Linux for System z Goody Bag - BOF

Session 9239 - February,2008

Mark Ver

Test and Integration Center for Linux - markver@us.ibm.com

Trademarks

Contents

return

zFCP SCSI

- Access SCSI disks over fibre channel attachments
- A typical setup makes use of an FCP switch. FCP point-to-point topology is not supported on all versions of the Linux distributions.
- The format of the 16 digit fcp_lun numbers can vary depending on the storage hardware, ex:

Lun on ESS:	0x5734000000000000
Lun on DS8000:	0x4057403400000000

- Modules needed: qdio, scsi_mod, scsi_transport_fc, zfcp, sd_mod(disk)/st(tape)
- Example manual setup:

```
modprobe zfcp
modprobe sd_mod
cd /sys/bus/ccw/drivers/zfcp
echo 1 > 0.0.a310/online
echo 0x5005076300ccafc4 > 0.0.a310/port_add
echo 0x572a000000000000 > 0.0.a310/0x5005076300ccafc4/unit_add
```

return

Configuring zFCP SCSI on SUSE Linux

- Each FCP device will have its own configuration file: /etc/sysconfig/hardware/hwcfg-zfcp-bus-ccw-0.0.xxxx

- The wwpn:lun mapping is defined with the ZFCP_LUNS parameter
  - Ex.

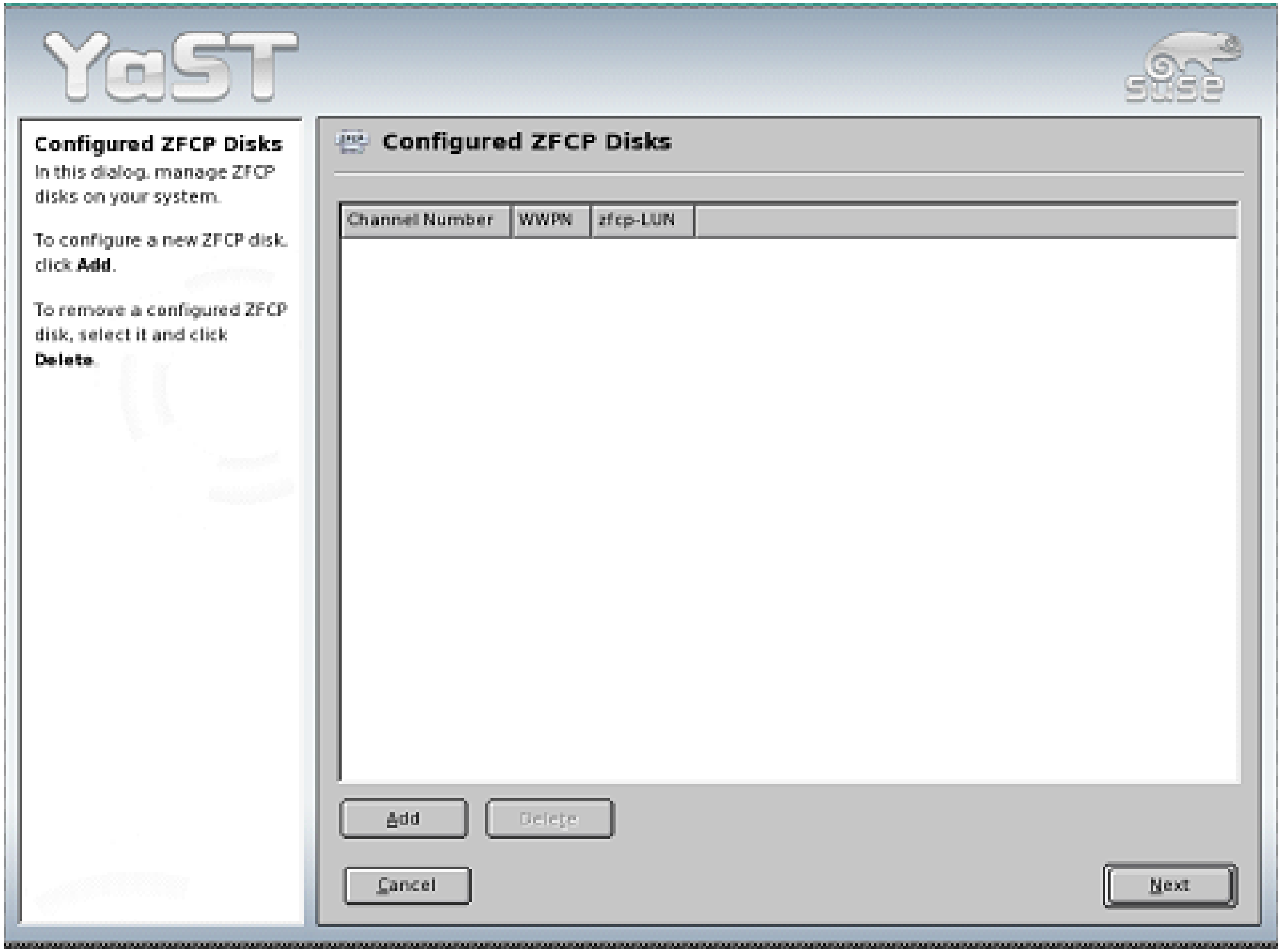
    ```
    ZFCP_LUNS="
    0x5005076300ccafc4:0x5735000000000000
    0x5005076300ceafc4:0x5735000000000000"
    ```

