Dynamically Mount Xen Block Devices

Xen is an excellent virtualization tool, but dynamically attaching/detaching can be confusing. Most of the articles I have read require you to use fairly confusing programs that require research for each volume you want to work with.

Xen provides the utility to attach/detach block devices (and others) to/from a running DOMU. Attaching is easy:

```
xl block-attach DOMU DeviceOnDOM0 NameOnDOMU w
```

example:

```
xl block-attach book phy:/dev/sdd1 xvda3 w
```

meaning attach /dev/sdd1 (on DOM0) to virtual “book” as xvda3, with write access. You could then log into the DOMU (book) and issue the command mount /dev/xvda3 /mnt and access it as normal.

The problem comes in that, during the attach, the block device is given a number which all subsequent attempts to detach must know. There have been many articles describing how to do this, the simplest of which is to monitor the xen logs on DOM0 to see what it is called when attached. It is usually a 5 digit number on my machines, and it is sequential (just like the dom-ids), so it is not constant.

The command to detach has the form:

```
xl block-detach dom-id vbd-id
```

where dom-id is the name or id of the runnign DOMU, and vbd-id is the id of the attached device. In the past, I have just given up and shut down the whole DOMU, done my job, then brought it back up (sorta' like rebooting a Windows machine when nothing else works).

I ran into a situation recently where I needed to unmount one partition out of 11 on a domU (I needed to add more space). I could easily tell the client I was setting up to stay off the server for the time it would take, but contacting the other nine clients who also use the machine would have taken time I did not want to spare, and require downtime for everyone using the server, not just the client receiving and upgrade.

Read some interesting articles on xenstore tools, and after some investigation, found it to be the most confusing way to do anything, but definitely worth the trouble as it was very amenable to programming. Thus, the attached script.

This script reads the xenstore, finding all domains and all vbd's (Virtual Block Device's) attached to them, then prints them out as a tab delimited list, good for parsing with grep and/or cut (or AWK).

One line of output might look like this:

```
book    26    51715    xvda3    /dev/virtuals/book.dailydata.net-home
```

The first column is the name of the DomU, and the second is the domid (a number given to each domain when it is started). Third number (51715) is the vbd number (again, assigned by the DOM0 on every attachment), the fourth is the name of the device on the DOMU (ie, /dev/xvda4) and the last is the name on the DOM0 (obviously, an LVM partition).

Using this information, you can safely manipulate the volumes on a DomU without having to shut it down. Here is an example, assuming you want to do something with the home directory on a virtual named “book”.

1. Log into book. Issue command mount to see what device name is. umount that drive.
2. Assume it was xvda3
3. Log into DOM0. Issue command getXenMounts.pl | grep book | grep xvda3
4. output similar to book 26 51715 xvda3 /dev/virtuals/book.dailydata.net-home
5. Detach the drive with command xm block-detach book 51715 (or, xm block-detach 26 51715)
6. 'book' is the name, 26 is the domain number, and they are interchangable
7. Perform the work you need to do on /dev/virtuals/book.dailydata.net-home (resize, replace, whatever)
8. Re-attach to domain with command xm block-attach book phy:/dev/virtuals/book.dailydata.net-home xvda3 w
9. Again, book and 26 are interchangable as long as you have not changed the domain id in the process
10. Log back into the virtual and remount drive.

All of this with 0 downtime for all users except the one you needed to change anyway.

