Saving Real Storage with xip2fs and DCSS

Ihno Krumreich
Project Manager for SLES on System z
ihno@suse.de

Novell®

Agenda

- Overview
- DCSS
  - What is DCSS?
- xip2fs
  - What is xip2fs?
- DCSS
  - How to create?
- Maintenance
  - Problems
  - Solutions

Overview

Prerequisites

- Several Linux guests of the same Linux distribution running in a single z/VM system.
- Files / directories that are used by all guests in a read-only way could be identified.

Results

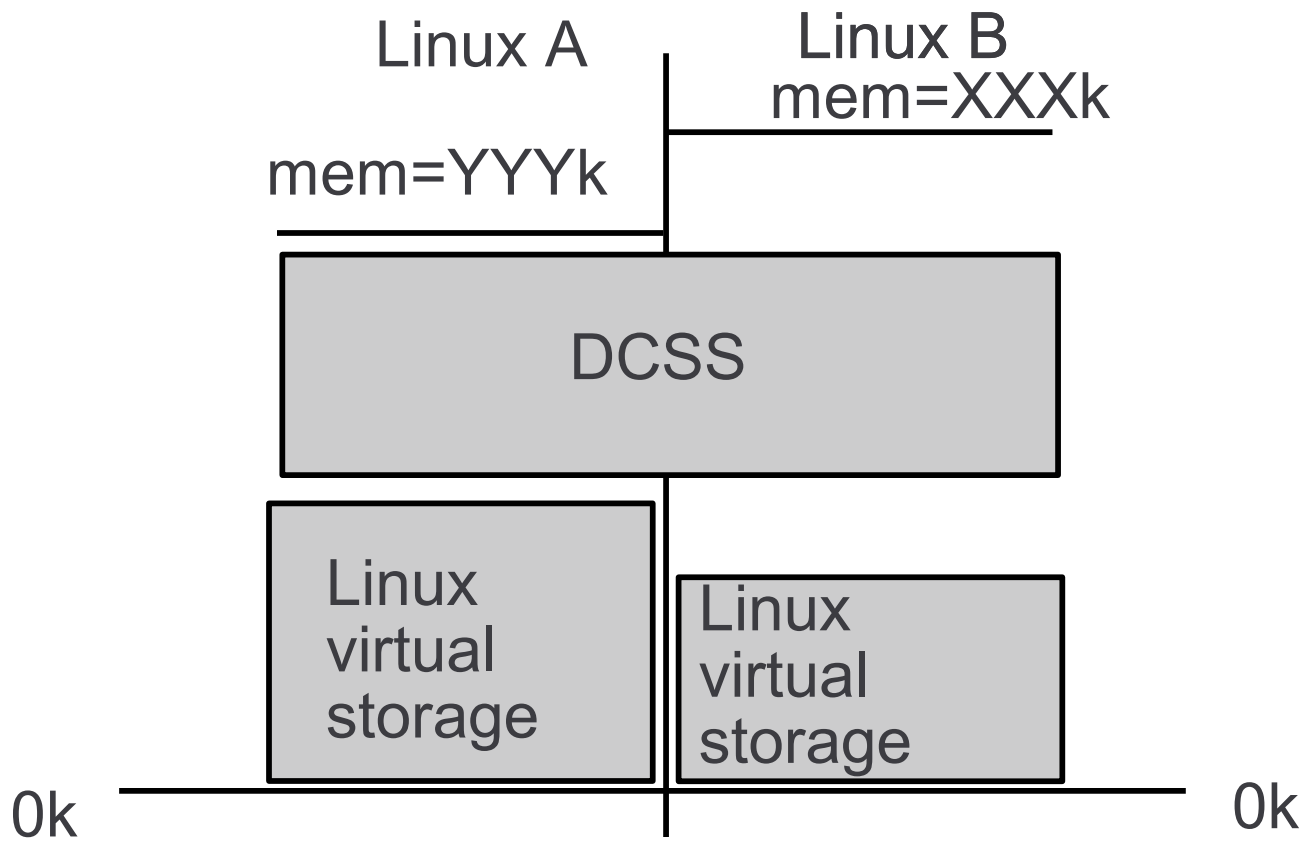
- Creating a DCSS with all those files and use them by the Linux guests gives you the following benefits:
  - Only one physical copy of those files exist for all Linux guests
    - save virtual/physical memory in z/VM
  - Execution of binary code files is faster

DCSS

What is a DCSS?

