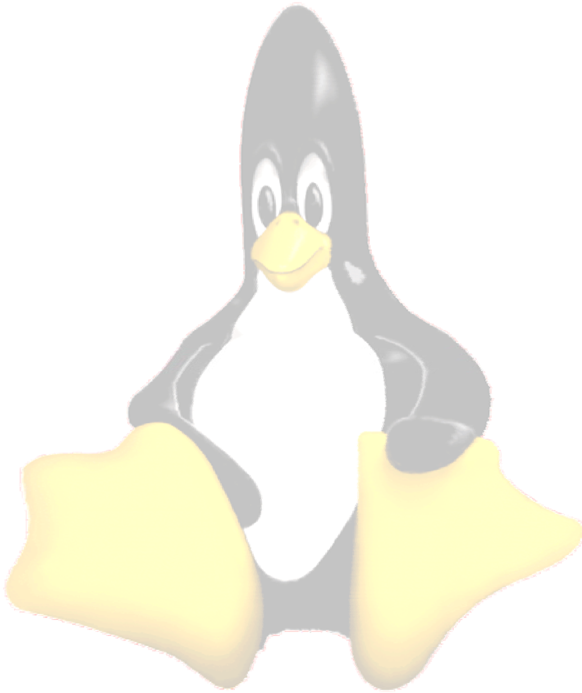
zSeries Technical Conference

The Art of Squeezing Penguins



Rob van der Heij
IBM Netherlands
rvdheij@nl.ibm.com

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

BookManager*
DB2*
DFSMS/MVS*
DFSMS/VM*
e-business logo*
Enterprise Storage Server
ESCON*
FICON
GDDM*
IBM*
IBM logo*
Language Environment*
Multiprise*
MVS
NetRexx
OpenEdition*
OpenExtensions
OS/390*
Parallel Sysplex*
PR/SM
QMF
RACF*
RAMAC*
S/390*
S/390 Parallel Enterprise Server
VisualAge*
VisualGen*
VM/ESA*
VSE/ESA
VTAM*
z/Architecture
z/OS
z/VM
zSeries

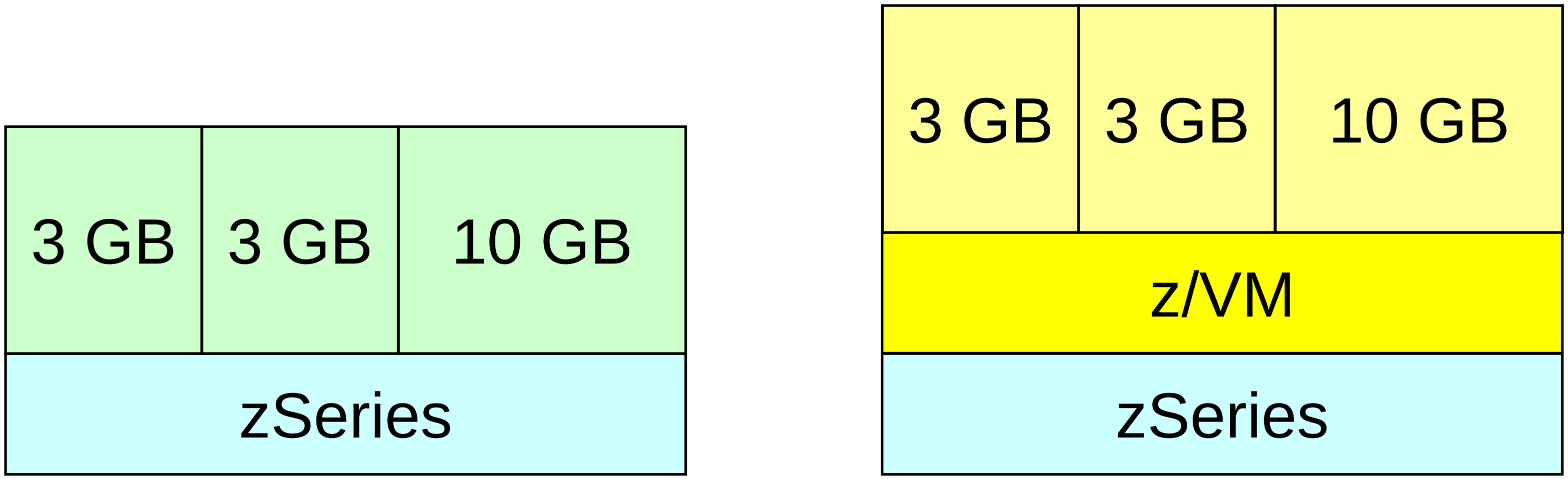
The following are trademarks or registered trademarks of other companies.

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation; LINUX is a registered trademark of Linus Torvalds; Penguin (Tux) compliments of Larry Ewing; Tivoli is a trademark of Tivoli Systems Inc.; Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries; UNIX is a registered trademark of The Open Group in the United States and other countries; Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation; SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC. * All other products may be trademarks or registered trademarks of their respective companies.

Notes: Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Allocating Memory to Servers

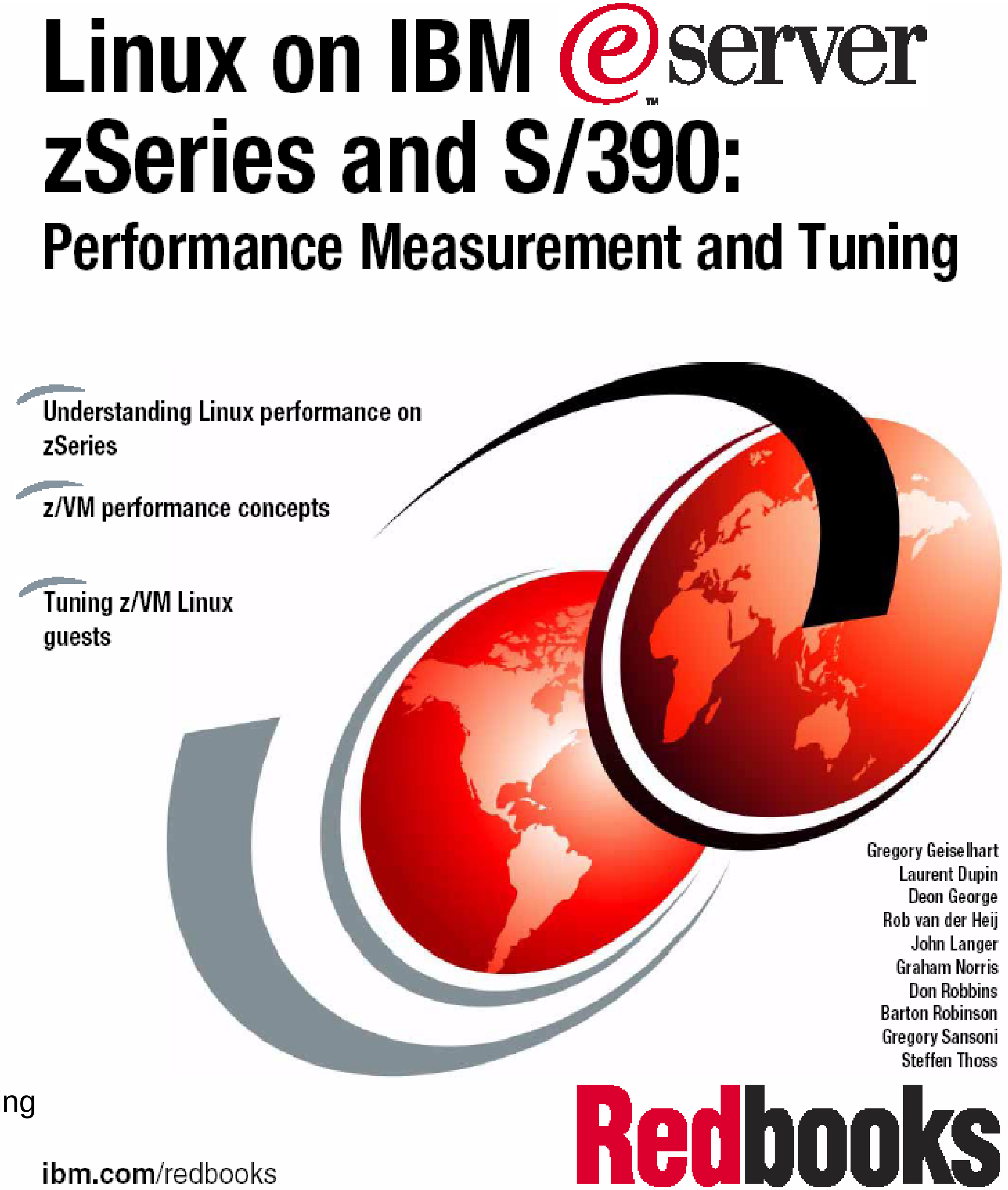
- **LPAR does not provide virtual memory**
  - Only hand out what you have physically installed
  - You could define virtual machines in the same way



