NAME

gmultipath - disk multipath control utility

SYNOPSIS

gmultipath create [**-ARv**] *name prov ...*
gmultipath label [**-ARv**] *name prov ...*
gmultipath configure [**-APRv**] *name*
gmultipath add [**-v**] *name prov*
gmultipath remove [**-v**] *name prov*
gmultipath fail [**-v**] *name prov*
gmultipath restore [**-v**] *name prov*
gmultipath rotate [**-v**] *name*
gmultipath prefer [**-v**] *name prov*
gmultipath getactive [**-v**] *name*
gmultipath destroy [**-v**] *name*
gmultipath stop [**-v**] *name*
gmultipath clear [**-v**] *prov ...*
gmultipath list
gmultipath status
gmultipath load
gmultipath unload

DESCRIPTION

The **gmultipath** utility is used for device multipath configuration.

The multipath device can be configured using two different methods: "manual" or "automatic". When using the "manual" method, no metadata are stored on the devices, so the multipath device has to be configured by hand every time it is needed. Additional device paths also will not be detected automatically. The "automatic" method uses on-disk metadata to detect device and all its paths. Metadata use the last sector of the underlying disk device and include device name and UUID. The UUID guarantees uniqueness in a shared storage environment but is in general too cumbersome to use. The name is what is exported via the device interface.

The first argument to **gmultipath** indicates an action to be performed:

create Create multipath device with "manual" method without writing any on-disk metadata. It is up to administrator, how to properly identify device paths. Kernel will only check that all given providers have same media and sector sizes.

-A option enables Active/Active mode, **-R** option enables Active/Read mode, otherwise

Active/Passive mode is used by default.

label Create multipath device with "automatic" method. Label the first given provider with on-disk metadata using the specified *name*. The rest of given providers will be retasted to detect these metadata. It reliably protects against specifying unrelated providers. Providers with no matching metadata detected will not be added to the device.

-A option enables Active/Active mode, **-R** option enables Active/Read mode, otherwise Active/Passive mode is used by default.

configure

Configure the given multipath device.

-A option enables Active/Active mode, **-P** option enables Active/Passive mode, **-R** option enables Active/Read mode.

add Add the given provider as a path to the given multipath device. Should normally be used only for devices created with "manual" method, unless you know what you are doing (you are sure that it is another device path, but tasting its metadata in regular "automatic" way is not possible).

remove

Remove the given provider as a path from the given multipath device. If the last path removed, the multipath device will be destroyed.

fail Mark specified provider as a path of the specified multipath device as failed. If there are other paths present, new requests will be forwarded there.

restore Mark specified provider as a path of the specified multipath device as operational, allowing it to handle requests.

rotate Change the active provider/path to the next available provider in Active/Passive mode.

prefer Change the active provider/path to the specified provider in Active/Passive mode.

getactive

Get the currently active provider(s)/path(s).

destroy Destroy the given multipath device clearing metadata.

stop Stop the given multipath device without clearing metadata.

clear Clear metadata on the given provider.

list See geom(8).

status See geom(8).

load See geom(8).

unload See geom(8).

SYSCTL VARIABLES

The following sysctl(8) variable can be used to control the behavior of the **MULTIPATH** GEOM class.

kern.geom.multipath.debug: 0

Debug level of the **MULTIPATH** GEOM class. This can be set to 0 (default) or 1 to disable or enable various forms of chattiness.

kern.geom.multipath.exclusive: 1

Open underlying providers exclusively, preventing individual paths access.

EXIT STATUS

Exit status is 0 on success, and 1 if the command fails.

MULTIPATH ARCHITECTURE

This is a multiple path architecture with no device knowledge or presumptions other than size matching built in. Therefore the user must exercise some care in selecting providers that do indeed represent multiple paths to the same underlying disk device. The reason for this is that there are several criteria across multiple underlying transport types that can *indicate* identity, but in all respects such identity can rarely be considered *definitive*.

For example, if you use the World Wide Port Name of a Fibre Channel disk object you might believe that two disks that have the same WWPN on different paths (or even disjoint fabrics) might be considered the same disk. Nearly always this would be a safe assumption, until you realize that a WWPN, like an Ethernet MAC address, is a soft programmable entity, and that a misconfigured Director Class switch could lead you to believe incorrectly that you have found multiple paths to the same device. This is an extreme and theoretical case, but it is possible enough to indicate that the policy for deciding which of multiple pathnames refer to the same device should be left to the system operator who will use tools and knowledge of their own storage subsystem to make the correct configuration selection.