- DisContiguous Saved Segment
  - This is a memory segment saved to the z/VM spool area, which can be loaded by a Linux guest. The load address is specified at DCSS creation time.
  - We use it as a read-only memory segment.
  - The upper address limit is 1960MB for 31-bit Linux and 2GB for 64-bit Linux
  - The lower address must be greater than the maximum virtual memory used by the Linux guests.
  - A DCSS must reside below 2GB (z/VM < 5.4)
  - z/VM 5.4: A DCSS can be above 2GB, but any one DCSS is limited to 2GB in size. From Linux, multiple DCSS can be concatenated into one logical unit.

Memory Layout with DCSS

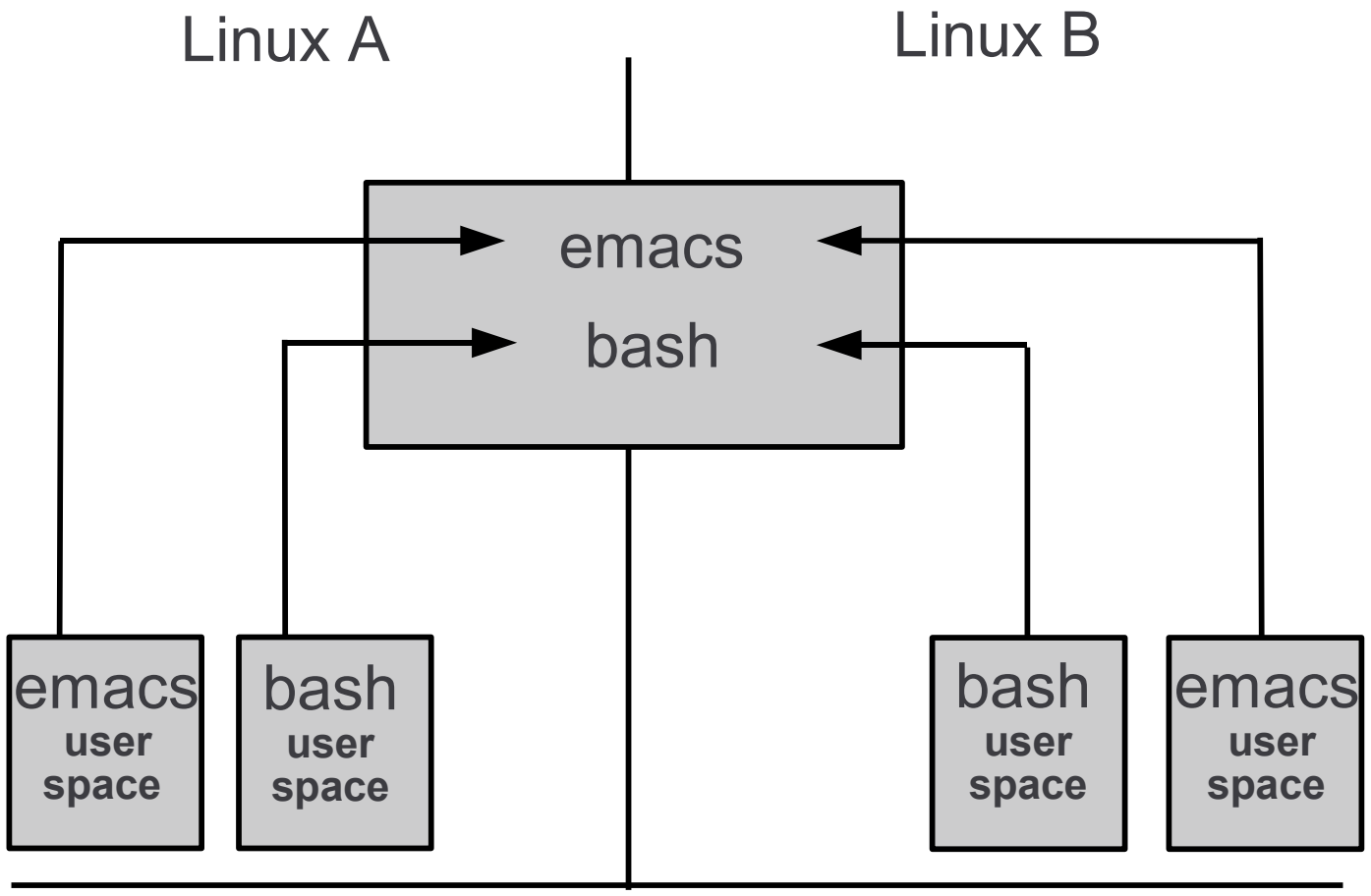


xip2fs

What is xip2fs?

- xip2fs is a new file system developed by IBM (e**x**ecute **i**n **p**lace **f**ile **s**ystem)
- The file system layout is identical to ext2, so all the tools for ext2 can be used.
- It is a read-only file system, no write functionality
- Read operations are done by returning the memory address, not by copying the data into buffers or cache.
- It is part of the upstream kernel code.

Executing programs out of a DCSS with xip2fs



Creating a DCSS with xip2fs

Creating a DCSS in xip2fs format

- Select the directories that should be part of the image
  - The following list names candidates:
    - Binary programs reachable via PATH variable
    - Directories listed in `/etc/ld.so.conf`
  - The following are not candidates:
    - Directories processes written to
    - Scripts. They are interpreted instead of being executed directly.
- Calculate the needed diskspace for the image
  - Issue the command `du -sk <directory>` for each identified directory
  - For each file add 4k for the meta data of the file
  - Add space for future updates (maintenance, additional files)