- Use zfcp_disk_configure to bring individual disks online
  - `zfcp_disk_configure 0.0.0100 0x5005076300c1afc4 0x572a000000000000 1`

- You can also use yast

- By default SUSE Linux has device files for /dev/sda through /dev/sdz

Configuring zFCP SCSI on SUSE Linux



Configured ZFCP Disks

In this dialog, manage ZFCP disks on your system.

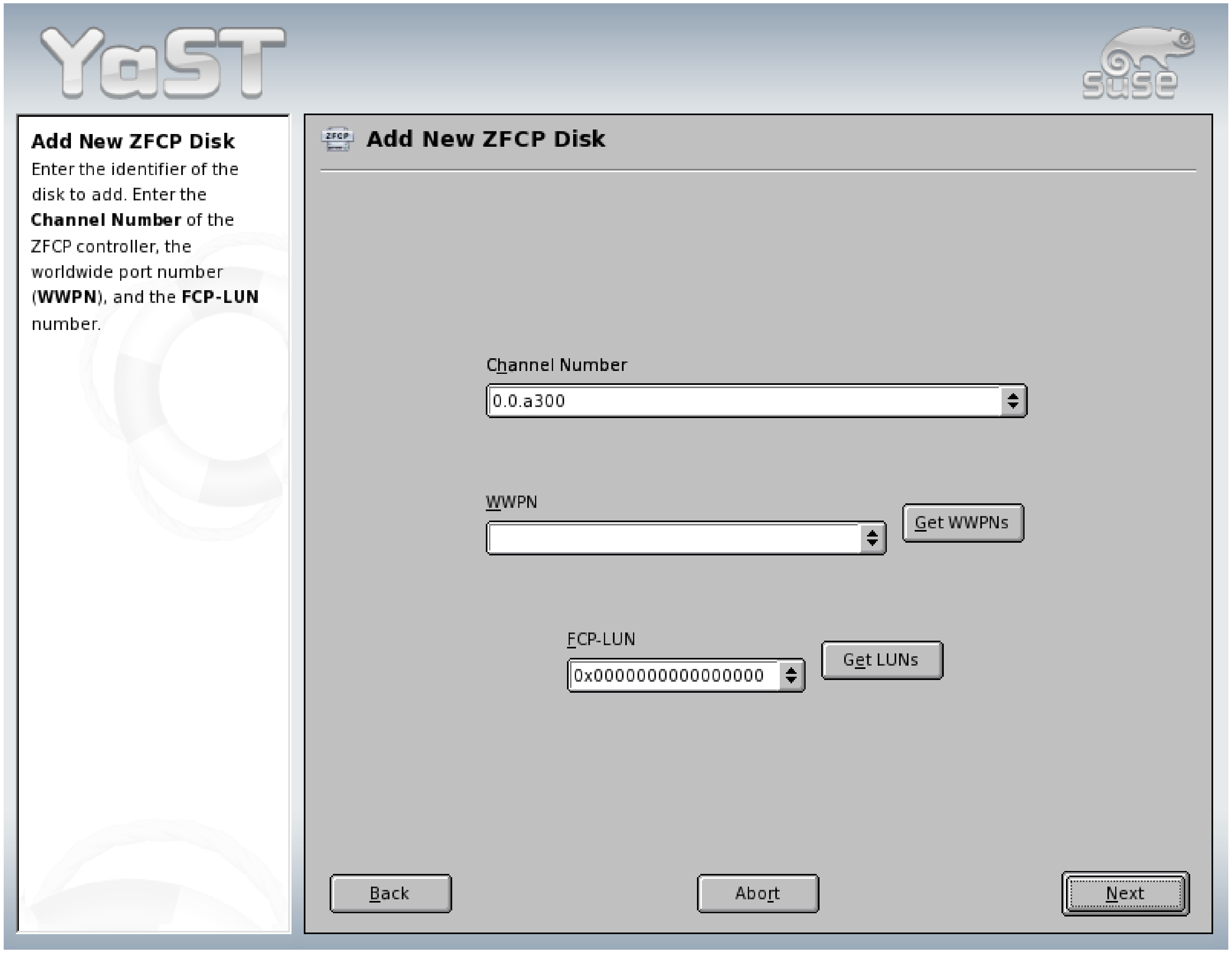
To configure a new ZFCP disk, click **Add**.

