Bare Metal Backup And Restore

A TundraWare Inc. Technical Note

Author: Tim Daneliuk (tundra@tundraware.com)

Version: `$Id: baremetal.rst,v 1.124 2014/08/26 13:15:28 tundra Exp $`

Précis

Many commercial and open source solutions exist to solve the problem of creating backups that can be restored to "bare metal". That is, restoring a system *image* back to a disk when the machine will no longer boot or a disk has to be replaced. Image backups are preferred when rebuilding systems so you don't have to manually reinstall every application, system setting and so forth.

The purpose here was to do just that - create images capable of being restored onto, say, a blank, new hard drive, but *using only standard linux command line tools.*

> **Warning**
>
> Doing this wrong can **clobber your systems and its data**. What you see here is just a simple example for purposes of explaining the general approach. **You should not trust this approach unless you prove these procedures are satisfactory in YOUR OWN ENVIRONMENT.**

General Approach

The idea is to make an image snapshot right after you build a machine, and anytime thereafter you make significant changes to its operating system and application configuration. You might do this, say, right before patching a server so that, if patching breaks the server to the point where it will no longer boot, you can just "pour" the previous image onto the disk.

To do this, we reboot our target machine using the *Linux System Rescue CD*. This CD has all the tools on it we need to do both image creation and restoration. You can find the iso image for this disk here:

http://sourceforge.net/projects/systemrescuecd/

You will also need access to a place to store and retrieve your images. In the examples below, we used a NAS NFS share, though you could also use another local hard drive, SAN connected storage or even a USB-connected drive.

Example Environment

In our examples below, we're imaging a CentOS 6.5 machine. The only thing we need to image is the *operating system itself*. In this example, we know there are 2 partitions of interest:

`sda1` - The `/boot` partition - about 500MB

`sda2` **- The rest of the operating system, in this case contained** in LVM containers - about 52GB

The idea is that if the machine were to go dead, a disk failed, or what have you, this would be sufficient to get the replacement booting properly again. Presumably, you could then restore any data files you have from your standard backup/restore tools.

Backup Procedure

```
Boot from  the System Rescue CD

mount nas1:/shared /shared                      # Mount shared storage

sfdisk -d /dev/sda >/shared/ptbl                # Preserve the partition table

dd if=/dev/sda of=/shared/MBR bs=512 count=1    # Backup the Master Boot Record

dd if=/dev/sda1 of=/shared/boot.dd bs=12M       # Backup /boot

dd if=/dev/sda2 of=/shared/root.dd bs=12M       # Backup rootvg LVM (rest of OS)

reboot machine to make it operational again
```

How long this takes depends on what your write speed to the shared storage is and how big your partitions are. In this case `sda1` is only about 500MB and completed rather quickly. But `sda2` was about 52GB and took around 25 min to complete on a slow nfs mount - about 26MB/min in this case or about a quarter of the capacity of the 1Ge network connecting the NAS.

The `bs=12` is environment-specific and you'll have to find a setting for this that makes best use of your network and NAS or other storage device.

Depending on the speed of your processor and network, and the kind of data on your hard disk, it may be possible to speed the process up by compressing the output of `dd` on-the-fly. There's no reason to bother doing this for the MBR, but for the actual data partitions, it's worth a try. Replace the last two `dd` commands with:

```
dd if=/dev/sda1 | lzop >/shared/boot.dd.gz      # Backup /boot

dd if=/dev/sda2 | lzop >/shared/root.dd.gz      # Backup rootvg LVM (rest of OS)
```

Restore Procedure

Now, imagine that your OS is borked or the hard disk had to be replaced and you need to take the image from the backup above and getting running on the machine.

```
Boot from  the System Rescue CD

mount nas1:/shared /shared                      # Mount shared storage

dd if=/dev/zero of=/dev/sda bs=512 count=10     # Nuke any old boot/partion info on the disk

partprobe /dev/sda                              # Reread the partition table into the kernel

sfdisk /dev/sda </shared/ptbl                   # Restore the partition table
```

```
dd if=/shared/MBR of=/dev/sda bs=512 count=1   # Restore the Master Boot Record

dd if=/shared/boot.dd of=/dev/sda1 bs=12M      # Restore /boot

dd if=/shared/root.dd of=/dev/sda2 bs=12M      # Restore rootvg LVM (rest of OS)

Reboot machine to make it operational again
```

On the same network described above, restoring the 52MB `rootvg` took about 35 mins.