Creating a DCSS in xip2fs format

- Determine the start and end address of the DCSS
  - startaddress must be greater than or equal to the largest storage size of all linux guests that will use the DCSS. example: two guests, one with 512MB and one with 1GB
    - startaddress is 1GB (or higher)
  - endaddress is startaddress plus the size of the DCSS.
  - Start and end addresses must be on a 4K page boundary
  - Add `mem=<endaddress>` to parameter line in `/etc/zipl.conf` and execute `zipl`

Creating a DCSS in xip2fs format

- Creating the DCSS in z/VM
  - Shutdown Linux and ipl CMS
  - You need to have CP privilege class E
  - Define the segment with:
    `defseg <name of the DCSS> <first page number>-<last page number> sr`
    example: `defseg lnxshare 40000-5ffff sr`
  - Define the storage of the Linux guest greater than or equal to the endaddress of the DCSS
    `define storage 1536M`

Creating a DCSS in xip2fs format

- Save the segment with:
  `saveseg lnxshare`
- Check the success with:
  `query nss map`
- The segment should have an "A" in the "CL" column

```
01: CP QUERY NSS MAP
01: FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
01: 0168 LNXSHARE DCSS       N/A    40000  5FFFF   SR  A  00000   N/A       N/A
```

Creating a DCSS in xip2fs format

- Filling the segment with content
  - Before rebooting Linux, set the storage value back to the default value (#cp define storage 512m)
  - Load the DCSS block device driver with
    `modprobe dcssblk`
  - Attach the DCSS segment to Linux with
    `echo lnxshare > /sys/devices/dcssblk/add`
  - Make the DCSS writable for this Linux guest
    `echo 0 > /sys/devices/dcssblk/lnxshare/shared`
  - Create the file system
    `mke2fs -b 4096 /dev/dcssblk0`
  - Mount it
    `mount -t ext2 -o xip /dev/dcssblk0 /mnt`