Agenda

- **Virtual Memory**
- **Define Linux “footprint”**
- **Measuring Linux memory usage**
- **Reducing Linux footprint**
  - Sharing memory
  - Collaborative Memory Management
- **Results**

Redbook



Contents

Virtualization and server consolidation

z/VM memory and storage concepts

Linux virtual memory concepts

Tuning memory for z/VM Linux guests

Examining Linux swap device options

CPU resources and the z/VM scheduler

Tuning processor performance for z/VM Linux guests

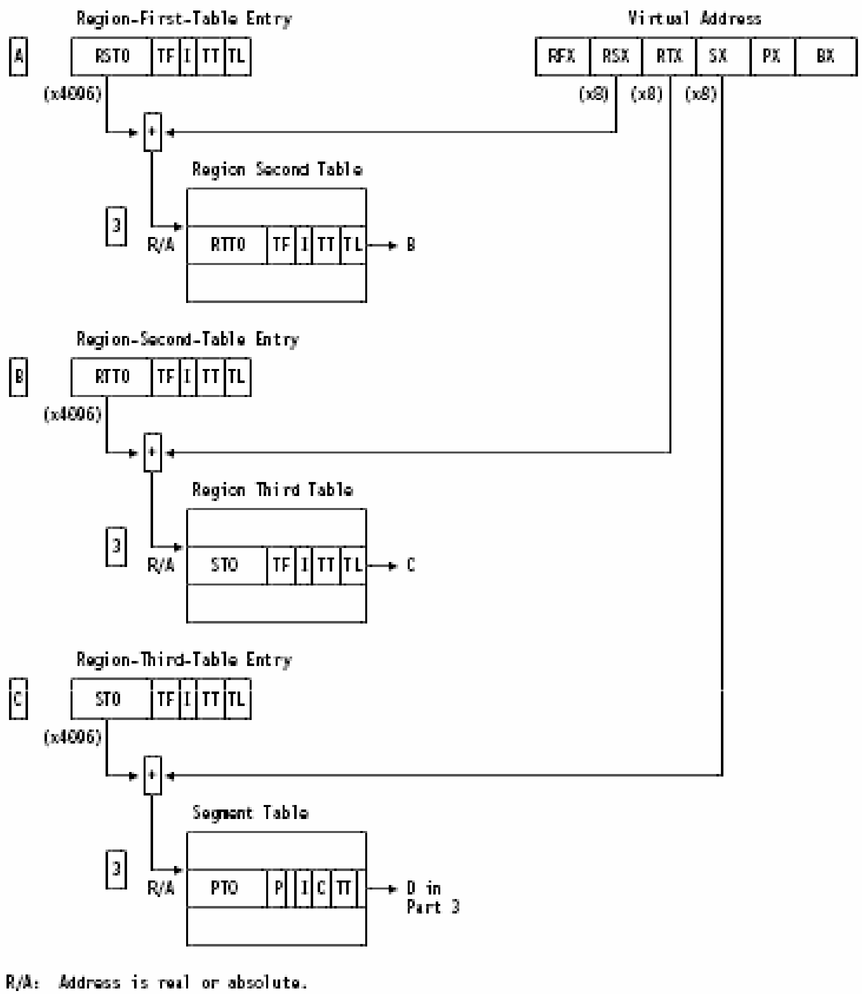
Tuning DASD performance for z/VM Linux guests

Measuring the cost of OSA, Linux, and z/VM networking

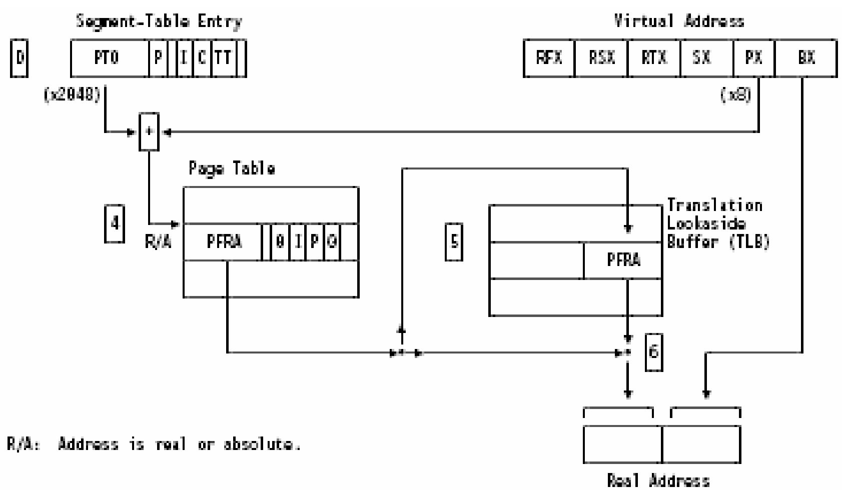
Virtual Memory

- **Virtual Memory builds on**
  - Address Translation hardware
  - Limited amount of real memory
  - Paging space
- **Virtual Memory provides**
  - Isolation by providing each their own virtual linear space
  - A way for a memory manager to over commit memory
    - Sometimes incorrectly referred to as “sharing memory”

Virtual Memory

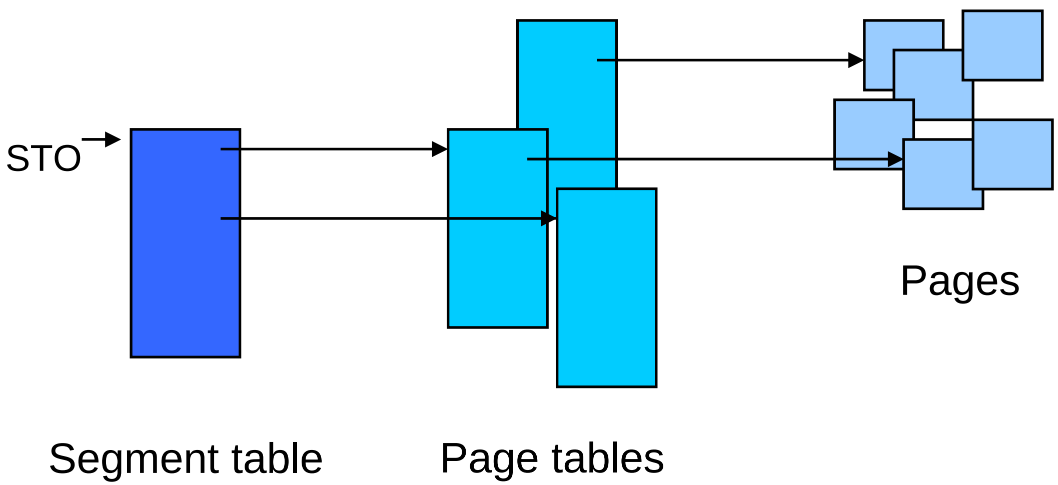


- **Dynamic Address Translation Translate virtual to real address**
  - Also for access registers



Virtual Memory

- **Simplified picture for small virtual machines**
  - Segment table and page tables span virtual memory
  - One such set for each virtual machine



Not all page table entries lead to a page frame

- Resident in storage
- Paged out to paging space
- Never touched (not present)

Virtual Memory

- **Segment table and page tables for each virtual machine**
- **Cost of page tables would be considerable**
  - VM also needs some management structures with the pages
  - Fortunately only needed when the segment is not empty

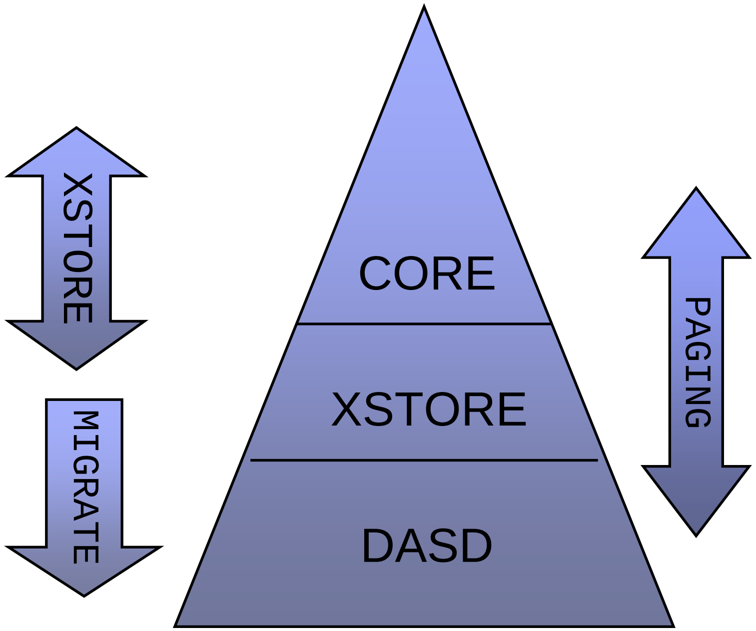
	Address Range	Size	Needed for full address space	Total cost
Page Frame	**4K**	**4 K**	**524288**	**2 G**
Page Table	**1 M**	**2 K**	**2048**	**4 M**
Segment Table	**2 G**	**16 K**	**1**	**16 K**