To remove a configured ZFCP disk, select it and click **Delete**.

Configured ZFCP Disks

Channel Number	WWPN	zfcp-LUN

Configuring zFCP SCSI on SUSE Linux



Add New ZFCP Disk

Enter the identifier of the disk to add. Enter the **Channel Number** of the ZFCP controller, the worldwide port number (**WWPN**), and the **FCP-LUN** number.

Add New ZFCP Disk

Channel Number: 0.0.a300

WWPN:

FCP-LUN: 0x0000000000000000

Configuring zFCP SCSI on Red Hat Enterprise Linux

- zFCP SCSI is set up in /etc/zfcp.conf
  - Ex.

```
0.0.a210 0x00 0x5005076300c2afc4 0x01 0x572c000000000000
0.0.a210 0x00 0x5005076300c2afc4 0x02 0x572d000000000000
```

- If you want the FCP devices to be available on boot up you need to run mkinitrd to get the configuration and needed modules into the initrd. Then you need to run zipl to pick up the new initrd and use it during boot up.
  - Ex.

```
cd /boot
mkinitrd -v --with=zfcp --with=sd_mod initrd.new `uname -r`
- rename initrd.new as needed to match ramdisk entry in /etc/zipl.conf …
zipl -V
```

- If you want to enable for current boot up, just run /sbin/zfcpconf.sh

Displaying zfcp scsi info

- Tools:
  - lszfcp (from s390-tools or s390utils package)

```
# lszfcp -D
0.0.0100/0x5005076300ccafc4/0x572b000000000000 0:0:0:0
0.0.0100/0x5005076300ccafc4/0x572a000000000000 0:0:0:1
```

  - lsscsi (from scsi or lsscsi packages)

```
# lsscsi
[0:0:0:0]    disk    IBM     2105800         .115  /dev/sda
[0:0:0:1]    disk    IBM     2105800         .115  /dev/sdb
```

  - sysfs:

```
# cd /sys/bus/scsi/devices/0\:0\:0\:0
# echo $(cat hba_id wwpn fcp_lun) $(basename $(readlink block*))
0.0.0100 0x5005076300ccafc4 0x572b000000000000 sda
```

Disabling a scsi device

- Manually disabling a scsi device from current configuration

```
# echo 1 > /sys/bus/scsi/devices/0:0:1:0/delete

# echo 0x572a000000000000 > /sys/.bus/ccw/drivers/zfcp/0.0.a310/0x5005076300ccafc4/unit_remove

# echo 0x5005076300ccafc4 > /sys/bus/ccw/drivers/zfcp/0.0.a310/port_remove

# echo 0 > /sys/bus/ccw/drivers/zfcp/0.0.a310/online
```

return

Other possible zfcp scsi topics

- Multipathing
  - Fibre Channel Protocol Implementation Guide:
    http://www.redbooks.ibm.com/abstracts/sg246344.html
  - Device Mapper Multipath tools:
    http://christophe.varoqui.free.fr/wiki/wakka.php?wiki=UsageFile