If you backed up using compression, then the last two `dd` commands will be:

```
dd if=/shared/boot.dd.gz | lzop -d >/dev/sda1  # Restore /boot

dd if=/shared/root.dd.gz | lzop -d >/dev/sda2  # Restore rootvg LVM (rest of OS)
```

Restoring To A Different Size Drive

Because this is partition based - that is, you are imaging and restoring *partitions*, not disks - you can actually restore to a physical disk that is a different size than the one from which the image was taken. Obviously, there has to be enough room for all the data on the new disk. You can even restore to a *smaller* disk overall, so long as your full partitions will fit on it. This might be the case if, say, you original installation's partitions did not use all of the disk's space.

More commonly, though, this makes it easy to lay your operating system down on a new, larger disk. Do an image of the old disk, restore it to the new disk, and then, while still running under the `System Rescue CD`, run `parted` or `gparted` to expand the partitions to use the additional disk space.

> **Warning**
>
> **Do NOT try restoring to anything other than the original disk with a machine that boots from SAN!!!** SAN-booted machines put information into their bootloaders about the boot LUN's WWID and the necessary configuration of HBAs to see those LUNs. If you present a bigger LUN with a different WWID, the OS bootstrap will probably fail, even though you can properly restore the image. There are ways around this. This document is not the place to find these ways.

Observations

- Upsides:
    1) You can use standard Linux commands to do imaged backups of your machine.
    2) It requires no software installation on the target machine.
    3) It's simple and scriptable.
    4) (In theory) It is OS and filesystem agnostic so long as ordinary partitioning practices are used.
    5) You can use this to migrate an OS to a different-sized disk.
- Downsides:
    1) You cannot do this while the machine is up and running because the blocks you're backing up may be changing during the backup period. That means this kind of imaging requires a machine outage.
    2) Every block in the partition gets copied whether it is used or not.

3) It's probably not as fast as imaging systems that understand the underlying filesystem layout - at least for filesystems that have a lot of unused space.

4) It requires you to boot from a CD. On consumer grade equipment, this is pretty much always possible. But, in data centers, this can be challenging because fewer and fewer servers have optical drives on them. The good news is that you can typically boot the target server from an iso image on your laptop by connecting to the server's ILO or KVM console system. In some cases, it may be necessary to create a bootable thumbdrive of the `Linux System Rescue CD` if your server only supports boot from USB.

- These tests were conducted on slow, consumer grade servers connected via 1Ge through an unmanaged switch. In an Enterprise class networking and NAS environment, we'd expect to see considerably faster backup and reimaging times, thereby minimizing server outage times.
- In *theory* this should also work on SAN-booted machines so long as the exact same LUN (WWID and size) is presented for the restore as was used for the backup. However, this was not tested and theory and Reality usually collide in rather nasty ways. Mr. Murphy is not our friend.
- Same story for VMs. Not tested. It's unclear whether a VM booted from a rescue disk would see the underlying disk storage (VMDK or whatever).
- Again, *theoretically* this should work with other operating system partitions and data partitions. But ... not tested.
- This works well for Unix-style operating systems because they boot with a full complement of drivers and discover what is on the machine at boot time.
- However, Windows may have a fit if you do this, especially if you restore to machine substantially different than the one where the backup image was created. First of all, Windows doesn't carry around any drivers it doesn't think it needs. Secondly, Windows licensing logic is designed to prevent this sort of thing as a deterrent to software piracy. This doesn't mean it cannot be done - it can - but it may take some extra fiddling after you reboot Windows.
- This approach assumes the disk is partitioned using standard `fdisk` type tools. Some OSs - notably the `*BSD` variants (aka "God's Own Operating System") - have the option to use their own disk slicing and labeling tools. This general approach will work, but you have to tweak it to ensure you preserve those boot loaders and custom slicing tables.

Copyright

License Terms

- By using this material in any way, you acknowledge you are doing so at your own risk. You agree to hold TundraWare Inc. harmless for any damage, direct or indirect, that this may or does cause to your computational environment, including, but not limited to, your or others' hardware, software, network, or data. You FURTHER AGREE TO HOLD TUNDRAWARE INC. HARMLESS FOR ANY ECONOMIC DAMAGE OR ANY OTHER ADVERSE CONSEQUENCE, DIRECT OR INDIRECT, CAUSED BY THE USE OF THIS MATERIAL.

Document Information

You can find the latest version of this document at:

http://www.tundraware.com/TechnicalNotes/Baremetal

A pdf version of this document can be downloaded at:

http://www.tundraware.com/TechnicalNotes/Baremetal/baremetal.pdf

This document produced with `emacs`, `RestructuredText`, and `TeXLive`.