Virtual Memory

- **z/VM Memory Management will**
  - At first reference to a virtual page
    - Allocate a page table when entire segment is absent
    - Allocate a page frame to hold the page
  - Page-out unused pages to paging space to make room
    - Driven by demand for free page frames
  - Page-in referenced page when needed again
  - Migrate pages from expanded storage to paging disks

z/VM Paging

- **Basic numbers with CP INDICATE LOAD**
  - Use Performance reporting for trends



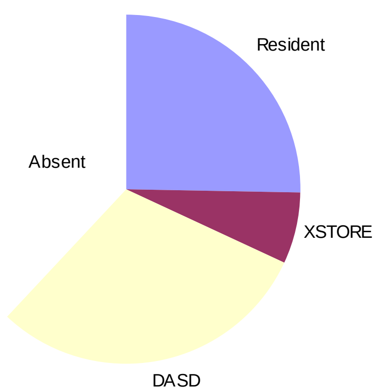

```
CP IND
AVGPROC-004% 02
XSTORE-000482/SEC MIGRATE-0194/SEC
MDC READS-000146/SEC WRITES-000001/SEC HIT RATIO-081%
STORAGE-050% PAGING-0321/SEC STEAL-000%
```


Footprint

Use a virtual NIC instead of dedicated OSA: ~1820 pages

- **Virtual machine memory breakdown**
  - Resident
  - Paged out to DASD
  - Paged out to XSTORE
  - Not there

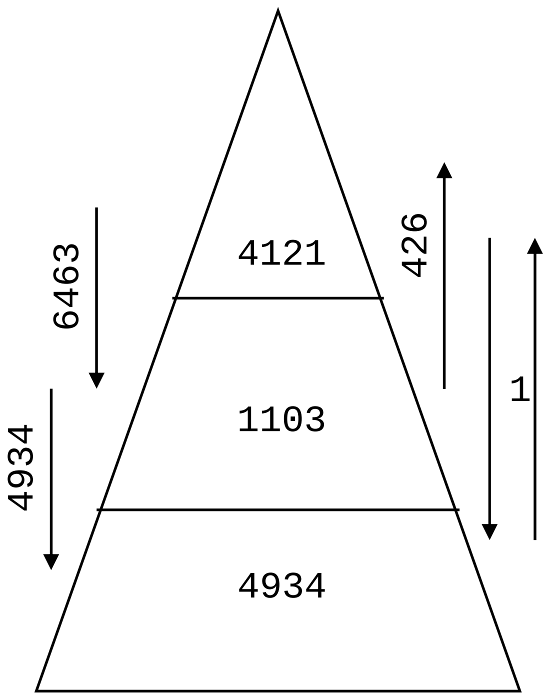
64 M = 16384 pages

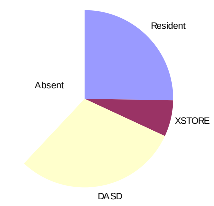



```
CP IND USER LINUX40
USERID=LINUX40  MACH=XA  STOR=64M VIRT=V XSTORE=NONE
IPLSYS=DEV 0205 DEVNUM=00014
PAGES: RES=00004121 WS=00004111 LOCK=00000010 RESVD=00000000
NPREF=00004934 PREF=00000000 READS=00000001 WRITES=00004935
XSTORE=001103 READS=000426 WRITES=006463 MIGRATES=004934
CPU 00: CTIME=00:44 VTIME=000:03 TTIME=000:04 IO=001987
        RDR=000000 PRT=000000 PCH=000000
```


Footprint

- **Most numbers from IND USER add up well**
  - Number of pages on DASD is not always accurate

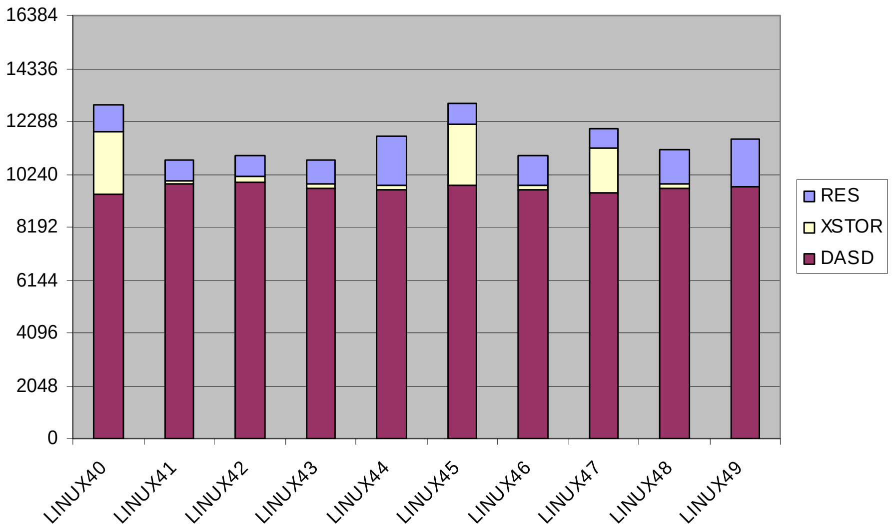





```
CP IND USER LINUX40
USERID=LINUX40  MACH=XA  STOR=64M VIRT=V XSTORE=NONE
IPLSYS=DEV 0205 DEVNUM=00014
PAGES: RES=00004121 WS=00004111 LOCK=00000010 RESVD=00000000
NPREF=00004934 PREF=00000000 READS=00000001 WRITES=00004935
XSTORE=001103 READS=000426 WRITES=006463 MIGRATES=004934
CPU 00: CTIME=00:44 VTIME=000:03 TTIME=000:04 IO=001987
        RDR=000000 PRT=000000 PCH=000000
```

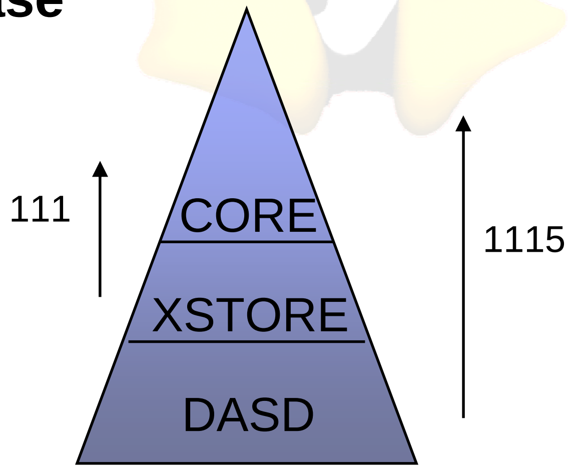

Footprint

- **Compare storage residency of virtual machines**
- **Footprint consists of different components**
  1. Resident pages
  2. Expanded storage
  3. DASD
- **Focus on resident pages and on expanded storage**



Footprint changes over time

- **INDICATE USER shows user footprint size**
- **Activity in Linux makes the footprint increase**
  - 1115 from DASD
  - 111 from XSTORE
  - 265 fresh pages