- SCSI IPL (FC9904)
  - How to use FC-attached SCSI devices with Linux on System z
    http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26cts02.pdf

- Using zfcp scsi for doing a stand-alone dump
  - Using the dump tools
    http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26cdt01.pdf

return

OSA Layer2 option

- Allows the OSA card to pass packets intact with Link Layer Control (LLC) headers to and from the Linux network stack.
- Enables more compatible support for Linux applications that require or examine the LLC headers (for example: tcpdump).
- The first system to use a shared OSA device sets the mode. All sharing systems will have to configure their network in the same mode.
- When directly attached to an OSA device (as opposed to using a VSWITCH configured for layer2 option) you need to specify a unique MAC for each Linux instance.
- Turn the option on by setting the ccwgroup device’s “layer2” attribute to “1”
  ex:

```
echo 1 > /sys/bus/ccwgroup/drivers/qeth/0.0.0920/layer2
```

return

OSA Layer2 option

- Configuration on SUSE, ex:
  - /etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.0600:

    QETH_LAYER2_SUPPORT=“1“
  - /etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.0600

    LLADDR="02:00:c0:a8:47:94"

- Configuration on RedHat, ex:
  - /etc/sysconfig/network-scripts/ifcfg-eth0:

    OPTIONS=“layer2=1”

    MACADDR="02:00:C0:A8:47:7F"

- Layer2 option during installation now available on SLES-10
  - SLES-10 base allows configuration for layer2 VSWITCH
  - SLES-10 SP1 allows configuration for VSWITCH and directly attached OSA device
  - Example prompts from a SLES-10 SP1 installation:

```
Enable OSI Layer 2 support?

1) Yes
2) No

> 1

MAC address>
```

return

Understanding where DASD configuration resides

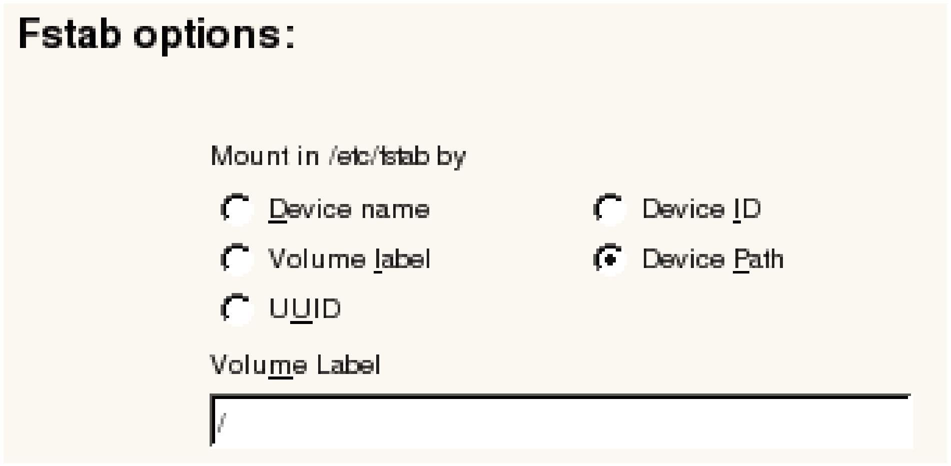
- SUSE configuration locations:
  - Kernel parameter line “dasd=…” option.
    - Configured via parameters line in /etc/zipl.conf
    - Requires that you run “zipl” after making changes to /etc/zipl.conf
  - The initial ramdisk (initrd)
    - Within the “linuxrc” or “init” script in the ramdisk (dasd_configure call per disk)
    - The list is pulled by mkinitrd from currently configured dasd (see lsdasd output)
    - Run zipl to use a newly created initrd
  - /etc/sysconfig/hwcfg-dasd-bus-ccw-0.0.*
    - Example: hwcfg-dasd-bus-ccw-0.0.0201

