Exporting a SNMP tree with GANESHA

1 Overview

Thanks to its backend modules called "File System Abstraction Layers" (FSAL), GANESHA NFS server makes it possible to export any sets of data organized as trees, where each entry has a name and path.

In SNMP (Simple Network Management Protocol), data are organized as a tree where all objects are addressed by their OID, which is the object's path in this tree. For example, ".iso" can be considered has a directory identified by the OID ".1", and the leaf ".iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0" (which corresponds to system description) can be considered as a file whose data is something like "Linux node_name 2.6.9-22.ELsmp #1 SMP Mon Sep 19 18:32:14 EDT 2005 i686".

As a result, exporting such a tree through NFS makes it possible to have a filesystem similar to "/proc", where administrators can read statistics about a system or an equipment (switch, router, ...) simply using `cat` command, and modify them easily, using `vi` or `echo "xxx" > file`.

This is what you can do using the SNMP FSAL. To use it, simply build GANESHA using the configure arg `-with-fsal=SNMP`:

```
cd src
./configure --with-fsal=SNMP
make
```

Net-snmp library and includes must be installed on your system.

2 SNMP relative options

2.1 The SNMP block

2.1.1 Parameters description

For configurating GANESHA's SNMP access, you have to set some options in the configuration file: this in done in a "`SNMP`" configuration block.

In such a block, you can set the following values:

- snmp_version: this indicates the SNMP protocol version that GANESHA will use for communicating with SNMP agent. Expected values are `1`, `2c` or `3`. Default is `"2c"`.
- snmp_server: this is the address of the SNMP master agent. A port number can also be specified by adding ":<port>" after the address. Default is `"localhost:161"`.

- nb_retries: number of retries for SNMP requests. Default value is `SNMP_DEFAULT_RETRIES`, defined in net-snmp library.
- microsec_timeout: number of microseconds until first timeout, then an exponential backoff algorithm is used for next timeouts. Default value is `SNMP_DEFAULT_TIMEOUT`, defined in net-snmp library.
- client_name: this is the client name that could be used internally by net-snmp for its traces. Default value is `"GANESHA"`.
- snmp_getbulk_count: this indicates the number of responses wanted for each SNMP GETBULK request. Default value is `64`.

SNMP v1 and v2 specific parameters:

- community: this is the SNMP community used for authentication. Default is `"public"`.

SNMP v3 specific parameters:

- auth_proto: the authentication protocol (MD5 or SHA). Default is `"MD5"`.
- enc_proto: the privacy protocol (DES or AES). Default is `"DES"`.
- username: the security name (or user name). This a private information: check rights on this config file!
- auth_phrase: authentifaction passphrase (>=8 char). This a private information: check rights on this config file!
- enc_phrase: authentifaction passphrase (>=8 char). This a private information: check rights on this config file!

2.1.2 A simple SNMP v2c example

```
SNMP
{
    snmp_version = 2c;
    snmp_server = "snmp_master.my_net";
    community = "public";
}
```

2.1.3 A simple SNMP v3 example

```
SNMP
{
    snmp_version = 3;
    snmp_server  = "snmp_master.my_net";
    username     = "snmpadm";
    auth_phrase  = "p4ssw0rd!";
    enc_phrase   = "p455w0rd?";
}
```

2.2 Export entries

For defining the path of an export entry, you must replace traditional SNMP dot separators '.' by slashes '/'. For example, you should set export path to '/iso/org' instead of '.iso.org'.

Note that you can give slash separated numerical OIDs for exports. Thus, exporting '/1/3/6/1/2/1' is equivalent to '/iso/org/dod/internet/mgmt/mib-2'.

3 Specific behaviors

Even if it is possible to export a SNMP tree as if it was a standard filesystem, SNMP however has some specificities that don't match POSIX or NFS semantics and features.

3.1 Creating and removing objects

The goal of exporting SNMP with NFS is to make it possible for an administrator to browse statistics, to read them, and modify some agent's configuration variables.

Thus, we don't have any interest in creating new entries (register some new values to SNMP agent) nor removing existing entries (unregister agent's objects). That would be very complex to operate and it could result in agent's problems or crashes...

Another problem in SNMP is that object types are very different from NFS types: an object can be an integer, a string, a counter, a timetick, so it is difficult to know the type of data a NFS client is going to write to a newly created file...

What's more, directories don't have a proper existence; they only exist if a child of them exist. As a result, creating or deleting an empty directory would have no sense in SNMP.

For all those reasons, **create and remove operations** (create, mkdir, remove, rmdir) **are not supported** and return an `EROFS` error.

3.2 Objects' attributes

3.2.1 Returned attributes

SNMP objects don't have the same metadata as NFS. They have no modification or access time, no owner, no inode number, etc... As a result, it has been necessary to emulate some of these attributes, to make the SNMP FSAL compatible with the NFS protocol.

- type: all SNMP objects are considered as regular files and their parent nodes as directories.
- size: unlike '`/proc`' file system (where are file sizes are shown as null), the returned size for a SNMP object is the size of its current string representation.