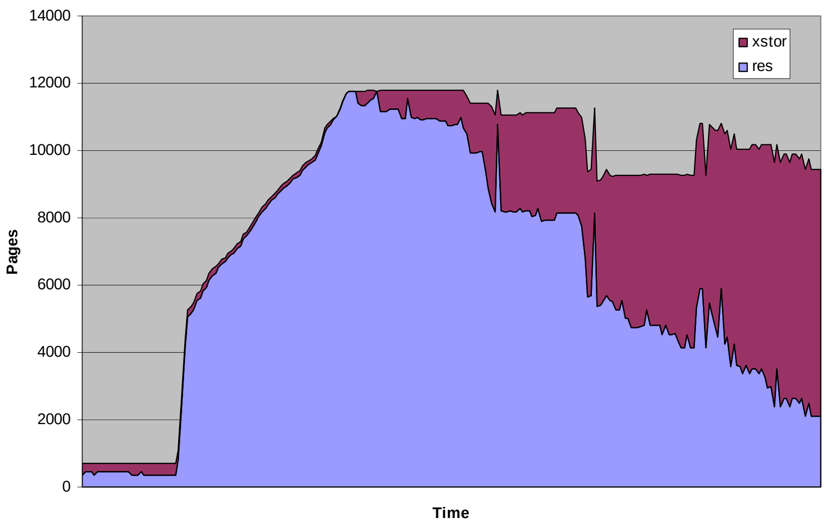



```
PAGES: RES=00004121 WS=00004111 LOCK=00000010 RESVD=00000000
NPREF=00004934 PREF=00000000 READS=00000001 WRITES=00004935
XSTORE=001103 READS=000426 WRITES=006463 MIGRATES=004934
CPU 00: CTIME=00:44 VTIME=000:03 TTIME=000:04 IO=001987
```

```
PAGES: RES=00005612 WS=00005602 LOCK=00000010 RESVD=00000000
NPREF=00004934 PREF=00000000 READS=00001116 WRITES=00004935
XSTORE=000992 READS=000537 WRITES=006463 MIGRATES=004934
CPU 00: CTIME=00:50 VTIME=000:04 TTIME=000:04 IO=002035
```

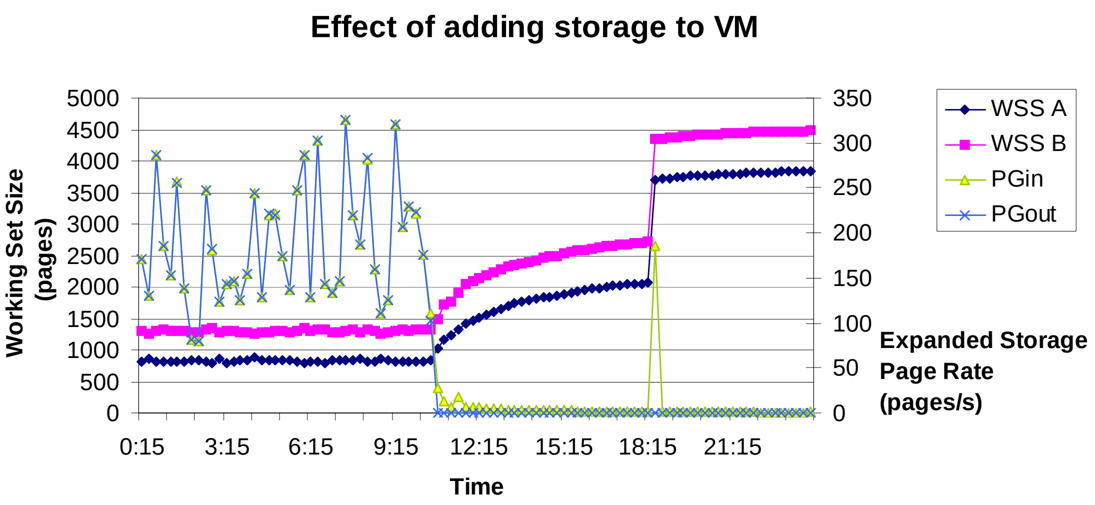
Footprint changes over time

- **z/VM will page-out only when necessary**
  - Minimal footprint can only be determined when there is enough memory pressure
  - Differences between virtual machines may not be representative
    - Compute average over a number of servers
    - Compare groups of servers



Unsolicited Experiment #1

Avoid work when you can: ~ 1500 pages

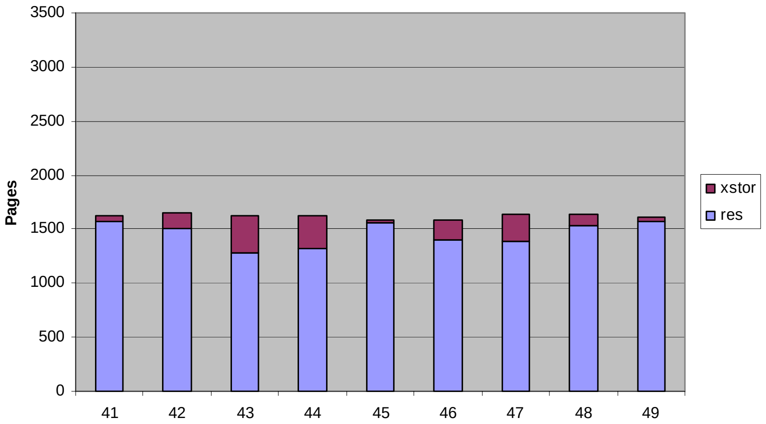


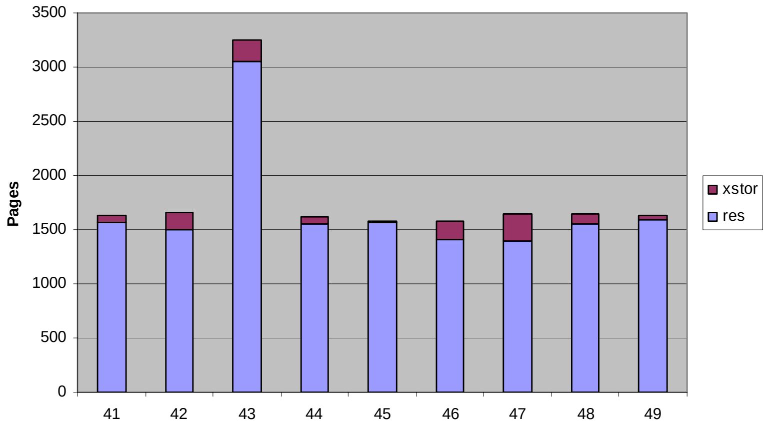
- **Linux Virtual machines grow as long as memory is available**
- **When memory constraint is taken away, Linux virtual machines grow again**

Measuring Linux footprint

- **Comparing a set of similar idle servers**
  - These servers appear to be happy with just 6 MB
- **Some activity in Linux will cause VM to page-in**

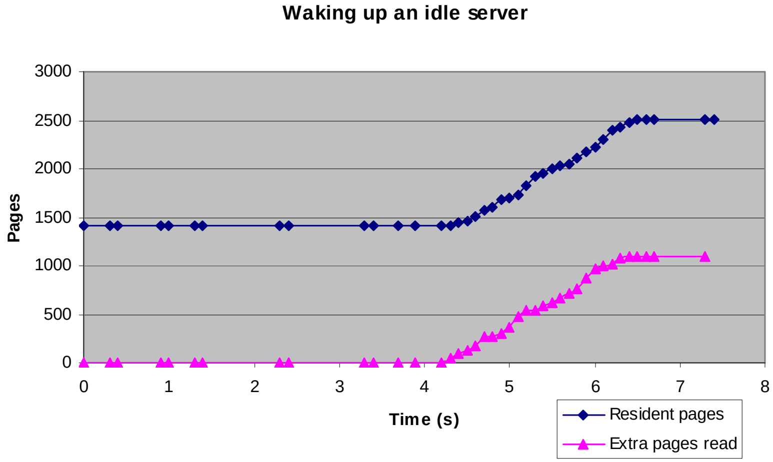
```
Connection rate: 10.1 conn/s (99.3 ms/conn, <=22 concurrent connections)
Connection time [ms]: min 1.0 avg 260.6 max 2144.9 median 1.5 stddev 554.1
Connection time [ms]: connect 9.2
```





Measuring Linux footprint

- **Paging in the missing pages is not for free**
  - Delay for the first transaction after idle period
  - Almost no “fresh” pages
    - Linux uses all memory
  - Wake-up delay depends on
    - Paging capacity available
    - Amount of pages needed

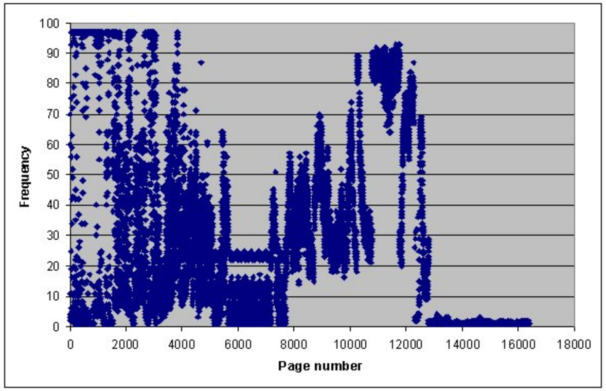


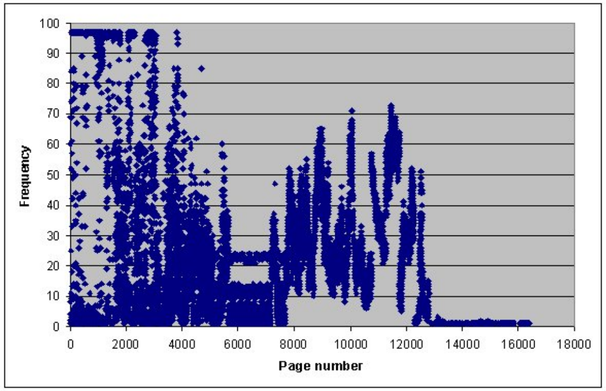
Linux Memory Usage

- **Over time Linux will use all memory**
  - Kernel code and core structures
  - Stack (process data)
  - Page Cache
    - Programs being executed
    - Shared libraries for programs
    - Disk files being used
    - Anything else that was used before
- **But what pages does it really need to run?**