- RedHat configuration locations:
  - The initial ramdisk (initrd)
    - Within the “init” script in the ramdisk (insmod call to dasd_mod dasd=…)
    - The list is pulled by mkinitrd from the dasd_mod options in /etc/modprobe.conf
      - Example: options dasd_mod dasd=201
    - Run mkinitrd and then zipl after making changes to /etc/modprobe.conf

return

Things to watch out for with SLES-10 SP1 DASD

- SLES-10 SP1 by default now references DASD according to the device id from the storage hardware, ex:
  parameters = "root=/dev/disk/by-id/ccw-IBM.750000000M1881.2c23.1c-part1 TERM=dumb"
- This affects systems that are using minidisks on the same physical device as well as cloning methods that rely on flashcopy or DDR of DASD.
- Can get around it by changing /etc/zipl.conf and /etc/fstab to use identifiers that do not care about the physical device id, ex:
  - /dev/disk/by-path/ccw-0.0.0201-part1
  - /dev/dasda1
  - LABEL=rootfs
- It can be modified during installation - just open up the "fstab options" when creating/editing partitions:


Fstab options:

Mount in /etc/fstab by

Device name | Device ID
Volume label | [✓ Device Path]
UUID

Volume Label
/

return

DIAG access for disks

- Simplifies IO instruction path for z/VM guests by letting z/VM handle the IO directly for the guest OS (diagnose x’250’ instruction).
- Supported for 64-bit on z/VM 5.2 and higher. Supported for 31-bit on all z/VM releases.
- 64-bit support requires CONFIG_DASD_DIAG option in the kernel. Early distro levels did not provide dasd_diag_mod driver with the 64-bit system.
- Requires CMS formatted or ldl formatted dasd (do not cdl format the dasd!).
- Depending on the kernel level, may be susceptible to an old bug where DIAG against FBA devices only worked correctly when the FBA device was CMS formatted with block size 512
  - Ex.   Format 200 c (blksize 512

DIAG access for disks

- DIAG manual set up example:
  - CMS format the dasd from VM:

```
format 200 k (blksize 512
```

  - Boot up linux, load drivers and enable a dasd for diag use:

```
#  modprobe dasd_fba_mod
#  modprobe dasd_diag_mod
#  echo 1 > /sys/bus/ccw/devices/0.0.0200/use_diag
#  echo 1 > /sys/bus/ccw/devices/0.0.0200/online
```

  - Check that disk is using the DIAG module for access:

Ex.

```
# lsdasd
0.0.0201(ECKD) at ( 94:  0) is dasda      : active at blocksize 4096, 601020
blocks, 2347 MB
0.0.0200(DIAG) at ( 94:  4) is dasdb      : active at blocksize 512, 2048000
blocks, 1000 MB
```

return

DIAG access for SUSE

- Using DIAG with SUSE on boot up:
  - Add the module “dasd_diag_mod” to the INITRD_MODULES list in /etc/sysconfig/kernel; then run mkinitrd and zipl.
  - Set the DASD_USE_DIAG flag in the /etc/sysconfig/hardware/hwcfg-* file for the target dasd device:

    DASD_USE_DIAG="1“
  - If the hwcfg-* file doesn’t exist yet you can create the configuration file and set diag use all in one go with the command “dasd_configure <ccwid> <online> <use_diag>”, ex:

    dasd_configure 0.0.0200 1 1
  - Or you can just use yast2 gui -> hardware -> DASD panel to set DIAG on the selected devices.

return

DIAG access for Red Hat

- Using DIAG with Red Hat Enterprise Linux on boot up:
  - Modify dasd_mod options in /etc/modprobe.conf to indicate "(diag)" for a range of devices in the dasd list.