- inode number: SNMP objects have no unic 32 or 64 bits identifier. However, their OID is unic and constant, like inode number might be. So, the returned inode number is a hash of this OID.

- owner and group: there is no such attributes in SNMP. Thus, owners and groups are set to 0 (root).

- mode: it is not always possible to determinate the access rights for an SNMP object. Thus, the SNMP FSAL makes its best to determine access rights from MIBs info, but sometimes it doesn't know, so it returns a default mode `rw-rw-rw` (0666) for files, and `r-xr-xr-x` (0555) for directories. As a result, you may get an `EACCES` error when writing or reading a file, even if you seemed to have the correct rights.

- numlinks: there is no hardlinks in SNMP, so link count is always 1 for regular files. What's more, for a given directory, il would be costful to count the number of child directories that would have a virtual '..' entry. As a result, the numlink is also set to 1 for directories.

- access, modification and change time: SNMP has no memory of the time when a variable has been created, read or modified. So, the SNMP FSAL always return current time for those attributes. What's more, given that objects' data can change continuously, this behavior forces clients to clear their datacaches.

3.2.2 Setting attributes

Given that the only attributes that have an equivalence in SNMP are static (read-only), **no attributes are supported for setattr operations**. Thus, trying to change an attribute will result in a `EINVAL` error.

3.3 Objects' data

3.3.1 String representation

SNMP objects divide in many types: integer, counter, gauge, string, opaque buffer, OID, timeticks, network address, etc. In order to make it easy to interpret, the data returned by a 'read' operation is a string representation that depends on the object type.

Indeed, even if integer, timeticks or IPv4 address are both 32 bits values, the integer will be displayed as a numerical value, timeticks will be displayed as days/hours/minutes/seconds, and the network address will be displayed in the classical dot-separated notation.

Some examples:

```
$ cd /mnt/iso/org/dod/internet/mgmt/mib-2
$ cat host/hrSystem/hrSystemUptime/0
528224338 (61 days, 03:17:23.38)

$ cat udp/udpInDatagrams/0
820798

$ cat udp/udpTable/udpEntry/udpLocalAddress/1/0/0/127/123
127.0.0.1
```

3.3.2 Modifying data

To modify a value, you just have to write a string repesentation to the associated file, using the way you want: a program, a script, or just the `echo` command, ...

For example:

```
$ cd /mnt/iso/org/dod/internet/mgmt/mib-2
$ echo "my_new_hostname" > system/sysName/0
```

But be careful: in SNMP, there are strong type constraints: you cannot write a string to an integer, etc. What's more, for a given object, the agent may also do extra checks (for example, if a configuration value is supposed to be a percentage, its SNMP type is unsigned integer, but the agent may return an error if you set a value greater than 100).

In general, you have to rewrite data in the format of the value read in the file. The only exception is for timeticks: you must only write the value in 100th of seconds (not its translation to days, hours, minutes, seconds).

An issue when developping the SNMP FSAL has been to translate "invalid type" SNMP errors to an appropriate NFS error code. Indeed, in NFS, there is no concept of invalid data content ! Returning NFSERR_INVAL or NFSERR_IO may result in a bad interpretation by the client, and it would be difficult for the client to distinguish the error from a real EIO or EINVAL error. So, it has been decided to convert such an error to NFSERR_DQUOT (that has no other sense with SNMP) so the user can know, with no ambiguity, that the value is not in the correct format, or is out of expected range.

3.3.3 Unsupported data operations

Read at offset>0 In the SNMP protocol, an object's data is read in a single GET request. As a result, a 'read' NFS operation must always read data from offset 0, and with a buffer big enough for writting the whole variable content. If the buffer is too small, the data representation will be truncated.

Thus, after a seek, a `read` operation may return an EINVAL error. The same error is returned for `pread` operation with a non-null offset.

Write at offset>0 To write in a file, it is the same: in the SNMP protocol, object's data is written in a single SET request. As a result, the NFS 'write' operation must give all the object data, starting at offset 0.

Thus, after a `seek` operation, a `write` operation may return EINVAL. The same error is returned for a `pwrite` operation with a non-null offset.

Truncate A truncate operation on an object that is not a string has no sense. For example, what about truncating an integer? Indeed, its string representation always contains at least 3 characters, e.g. for representing zero: `"0\n\0"`.

However, we must not return an error for truncate operations, because when user do something like '`echo "0" > an_integer_object`', the first system operation is:

```
open( "an_integer_object", O_CREAT|O_TRUNC|O_WRONLY );
```

which will be translated to the NFS request `SETATTR(size=0)` (i.e. truncate). As a result, if we returned an error for the truncate operation, the user could not write data to files...

Thus, we decided to ignore 'truncate' operations, given that the object's data is totaly replaced at the following 'write'.

The only border side effect is that '`cp /dev/null`' or '`truncate`' on a file has no effect (to truncate a string, you must write an empty string to it).