Creating a DCSS in xip2fs format

- Filling the segment with content (cont.)
  - Copy the directories to the filesystem
    ```
    cp -a <list of directories> /mnt
    ```
  - Unmount the segment
    ```
    umount /mnt
    ```
  - Save the new contents of the DCSS
    ```
    echo 1 > /sys/devices/dcssblk/lnxshare/save
    ```
  - Remove the segment from the Linux guest
    ```
    echo lnxshare > /sys/devices/dcssblk/remove
    ```

Using the created DCSS in a guest

- For every guest do the following:
  - Add `mem=<endaddress of DCSS>` to parameter line in `/etc/zipl.conf` and execute `zipl`
  - Boot the Linux system
  - Load the driver:
    `modprobe dcssblk`
  - Mount the segment:
    `mount -t ext2 -o ro,xip /dev/dcssblk0 /mnt`
  - Bind mount the directories that are in the DCSS:
    `mount –bind /mnt/usr/bin /usr/bin`

Advantages of bind mounts

- Can be used on files and directories
- The setup is failsafe. If any of the steps to set up the DCSS for the guest are not successful, the guest is still operational.

Maintenance

Maintenance

Setting up multiple clients is quite easy.

But the real world requires

Maintenance

Setting up multiple guests with DCSS

- A "Gold master" is created
  - It contains all RPMs, application needed for all guests
  - It serves as a source for creating the DCSS
  - All clients are created by copying the Gold master and then configuring the client for its particular purpose.
  - This implies that all clients have the same set of RPMs.

How to do maintenance?

- Maintenance for a standard Linux guest:
  - Download the RPM from the vendor site
  - Shutdown services/applications touched by the RPM
  - Apply the RPM
  - Restart services / guest (depending on the RPM)

- RPM has full write access to the system
- Pre- and post-install scripts have full write access.

Problems applying maintenance to a system using DCSS/xip2fs

- With DCSS, a part of the system is read-only and cannot get updated like a standard Linux System!
- Problem of the system:
  - Part of the system is read-only
  - Part of the system is read/write
  - The read-only part exists only once
  - The read/write part has to be updated n-times
  - The read/write part may be modified by the sysadmin

Goals for maintaining Linux systems that are using DCSS/xip2fs

- A new DCSS segment is needed.
- The read/write part needs to be updated without destroying any sysadmin-modified parts.
- It must be possible to update the clients at different points in time.

Installing maintenance to a system with DCSS/xip2fs

- The Gold master (GM1) is copied to create a new Gold master (GM2) by applying the update.
- Run the `cmp` command on the files of GM1 and GM2 to find out which files have been changed. Call this list GMDIFF
- Remove from this list all files that are part of the DCSS and call the new list GMDELTA
- Create the new DCSS and write it to the system.
- All currently running Linux systems using the old DCSS will not see this update.
  - Only newly started systems will see the new version.

- Make sure you have enough z/VM spool space, as now two (or more!) copies of the DCSS are stored there.

```
#cp q alloc spool
```

- With this list, the new Gold master (GM2) and the new DCSS now every client could be updated.
  - Run the command rpm -Va on the Client (call the client C1). This gives a list of files that have been modified. Call the list C1DIFF.
  - Look which files are in both lists C1DIFF and GMDELTA. These files cause a conflict which has to solved by a person.
  - Files which are in GMDELTA and not in C1DIFF can be copied over from the Gold master to the client.

- Adjust the conflicting files.
- Logoff/logon the Client (to get the new DCSS)
- Boot the client and .... done.

Repeat the above steps for each client.

Useful Links and Documentation

Useful links

- IBM developerWorks
  - http://www-128.ibm.com/developerworks/linux/linux390/october2005_documentation.html
- Documents
  - CP Command and Utility Reference SC24-6081
  - How to use Execute-in-Place Technology with Linux on z/VM SC33-8286

Thank you for your attention!

Additional questions?

Now – Later – or send them to
lhno@suse.de


Novell®