The script was written with extensibility in mind, and is released under the GPLv3, so feel free to look at it, adapt it for your own use, and have fun.

getXenMounts.pl

```
#! /usr/bin/perl -w

# getXenMounts.pl
# Author: R. W. Rodolico (http://www.dailydata.net)
# Date: 20121121
# Description:
# Script that uses xenstore to read all attached block devices on all
domU's
# in running DOM0. Output is a tab delimited file designed for easy
parsing
#
# Field 1: domu name
# Field 2: domu number
# Field 3: drive number (vbd number)
# Field 4: drive name on DOMU
# Field 5: drive number on DOM0
#
# designed to allow you easily determine what to use for xm block-
attach and
# xm block-detach, for example if a drive needs to be detached from a
DOMU to
```

```
# resize or replace.
#
# Simple usage example:
# determine which drive is mounted as /home on virtual named book:
#    1. Log into book. Issue command 'mount' to see what device name
is. umount that drive.
#       Assume it was xvda3
#    2. Log into DOM0. Issue command getXenMounts.pl | grep book | grep
xvda3
#       output similar to 'book 26 51715 xvda3
/dev/virtuals/book.dailydata.net-home'
#    3. Detach the drive with command 'xm block-detach book 51715' (or,
'xm block-detach 26 51715')
#       'book' is the name, 26 is the domain number, and they are
interchangable
#    4. Perform the work you need to do on
/dev/virtuals/book.dailydata.net-home (resize, replace, whatever)
#    5. Re-attach to domain with command 'xm block-attach book
phy:/dev/virtuals/book.dailydata.net-home xvda3 w'
#       Again, book and 26 are interchangable as long as you have not
changed the domain id in the process
#    6. Log back into the server and remount drive.

# Deficiencies:
#    Probably would be nice to accept a couple of parameters denoting
domain ID and device on domain, then get a single value returned
#    Subject to change of xenstore database format. This works on
xenstore v4.0.1

# Copyright 2012, R. W. Rodolico
#    This program is free software: you can redistribute it and/or
modify
#    it under the terms of the GNU General Public License as published
by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see .

# Revisions:
# 20130813 - fixed bug where domu name was not printing correctly
# 20121121 - Initial release


# get a list of running virtuals, return as an array
```

```
sub getRunningVirtuals {
   my @virtuals = `xenstore-list /local/domain/0/backend/vbd`;
   chomp @virtuals;
   return @virtuals;
}

# find all drives for a given virtual, then retrieve the device name on
the virtual (dev)
# and the device name on DOM0 (params)
# return a hash reference
sub getDrivesForVirtual {
   my $virnum = shift;
   my %output;
   my @drives = `xenstore-list /local/domain/0/backend/vbd/$virnum`; #
get all attached vbd's
   chomp @drives;
   foreach my $thisDrive ( @drives ) { # go through the list of drives
      # find the name on the virtual (sans /dev/, which is assumed)
      $output{$thisDrive}{'DomU'} = `xenstore-read
/local/domain/0/backend/vbd/$virnum/$thisDrive/dev`;
      chomp $output{$thisDrive}{'DomU'};
      # and, find the name on DOM0 that provides this
      $output{$thisDrive}{'Dom0'} = `xenstore-read
/local/domain/0/backend/vbd/$virnum/$thisDrive/params`;
      chomp $output{$thisDrive}{'Dom0'};
   }
   return \%output;
}

# helper sub that simply determines the "human" name of a given DomU
and returns it
sub getDOMUName {
   my $virtnum = shift;
   my $virtname = `xenstore-read /local/domain/$virtnum/name`;
   chomp $virtname;
   return $virtname;
}


my %output; # hash for output

my @virtuals = &getRunningVirtuals(); # store dom-id (integer) in array
for all running DOMU's

foreach my $thisVirt ( @virtuals ) { # for each of them
   # create a hash entry and attach the drive information for it
   $output{$thisVirt} = &getDrivesForVirtual( $thisVirt );
}

# print them out.
```

```
print "domu\tdomu\tdrive\tdomu\tdom0\n";
print "name\tid\tid\tdevice\tdevice\n";
print "====\t====\t=====\t====\t====\n";
foreach my $virtual ( keys %output ) {
   my $driveList = $output{$virtual};
   foreach my $drive ( keys %$driveList ) {
      print &getDOMUName($virtual) .
"\t$virtual\t$drive\t$$driveList{$drive}{'DomU'}\t$$driveList{$drive}{'
Dom0'}\n";
   }
}

1;
```