    ```
    options dasd_mod dasd=201,202,200(diag),300
    ```

  - Use mkinitrd to create new initrd that loads the dasd_diag_mod in the right order and that picks up the changes from modprobe.conf:

    ```
    cd boot
    mkinitrd --preload=dasd_diag_mod --with=dasd_fba_mod –f initrd* $(uname –r)
    zipl –V
    ```

Integrated ASCII Console

- Provides a console for Linux systems on LPAR that allows execution of full screen commands (such as vi).
- Accessed via the “Recovery” menu on the HMC:

Integrated ASCII Console

- Requires console statement in the kernel parameters line, ex:

```
root=/dev/dasda1 console=ttyS1 console=ttyS0
```

- Requires a mgetty line in /etc/inittab to provide a logon mechanism, ex:
  - SUSE:

```
2:2345:respawn:/sbin/mingetty --noclear /dev/ttyS1 vt220
```

  - Red Hat:

```
2:2345:respawn:/sbin/mingetty --noclear /dev/ttysclp0 vt220
```

- Requires a device entry, “ttyS1” for SUSE and “ttysclp0” for Red Hat, in /etc/securetty to allow root logon from the console device
- Console supports only ASCII character set (not UTF8) – requires playing with the TERM and LANG settings to get more compatible support for some applications

Integrated ASCII Console

- z/VM 5.3 introduces support for attaching the Integrated ASCII Console to a z/VM guest.
- Configure the Linux system in the same way as you would for accessing the console on an LPAR.
- Attach the console to the target guest, ex:

```
Ready; T=0.01/0.01 21:29:18
attach sysascii to LAC0034
SYSASCII attached to LAC0034
```

- Access the console on the HMC same as you would for a Linux system booted directly on an LPAR.
- The console can only be attached to only one guest at a time, ex:

```
Ready; T=0.01/0.01 21:29:26
attach sysascii to LAC0008
HCPSEA122E SYSASCII already attached to LAC0034
```

- The feature is intended mostly to help with system recovery (access to fullscreen editors and such even when the network is down).

Cooperative Memory Management (CMM)

- The CMM feature allows an external entity, like VMRM, to dynamically adjust the amount of usable memory available to a Linux system running under z/VM.

- CMM works by allocating pages of memory to a special page pool, and then sending the diagnose X'10' instruction to notify z/VM that these pages are available for reuse.

- Enabled in the kernel with the following options:

```
CONFIG_CMM=m
CONFIG_CMM_PROC=y
CONFIG_CMM_IUCV=y
CONFIG_SMSGIUCV=m
```

- Example module load:

```
modprobe smsgiucv
modprobe cmm sender=VMRMSVM
```

- On distros running with kernel 2.6.16 or higher, can verify the assigned sender through a file on the sysfs:

  Ex. cat /sys/module/cmm/parameters/sender

return

Cooperative Memory Management (CMM)

- The CMM /proc interface provides 3 configuration settings:
  - /proc/sys/vm/cmm_pages

    -> used to read or set size of static page pool (memory block immediately available for z/VM reuse)
  - /proc/sys/vm/cmm_timed_pages

    -> used to read or set size of timed page pool (memory block made available to z/VM gradually according to timeout rate)
  - /proc/sys/vm/cmm_timeout

    -> used to set or read release rate of timed page pool. Uses 2 values. Example: “echo 100 30 > cmm_timeout” (100 pages made available to z/VM every 30 seconds)

- The CMM special message interface provides 3 corresponding instructions:
  - SMSG <guestname> CMM SHRINK <cmm_pages_value>
  - SMSG <guestname> CMM RELEASE <cmm_timed_pages_value>
  - SMSG <guestname> CMM REUSE <pages> <seconds>

Cooperative Memory Management (CMM)

- VMRM and CMM
  - Officially supported on z/VM 5.2 with APAR VM64085
  - VMRM new configuration setting "NOTIFY MEMORY <user list> …", ex:

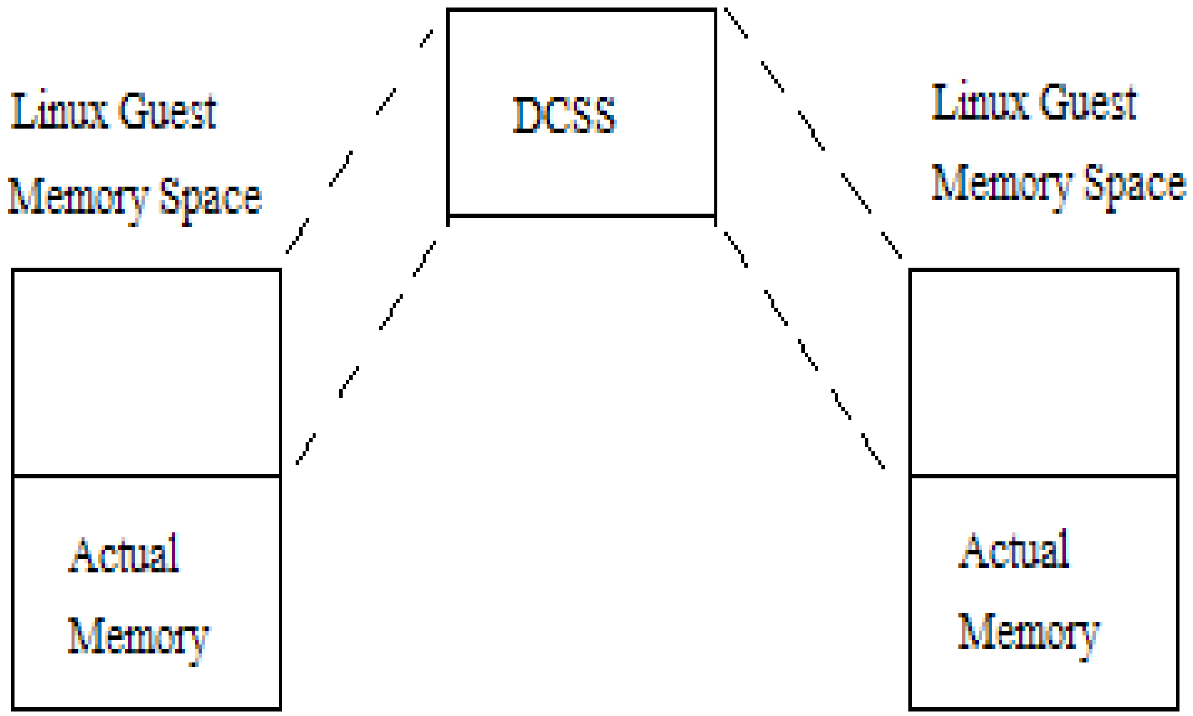
    NOTIFY MEMORY LTIC0001 LINUX* MARKVER DIST01 DIST34
  - Basically the guests in the list have been volunteered to let VMRM adjust their memory when needed.
  - When z/VM is memory constrained, VMRM sends CMM special message instructions to the guests to give up some memory for z/VM to reuse elsewhere. Target guests have to be configured to receive special message instructions (check the "SMSG" and "SET SMSG" commands section in the z/VM CP Commands and Utilities Reference).
  - When z/VM is no longer constrained; VMRM sends the appropriate instructions to shrink the CMM special page spools, thus allowing linux to reclaim the memory for its own use.
  - Checkout http://www.vm.ibm.com/sysman/vmrm/vmrmcmm.html for more information.

return

DCSS and xip

- z/VM defseg and saveseg commands allow you to map pages of current memory contents and store them away in a disk backed memory allocation that can be made commonly accessible to multiple guests.
- The DCSS device driver is used to provide disk-like access to a such a saved segment.
- The XIP technology allows you to treat code on a memory backed file system as if it were a part of the virtual memory space.
- Together these allow multiple linux guests to share one memory copy of commonly executed code (such as often used library routines), and reduce over all memory usage by linux guests.

DCSS and xip



- The Linux memory space is extended with the mem= kernel parameter to allow reference of additional page ranges (enough to cover the size of the DCSS).
- DCSS device driver for Linux on System z is used to provide disk-like access to the Discontiguous Saved Segment.
- Built-in xip2 support in ext2/ext3 drivers is used to map the DCSS contents to virtual memory when mounting the file system.

return

DCSS and xip

- Example: Creating a DCSS in a storage gap (64bit system)
  - Deciding on my DCSS address range:
    start-address: 512M = 0x20000000
    end-address: 1G – 1 byte = 0x3FFFFFFF
  - Calculate page frame number for start and end address:
    start-address / 4K page size = 0x20000000 / 0x1000 = 0x20000
    end-address / 4K page size = 0x3FFFFFFF / 0x1000 = 0x3FFFF
  - Define address range to be saved:
    defseg MYDCSS 0x20000-0x3FFFF sr
  - Make sure entire DCSS address range is reachable:
    def stor 2g
  - Allocate the DCSS space
    saveseg MYDCSS
  - Define a 512M gap for the DCSS (and ~2G of available memory)
    def stor config 0.512m 1g.1536m