Linux Memory Usage

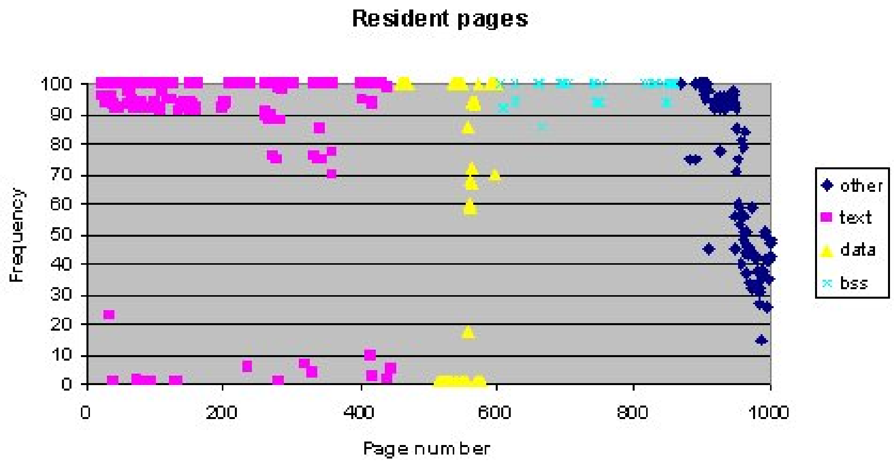
- **Look at which pages are kept resident on VM**
  - Walk the segment and page tables
  - With sufficient servers that should reveal a pattern
- **It does show a pattern**
  - But not very helpful





Linux Memory Usage

- **Linux implements virtual memory as well**
  - The same object resides in different “guest real” pages
    - The 2.4 kernel lacks mapping of real page to virtual
  - Very hard to compare different servers
    - Kernel and static allocations are fixed



Reducing Linux Footprint

- **Linux kernel in NSS**
  - Code has been around for some time
    - Moves code and data into different segments
  - Included in SuSE distribution
    - By default not enabled
    - Must compile your own kernel: not supported
  - Could save some 300-400 pages per Linux server

```
Filename=SUSE80T4  Filetype=NSS  Class=A  Spoolid=0139
Time loaded=6 11:07   Size=4M
Pages:  Main=379  Xstore=51  Dasd=430  Locked=0
Paging:
  Xstore:  Reads=1549       Writes=2838       Migrates=286
  Dasd:    Reads=1843       Writes=1400
```


Linux Kernel in NSS

Kernel code shared in NSS: ~400 pages

- Kernel is a relatively small portion of total code
  - Typically less than 2 MB (< 1% of the full system)
  - Kernel pages are popular
- Shared pages are less likely to be paged out by VM
  - Keeping kernel pages in is a Good Thing (pseudo page fault)

Drivers built-in: ~100 pages

```
                  <---CPU time--> <--------Main Storage (pages)--------->
         UserID   <(seconds)> T:V <Resident>  Lock <-----WSS----->
Time     /Class   Total  Virt Rat Total Activ  -ed Total Actv  Avg Resrvd
-------- -------- ----- ----- --- ----- ----- ---- ----- ---- ---- ------
17:08:27 System:   2.24  2.02 1.1  715K  715K 5315  711K 711K 2246      0
         LINUX-C   0.60  0.53 1.1  296K  296K 1000  295K 295K 2957      0
         *TheUsrs  0.55  0.49 1.1  149K  149K 3030  148K 148K 1399      0
         *Servers  0.41  0.39 1.1   592   592    4   999  999  100      0
         Linux-B   0.38  0.35 1.1  145K  145K  500  145K 145K 2903      0
         Linux-A   0.30  0.26 1.1  122K  122K  500  121K 121K 2434      0
```

Linux-B: IPL from DASD

Linux-A: IPL from NSS

Reducing Linux Footprint

- **Recent patches on IBM developerWorks**
  - May not be in your favorite distribution yet
- **Exploiting VM Shared Segments (DCSS)**
  - DCSS Block Device
  - The xip2 file system
- **Collaborative Memory Management**
  - Dynamically adjust Linux memory management through external controls

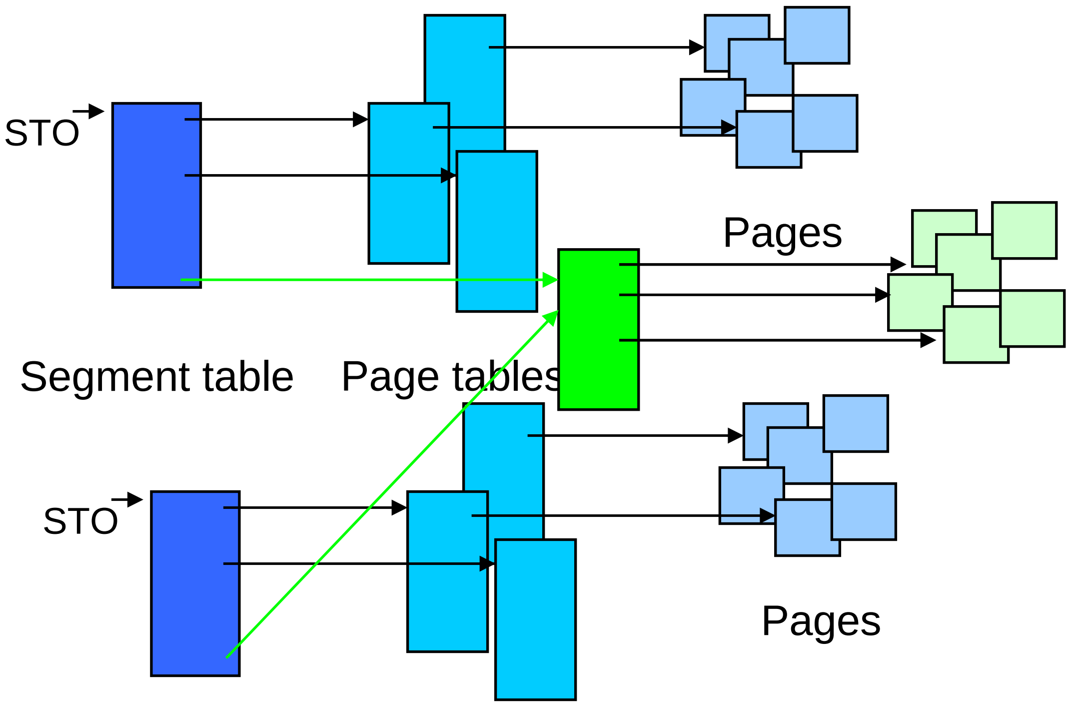
VM Shared Segments

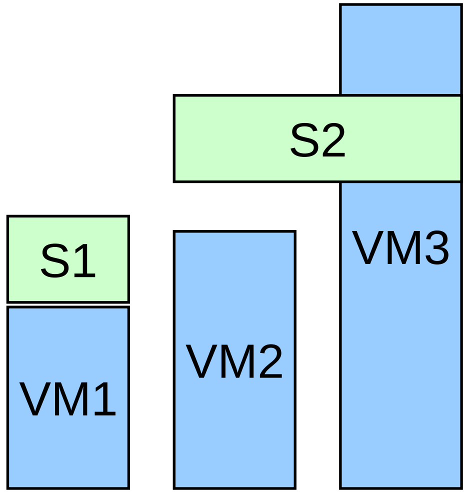
Facility to share storage between virtual machines

- Sharing is normally done read-only
- Exploited by CMS and Program Products
  - Small virtual machine size and small working set
  - Sub-second response times
- Exploited by GCS to simulate z/OS memory layout
- Exploited by Linux
  - Sharing the kernel code
  - Sharing application code (userspace binaries)
  - Very fast swap device
  - Shared R/W memory between servers

