Using Bacula with VM-managed Tape Drives

Adam Thornton, Sine Nomine Associates
24 January 2005

Overview

Sine Nomine Associates has created a product that allows the use of the Open Source Bacula server running on Linux on zSeries under VM with 3480/3490 tape drives managed from VM. It is believed that this will work with 3590 drives using the IBM OCO drivers, but due to a lack of 3590 hardware, we have not tested this functionality.

The product consists of three major parts: a replacement for Bacula's mtx-changer script, a VM network server to handle Bacula connection requests, and a VM backend tape-handling program to actually manage the tape.

Linux

The Linux requirements are fairly simple: the Linux guest must have a tape drive defined to it, and must have TCP/IP connectivity. It must have Perl installed, and additionally must have the CPAN module Net::Telnet installed. It has only been tested on 2.4 kernels and may need some modification to run under 2.6. The tape drive must be attached to the machine when the mtx-changer.pl script runs. Thus, it becomes the responsibility of the back-end tape management functionality to make sure that a tape drive is assigned to the Linux guest before control is returned to it at the completion of a tape handling request.

There is a single file required on the Linux side, mtx-changer.pl. This is a drop-in replacement for mtx-changer as described in "Bacula Autochanger Interface" on page 259 of the Bacula 1.36 User's Guide.

In mtx-changer.pl, the arguments are as follows: the first argument, which is the changer device, should be the address of the VM Network Server, specified as *host:port*. The default port is 50200. The second argument is the command for the changer, and this is passed through unchanged. Supported commands are *load*, *loaded*, *list*, *unload*, and *slots*. The third argument is the slot: this is expected to be the tape label as it is known to VM. Specify this in ASCII: the wire protocol is ASCII and the VM Network Server converts data on its end. The fourth argument is the tape device (e.g. */dev/ntibm0*). This will be converted to a device address before being transmitted. The fifth argument is the drive index within the changer. At the current time it is unused and should be zero.

The mtx-changer.pl converts its tape device string to a device address and reads /proc/sysinfo to determine its own guest userid. It constructs a string of *userid:devaddr* from this information. Presuming it is able to do this, it opens a connection to the host:port combination given in its first argument, and prints a space-separated string consisting of the command, the slot (i.e. tape label), the device address string, and the tape index, terminated by CRLF, to the network socket.

It then reads all waiting lines from the socket. The first line read will either be "0" or a nonzero integer. This is the return code. If it is nonzero, all further lines are ignored and mtx-changer.pl exits with the return code it read. If it is zero, then each additional line is read from the console and printed to the standard output. This is simply an implementation of the expected behavior of the standard Bacula autochanger script.

VM Network Server

The VM Network Server is a simple Rexx socket server named BACULATM EXEC. It receives the command from the Linux side's mtx-changer.pl, and dispatches the request to a back-end executable. Sine Nomine Associates currently supplies a single back-end: a back end that manages a manually-loaded 3480/3490 cartridge tape.

The back-end EXEC is always named *XXX*BACIF EXEC, where *XXX* is replaced by a three-letter code specifying the back end. Sine Nomine Associates provides RAWBACIF EXEC, which implements type RAW, which is an operator-driven, manual tape drive. When invoking the server, the prefix code can be specified as an argument to the EXEC invocation. The option "(DEBUG" can also be specified in order to emit verbose debugging information from the server. For instance, to invoke BACULATM with a (hypothetical) HSM back end, the appropriate command would be:

```
BACULATM HSM
```

The BACIF EXEC is always called with the line received from the network as its argument. Thus, if Bacula, running in virtual machine BACULA, with a 3480 cartridge drive at device address 0181 accessed as /dev/ntibm0, had invoked "mtx-changer.pl vmhost:50200 load TAPE1 /dev/ntibm0 0" then the VM Network Server would be contacted on vmhost:50200, and would receive the line:

load TAPE1 BACULA:0181 0

This string would be in ASCII and terminated with CRLF. The network server would strip the CRLF from the end, convert the argument to EBCDIC, and then call the appropriate BACIF EXEC program (RAWBACIF EXEC, by default):

```
EXEC RAWBACIF load TAPE1 BACULA1:0181 0
```

The way the Network Server receives its information from the back end bears mention, as it is highly unusual in the VM world, although very similar to mtx-changer's interface: the *XXX*BACIF EXEC program must print its return status (zero for success, and nonzero for failure) as the first line of its output (to the console); any additional results are printed as successive lines. BACULATM EXEC simply reads the results into a stem variable and then prints lines from the stem to the waiting socket on the Linux side. When it has finished, it closes the socket.

VM Tape-Handling Back Ends

Each back end is expected to take, as an argument, a Bacula command in the form *command label guest:devaddr index*. It will then uppercase each of the parameters passed to it, execute the Bacula command, and print its results (either "0" followed by a set of lines to return, or a nonzero integer, which may be followed by lines useful for debugging purposes, but which are ignored by the other components of the system) to the console. For example, failure to load the requested tape because it is not present in the catalogue should trigger a zero return code from the program itself, although it will *print* a nonzero return code for BACULATM's consumption (and delivery back to the Linux side).

Each back end maintains a catalogue of tapes in a manner appropriate to the back end. These are tapes that are reserved for Bacula's use. The intention is that this be integrated with the system tape catalogue.

RAWBACIF EXEC

RAWBACIF EXEC is intended to serve as a model for other, more sophisticated, tape-handling backends for the Bacula VM Tape Server. Because RAWBACIF needs to be able to take and give tape drives to unprivileged users (such as the Bacula server) it requires privilege class B.

The back end tape catalogue in RAWBACIF EXEC is simply the file CHANGER LIST, which must reside on the minidisk or SFS directory accessed as A by the VM guest running BACULATM and RAWBACIF. The format of CHANGER LIST is very simple: blank lines are ignored, comments start with an asterisk followed by a space, and other than that, it's one tape label per line.

RAWBACIF takes requests, and sends a message to the operator asking him to perform the manipulation requested; it's got a nonconfigurable (well, configurable in the source code) delay timeout and repetition schedule; if at the end of its timeouts it doesn't have the tape in the correct state, it returns failure; otherwise, it attaches the tape to the appropriate Linux image and returns success. Remember that "return" here means "emit a line containing the return code to the console, followed by result lines for a zero return code, or optional diagnostic information for a nonzero return code."

Conclusion

The only component third-party vendors need to implement to enable interoperability is the *XXX*BACIF EXEC piece. This piece must maintain a tape catalogue and respond appropriately to the following commands: *LOAD*, *LOADED*, *LIST*, *UNLOAD*, and *SLOTS*. It must be able to mediate attaching and detaching the tape drive to and from the Linux guest while loading or unloading the appropriately-labeled tapes; specifically, before it exits, it must have reattached the tape drive to the Linux guest.