There are Active/Passive, Active/Read and Active/Active operation modes supported. In Active/Passive mode only one path has I/O moving on it at any point in time. This I/O continues until an I/O is returned with a generic I/O error or a "Nonexistent Device" error. When this occurs, that path is marked FAIL, the next path in a list is selected as active and the failed I/O reissued. In Active/Active mode all paths not marked FAIL may handle I/O at the same time. Requests are distributed between paths to equalize load. For capable devices it allows the utilisation of the bandwidth available on all paths. In Active/Read mode all paths not marked FAIL may handle reads at the same time, but unlike in Active/Active mode only one path handles write requests at any point in time; closely following the original write request order if the layer above needs it for data consistency (not waiting for requisite write completion before sending dependent write).

When new devices are added to the system the **MULTIPATH** GEOM class is given an opportunity to taste these new devices. If a new device has a **MULTIPATH** on-disk metadata label, the device is either used to create a new **MULTIPATH** GEOM, or added to the list of paths for an existing **MULTIPATH** GEOM.

It is this mechanism that works reasonably with isp(4) and mpt(4) based Fibre Channel disk devices. For these devices, when a device disappears (due to e.g., a cable pull or power failure to a switch), the device is proactively marked as gone and I/O to it failed. This causes the **MULTIPATH** failure event just described.

When Fibre Channel events inform either isp(4) or mpt(4) host bus adapters that new devices may have arrived (e.g., the arrival of an RSCN event from the Fabric Domain Controller), they can cause a rescan to occur and cause the attachment and configuration of any (now) new devices to occur, causing the taste event described above.

This means that this multipath architecture is not a one-shot path failover, but can be considered to be steady state as long as failed paths are repaired (automatically or otherwise).

Automatic rescanning is not a requirement. Nor is Fibre Channel. The same failover mechanisms work equally well for traditional "Parallel" SCSI but may require manual intervention with camcontrol(8) to cause the reattachment of repaired device links.

EXAMPLES

The following example shows how to use camcontrol(8) to find possible multiple path devices and to create a **MULTIPATH** GEOM class for them.

```
mysys# camcontrol devlist
<ECNCTX @WESTVILLE >   at scbus0 target 0 lun 0 (da0,pass0)
<ECNCTX @WESTVILLE >   at scbus0 target 0 lun 1 (da1,pass1)
```

```
<ECNCTX @WESTVILLE >   at scbus1 target 0 lun 0 (da2,pass2)
<ECNCTX @WESTVILLE >   at scbus1 target 0 lun 1 (da3,pass3)
mysys# camcontrol inquiry da0 -S
ECNTX0LUN000000SER10ac0d01
mysys# camcontrol inquiry da2 -S
ECNTX0LUN000000SER10ac0d01
```

Now that you have used the Serial Number to compare two disk paths it is not entirely unreasonable to conclude that these are multiple paths to the same device. However, only the user who is familiar with their storage is qualified to make this judgement.

You can then use the **gmultipath** command to label and create a **MULTIPATH** GEOM provider named *FRED*.

```
gmultipath label -v FRED /dev/da0 /dev/da2
disklabel -Bw /dev/multipath/FRED auto
newfs /dev/multipath/FREDa
mount /dev/multipath/FREDa /mnt....
```

The resultant console output looks something like:

```
GEOM_MULTIPATH: da0 added to FRED
GEOM_MULTIPATH: da0 is now active path in FRED
GEOM_MULTIPATH: da2 added to FRED
```

To load the **gmultipath** module at boot time, add this entry to */boot/loader.conf*:

```
geom_multipath_load="YES"
```

SEE ALSO

geom(4), isp(4), mpt(4), loader.conf(5), camcontrol(8), geom(8), mount(8), newfs(8), sysctl(8)

HISTORY

The **gmultipath** utility first appeared in FreeBSD 7.0

AUTHORS

Matthew Jacob <*mjacob@FreeBSD.org*>
Alexander Motin <*mav@FreeBSD.org*>