VM Shared Segments

- Page tables and pages shared by virtual machines
  - At the same virtual address for each virtual machine
- Virtual machine storage not necessarily contiguous





Virtual Machine memory layout

VM Shared Segments

- **Defined and saved by the systems programmer**
  - DEFSEG Define address range
  - SAVESEG Saves the contents of the segment
  - Updates through replacement of the complete segment
- **Contents is kept in special spool files**
  - SDF - System Data File
  - Referenced by name, Identified by number
- **Resides in paging subsystem while in use**
  - Paged preferred by z/VM to keep in memory

VM Shared Segments

- **NSS – Named Saved System**
  - Virtual machine uses one at a time
  - Is attached by IPL
  - Overlaps portions of the virtual machine
- **DCSS – Discontiguous Shared Segment**
  - Can use different segments at the same time
  - Attached through diagnose 64
  - Can overlap virtual machine or be above it

Linux and Shared Segments

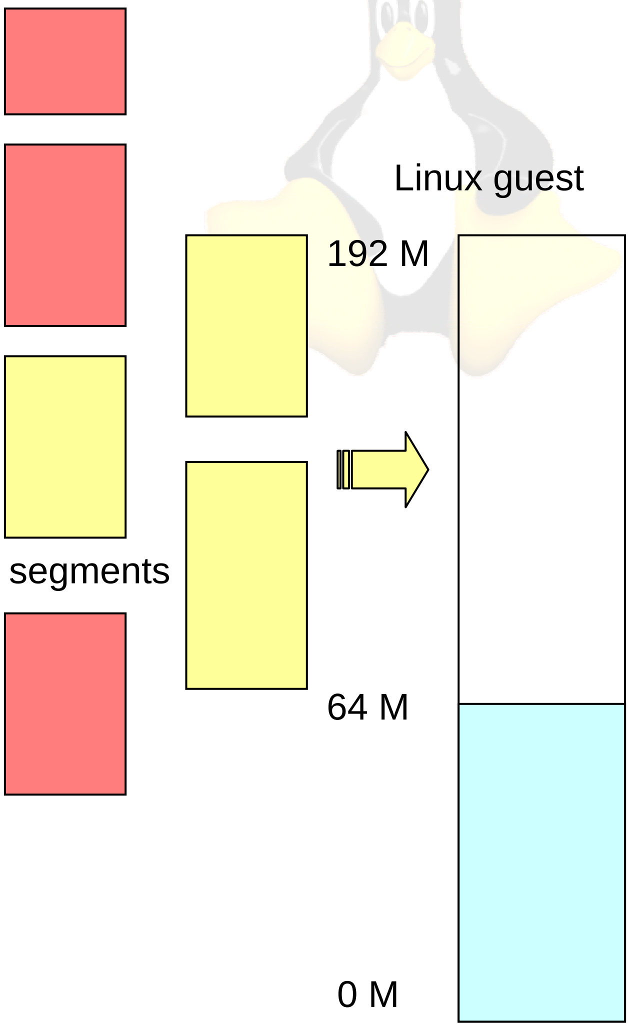
- **With recent patches Linux can attach DCSS**
  - Segment becomes part of Linux “real memory”
    - Memory management structures to map it
    - Structures are set up when Linux boots
    - Override memory size at boot time with `mem=`
  - Kernel can access contents of shared segments
    - Access is R/O or R/W depending on space type
    - Kernel can map pages into process address space

Patches currently on linuxvm.org

Expected to appear in SuSE soon

Linux and Shared Segments

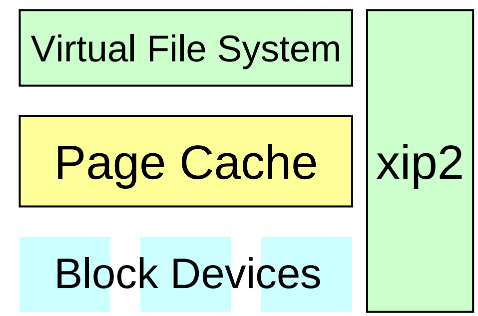
- **Override memory size in kernel command line**
  - Linux will still probe memory to see what it can use
  - Defines the amount of real address space that can be used
- **Segment to be used must be**
  - Completely outside virtual machine
  - Defined below the set maximum
  - Can not overlap




```
dasd=200-207 root=/dev/dasdb1 mem=192m
```

Linux and Shared Segments

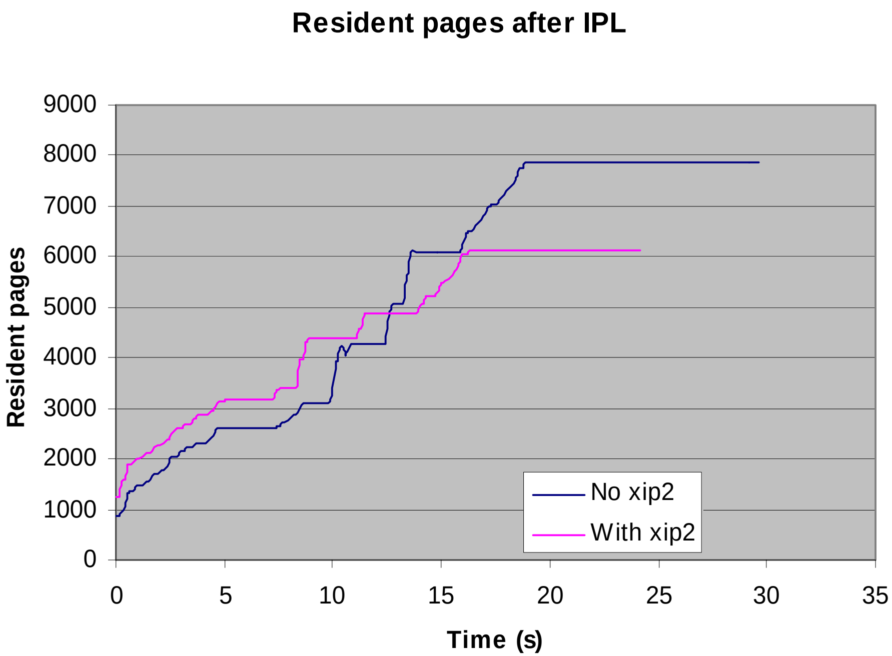
- **DCSS Block Device**
  - Like a disk in memory
  - Could be used to hold common code
    - But would be copied into private memory
    - Would only avoid the disk I/O
- **The xip2 file system**
  - Execute in Place
  - DCSS pages mapped into process address space
    - Avoids I/O and allows sharing



Benefits of xip2 file system

Booting server with xip2: ~1850 pages

- **Compare boot with and without xip2**
  - 64 MB virtual machine
  - 128 MB for xip2 file system
  - Savings because code is not loaded in memory
  - Faster booting



Benefits of xip2 file system

- **Less code to load into private memory**
  - Difference for sample configuration ~ 10 MB

```
linuxgw:~ # rsh linux10 `which free`
             total       used       free     shared    buffers     cached
Mem:         59052      28160      30892          0        424       9208
-/+ buffers/cache:      18528      40524
Swap:            0          0          0
linuxgw:~ # rsh linux60 `which free`
             total       used       free     shared    buffers     cached
Mem:         60596      38304      22292          0        772      19428
-/+ buffers/cache:      18104      42492
Swap:            0          0          0
```


Benefits of xip2 file system

Sample shows invocation of `tcsh id`

– Less I/O done

– Less memory used

```
linuxgw:~ # rsh linux10 `which vmstat` 10
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy  id wa
 0  0      0  30740    424   9244    0    0     0     5    0     4  0  0 100  0
 0  0      0  30712    424   9264    0    0     2     0    0    23  0  2  98  0
 0  0      0  30712    424   9264    0    0     0     0    0     5  0  0 100  0
linuxgw:~ # rsh linux60 `which vmstat` 10
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy  id wa
 0  0      0  21608    784  19920    0    0     0     0    0     4  0  0 100  0
 0  0      0  21224    788  20280    0    0    36     0    0    24  0  0 100  0
 0  0      0  21224    788  20280    0    0     0     0    0     5  0  0 100  0
```

Creating the xip2 file system

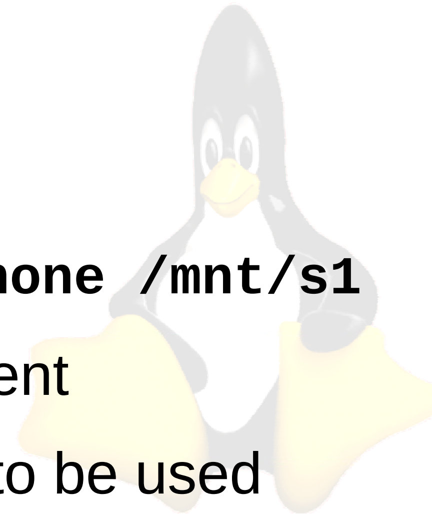
- **Using zipl**
  - Create image file and mount via the loop device
  - Populate with necessary files
  - Run zipl with new “segment” option to prepare disk
  - Use class E user to IPL disk
  - Issue SAVESEG command