DCSS and xip

- Example continued: Creating a DCSS in a storage gap (64bit system)
  - Boot up linux:

    ipl 201
  - Login and load dcssblk module:

    # modprobe dcssblk
  - Add the DCSS for dcssblk access:

    # echo MYDCSS > /sys/devices/dcssblk/add
  - Change DCSS access mode to “exclusive-writable” so DCSS can be filled with data (note what this does is create a private copy of the existing DCSS so make sure there is enough spool space to support two copies of the DCSS! Also, must have class E privileges to actually make it work):

    # echo 0 > /sys/devices/dcssblk/MYDCSS/shared
  - Check for your device file:

    # ls -l /dev/dcssblk0
  - If it doesn’t exist then create the device file:

    # cat /sys/devices/dcssblk/MYDCSS/block/dev

    252:0

    # mknod /dev/dcssblk0 b 252 0

DCSS and xip

- Example continued: Creating a DCSS in a storage gap (64bit system)
  - Create an ext2 file system on the block device

    ```
    # mke2fs -b 4096 /dev/dcssblk0
    ```

  - Mount the filesystem (actual format depends on distro level and support for xip):

    ```
    # mount -t xip2 -o ro,memarea=MYDCSS /dev/dcssblk0 /mnt
    ```

  - Copy desired data onto the DCSS:

    ```
    # cp -a /usr/lib64/* /mnt
    ```

  - Issue the save request (the actual save request is done after the device is unmounted):

    ```
    # echo 1 > /sys/devices/dcssblk/MYDCSS/save
    ```

  - Unmount the DCSS (at this point guests that request to open the DCSS see the new changed copy. The original copy of the DCSS is retained for guests that were already accessing it and is removed when the last guest has stopped usage):

    ```
    # umount /mnt
    ```

  - Remove the DCSS device:

    ```
    # echo MYDCSS > /sys/devices/dcssblk/remove
    ```

DCSS and xip

- The DCSS data is saved to spool space (so have plenty of spool available)
- DCSS requires class E privileges to create (and to modify)
- z/VM supports DCSS max 2047 MB (page frame 0x7feff) for 64bit and only up to 1960 MB (page frame 0x7a7ff) for 31bit.
- The z/VM "define storage" command can be used to define multiple memory segments with a gap. And in that case the DCSS can be placed within the gap rather than above the highest address of actual available memory.
- The dcssblk major device number is not fixed, but is assigned dynamically when the driver is loaded. This must be accounted for if the device file has to be created manually.
- Enablement of DCSS with XIP on boot up requires specialized boot scripts. These are available from the execute-in-place documentation on the developerworks site.
- For more details check the following text:
  - Documentation/filesystems/xip.txt file in the linux kernel source
  - How to use Execute-in-Place Technology with Linux on z/VM - SC33-8287-00:
    http://www-128.ibm.com/developerworks/linux/linux390/october2005_documentation.html
  - z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES9:
    http://www.redbooks.ibm.com/abstracts/sg246695.html

XPRAM

- Provides mechanism for using System z expanded storage under Linux (typically used to augment S/390 31-bit architecture which can access at most 2 gigabytes of main memory)
- The storage can be used to provide fast swap device or fast file systems.
- Can divide the available expanded storage with up to 32 partitions. The device nodes are typically called /dev/slram0 - /dev/slram31
- Example module load:
  modprobe xpram devs=2 sizes=512000
- SUSE now provides configurations files for setting up 1 xpram device
  - /etc/sysconfig/xpram
  - /etc/init.d/xpram

Discussion of other possible topics

- Source VIPA
  - http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/docu/l26cdd03.pdf
- VLAN tagging
- Hipersockets
- Auto-installations (kickstart & autostart)
  - https://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/index
  - YaST module Autoinstallation

return