP -static
;;

'stop')
echo "not implemented"
route delete -host $IP $IP -static
;;

'refresh')
 route delete -host $IP $IP -static
 route add -host $IP $IP -static
;;

*)
echo $"Usage: $0 {start|refresh}"
exit 1
;;
```

```
esac
exit $SMF_EXIT_OK
```
Okay. Now we have to import the the SMF manifest into the repository.
```
jmoekamp@hivemind:~# svccfg import  ipmp_hostroutes.xml
```
It's ready to use now. You can enable and disable your IPMP host routes as you need them:
```
jmoekamp@hivemind:~# svcadm enable ipmptargets:target1
jmoekamp@hivemind:~# netstat -nr

Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              192.168.178.1        UG       23     496909
127.0.0.1            127.0.0.1            UH        2       2796 lo0
192.168.56.0         192.168.56.1         U         2          0 vboxnet0
192.168.178.0        192.168.178.9        U         3          0 production0
192.168.178.1        192.168.178.1        UGH       1          0

Routing Table: IPv6
  Destination/Mask            Gateway                 Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2      20 lo0
jmoekamp@hivemind:~# svcadm disable ipmptargets:target1
jmoekamp@hivemind:~# netstat -nr

Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              192.168.178.1        UG       24     496941
127.0.0.1            127.0.0.1            UH        2       2796 lo0
192.168.56.0         192.168.56.1         U         2          0 vboxnet0
192.168.178.0        192.168.178.9        U         3          0 production0

Routing Table: IPv6
  Destination/Mask            Gateway                 Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2      20 lo0
```

Migration of the classic IPMP configurationAlbeit new IPMP should be configured like described above a correct configuration for the classic IPMP will setup the new IPMP correctly as well.

Setting a shorter or longer Failure detection timeWhen i talked about probe based failure detection, i've told you that you have to wait a certain time until a failure or a repair is detected by the \verb=in.mpathd=. The default is 10 seconds for failure detection. The time for repair detection is always twice the time of the failure detection, so it's 20 seconds per default. But sometimes you want a faster reaction to failures and repairs. You can control the failure detection time by editing /etc/default/mpathd. You will find a configuration item controlling this time span in the file:
```
#
# Time taken by mpathd to detect a NIC failure in ms. The minimum time
# that can be specified is 100 ms.
#
FAILURE_DETECTION_TIME=10000
```
By using a smaller number, you can speed up the failure detection but you have a much higher load of ICMP probes on you system. Keep in mind that i've told you that if 5 consecutive probes fail, the interfaces is considered as failed. When the failure detection time is 10000 ms, the probes have to be sent every 2000 ms. When you configure 100 ms, you will see a probe every 20ms. Furthermore this probing is done on every interface. Thus at 100ms failure detection time, the targets will see 3 ping requests every 20 milliseconds.

So keep in mind that lowering this number will increase load on other systems. So choose your the failure detection based on your business and application needs, not on the thought "I want the lowest possible time".

Just to demonstrate this effect and to learn how you set the failure detection time, you should modify the value in the line

FAILURE_DETECTION_TIME to 100. Restart the in.mpathd afterwards by sending a HUP signal to it with pkill -HUP in.mpathd. When you start a snoop with snoop -d e1000g0 -t a -r icmp you will have an output on your display scrolling at a very high speed.

Posted by Joerg Moellenkamp in English, Oracle, Solaris at 13:31