- **Using DCSS block device**
  - Attach segment with “load non-shared” function call
  - Simply add and delete files as necessary
  - Have the block device driver issue SAVESEG

Requires CP class E for Linux

Performance considerations

Using the xip2 file system



```
mount -t xip2 -o ro,memarea=SUSE80S1 none /mnt/s1
```

- Invokes Diagnose 64 to attach shared segment
- The memarea option identifies the segment to be used
- Major benefit for files used through mmap() call
  - Executables (e.g. /bin, /sbin, /usr/bin, /usr/sbin)
  - Shared Libraries (e.g. /lib, /usr/lib)
- The xip2 files will be spread over the entire file system
  - Use the –bind option of mount

Redbook SG24-6824
Large Scale Linux
Deployment

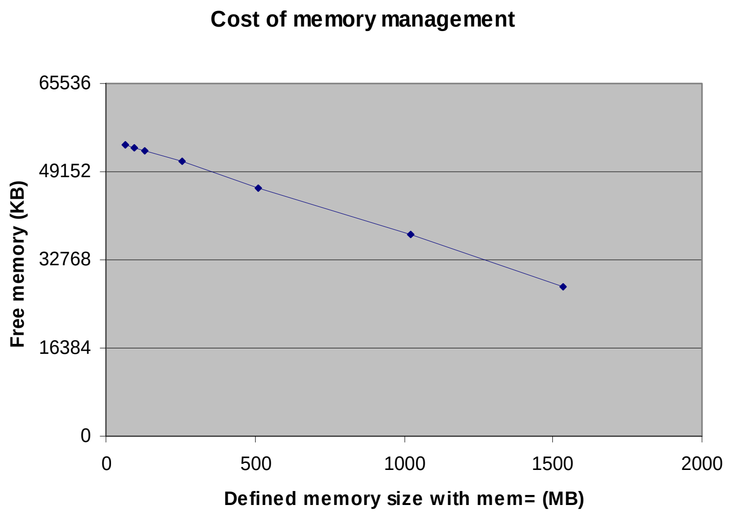
Sizing the xip2 file system

- **Define segments high enough**
  – Above all virtual machines that want to use it
- **Define segments low enough**
  – So that they fit under the mem= setting
- **Define segments large enough**
  – That way you can put more stuff in
- **Define segments small enough**
  – So they allow for large virtual machines

This needs tuning!

Sizing the xip2 file system

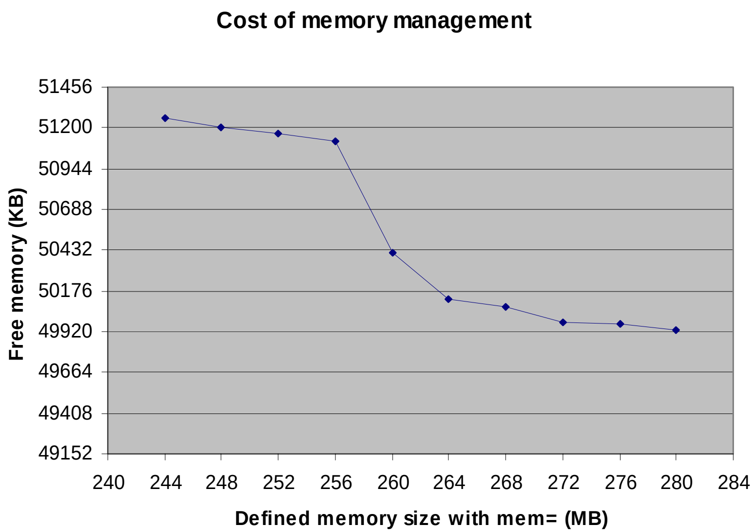
- **Set Linux memory size with mem=**
  - Attractive to set mem= very high to use many segments
  - But memory management structures take room too
  - Various tables are sized as part of total memory: about 20 MB / GB
    - 11 MB for struct page entries will mostly be paged out later



Sizing the xip2 file system

- **Funny "bump" in the curve**
  - Like switching gears
  - Probably not spent very well
    - Hash tables will be used less dense and increase footprint

Stick to "magic numbers" for the memory size: ~150 pages



Finding candidates for xip2

- **Not possible to put all software in**
- **Most benefit for access through mmap()**
  - Binaries, libraries, some data files
- **Attractive for very popular files used by many servers**
  - GNU C Runtime, application binaries and libraries
- **Effective for large files**
- **Selection is done on directory granularity**

Finding candidates for xip2

- **See maps entry in /proc to find files mapped**
  - Linux already handles sharing between processes
  - Files are loaded on demand

```
linux02:~ # ps -e | grep http
  537 ?        00:00:00 httpd
  538 ?        00:00:00 httpd
linux02:~ # head -n 10 /proc/537/maps
00400000-0043e000 r-xp 00000000 5e:09 75595      /usr/sbin/httpd
0043e000-00446000 rw-p 0003d000 5e:09 75595      /usr/sbin/httpd
00446000-0046f000 rwxp 00000000 00:00 0
40000000-40013000 r-xp 00000000 5e:05 3053       /lib/ld-2.2.5.so
40013000-40015000 rw-p 00012000 5e:05 3053       /lib/ld-2.2.5.so
40015000-40016000 rw-p 00000000 00:00 0
40018000-4001c000 r-xp 00000000 5e:09 105675     /usr/lib/libmm.so.12.0.21
4001c000-4001d000 rw-p 00003000 5e:09 105675     /usr/lib/libmm.so.12.0.21
4001d000-40088000 r-xp 00000000 5e:05 3063       /lib/libm.so.6
40088000-4008a000 rw-p 0006a000 5e:05 3063       /lib/libm.so.6
```


Boot script to mount xip2

```
#! /bin/sh -x

case "$1" in
  start)
        if [ -n "$dcss" ]; then
          dcss=`echo $dcss | tr [a-z] [A-Z]`
          mkdir /mnt/$dcss
          modprobe xip2fs
          mount -t xip2 -o ro,memarea=$dcss none /mnt/$dcss
          { while read tag path; do
              case "$tag" in
                "-") mount -n --bind /$path /mnt/$dcss/$path ;
                     fixup="$fixup $path" ;;
                "+") mount -n -r --bind /mnt/$dcss/$path /$path;;
              esac
            done
          } < /mnt/$dcss/$dcss.idx
          for path in $fixup ; do
            mount -n --bind /mnt/$dcss/$path /$path
          done
        fi ;;
  *) ;;
esac
```

```
+ lib
+ bin
+ sbin
+ usr/bin
+ usr/sbin
- usr/lib/locale
- usr/lib/perl5
+ usr/lib
```

Using xip2 file system

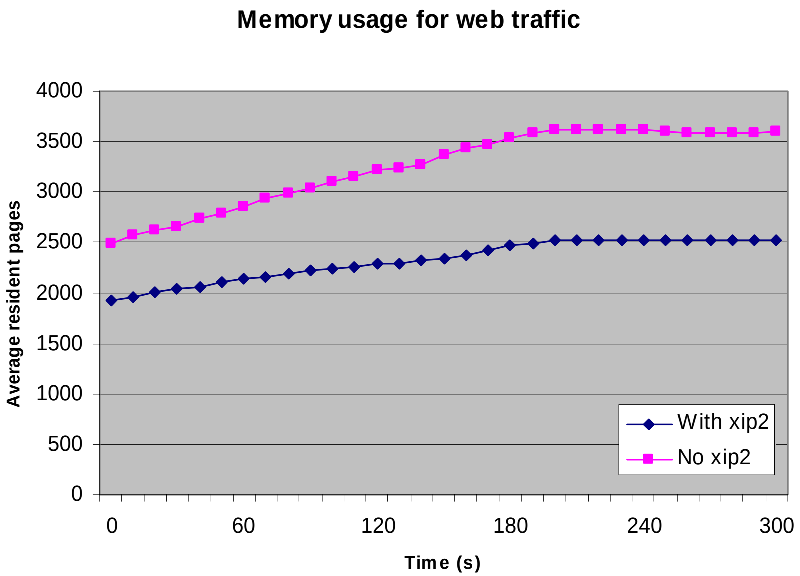
- **Comparing web servers**
  - z/VM slightly short on memory
  - Half of servers with xip2 and half without

Web server with xip2: ~1050 pages

Average initial delay:

With xip2: 1.5 s

No xip2: 2.6 s



Using the xip2 file system

- **Major issue is to be able to share code R/O**
  - Same issues as with shared R/O disk
    - No possibility to update code with servers running
    - Application code split in shared and non-shared portion
      Software maintenance nightmare
- **Various restrictions require proper planning**
  - Probably different segments for each class of server
    - Combine general segment and application specific segments
    - Could be combined with automount

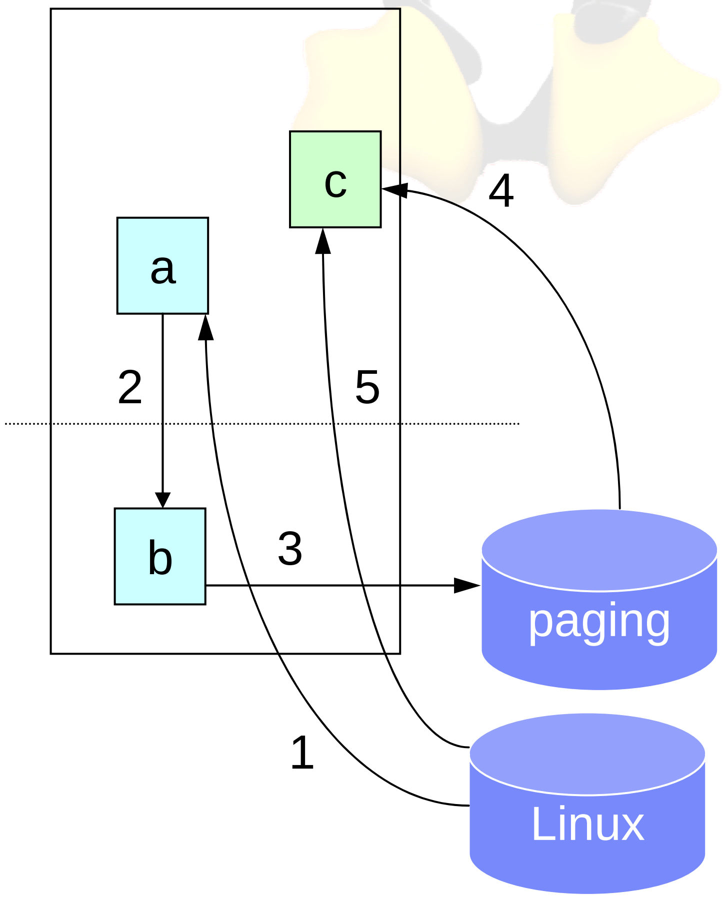
Collaborative Memory Management

- **Recent development**
  - – Patches on IBM developerWorks (January 2004)
- **Both Linux and z/VM do memory management**
  - – Local optimization versus global resource allocation
- **Linux CMM driver to reduce memory usage**
  - – Interface for VM to steer CMM driver

Page Cache Paging Problem

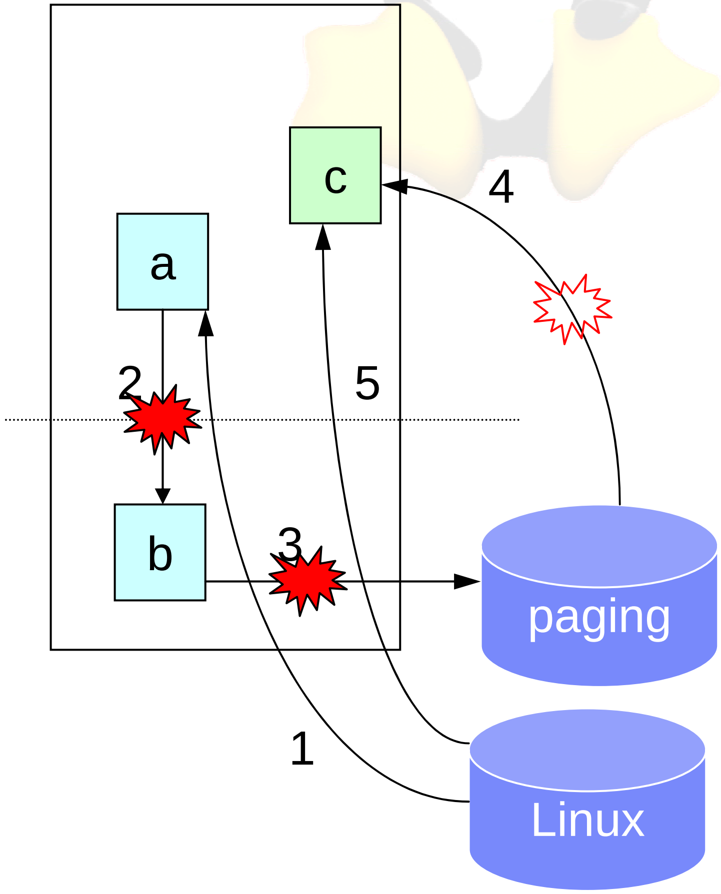
1. **Linux application wants to read, new page (a) is taken to read data in. Application continues to read new data in, in new fresh pages.**
2. **Working set grows and VM moves least recently used page to expanded memory page frame (b).**
3. **After more time without reference, contents of the page moves to paging volumes.**
4. **Application reads more data and Linux ran out of unused pages. Linux will use the oldest one which is paged out. The reference causes VM to allocate a frame (c) and page it in.**
5. **Old contents is disposed, and replaced by new data read by Linux into the page frame.**

- **Net result: local optimization to reduce I/O resulted in 3 I/O operations for one block of data.**



Collaborative Memory Management

- **Kernel thread that can take memory away from Linux and return to z/VM**
  - Triggered by z/VM memory management
  - Pages returned to Linux
    - trigger by external controls
    - timed release
- **Can shrink Linux footprint to avoid z/VM having to page it out to expanded (2)**
- **Can make Linux drop pages that were already out in expanded storage (3)**
- **Maybe also prevent page-in by z/VM when Linux frees the page (4)**



Collaborative Memory Management

- **Kernel thread allocates memory**
  - Least Recently Used pages are allocated
  - VM is told to drop the page via Diagnose 10
- **Kernel module cmm.o registers /proc variables**
  - cmm_pages — target reserved pages
  - cmm_timed_pages — reserved pages with timer
  - cmm_timeout — timer and pages to release

Collaborative Memory Management

- **Kernel module smsgiucv.o**
  - Set up of IUCV handler for SMSG
  - SET SMSG IUCV
- **Passes messages to cmm.o**
  - CMM SHRINK   Set the cmm_pages target
  - CMM RELEASE   Increase the cmm_timed_pages
  - CMM REUSE   Timed pages return plus interval

Collaborative Memory Management

- **Kernel thread allocates memory from Linux**
  - Initially it takes all unused memory (see **free**)
  - Beyond free memory it takes pages based on LRU
    - The real page could be in expanded storage or on disk
- **The cmm driver needs directives from "outside"**
  - Determine when to reserve memory
    - Most effective would be short after transaction end
  - Decide how much memory should be reserved
    - Probably requires knowledge about Linux utilization

Collaborative Memory Management

- **Permanent reservation**
  Reduce Linux footprint during a longer period
  – Front-end servers off-shift
  – Backup servers during office hours
- **Temporary reservation**
  Return memory to Linux short after reservation
  – Either “manually” or through cmm_timeout
  – Makes z/VM drop pages to prevent page-out

Collaborative Memory Management

- **Still early development**
  - We need to learn how this can be used
  - The current code has some rough edges
    - The SMSG interface may need authentication check
    - It’s easy to over commit and bring Linux down
    - SMSG and /proc interface are not consistent
  - Steering needs information from z/VM and Linux

DCSS Block Device for Swapping

- **Linux on z/VM can benefit from fast swap device**
  - Allows to make virtual machine size smaller
  - VDISK turns out to be a popular candidate
    - Especially using the diagnose driver instead of FBA
  - Look at it as memory, not disk

  - Diagnose driver requires CMS formatting before use

DCSS Block Device for Swapping

- **Define the DCSS as Exclusive Write**
  – Each virtual machine gets a private copy of the DCSS
- **Run mkswap before saving the DCSS**
- **Swap signature in first block only**
  – Save only one page, rest can be “Exclusive R/W not-saved”
  – Minimal cost when not in use, fast in setup
- **Swapping to DCSS happens without I/O**

```
q nss name suse80sw map
FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
0142 SUSE80SW DCSS       N/A    04000  04000   EW  A  00001   N/A       N/A
                                04001  04FFF   EN
```


DCSS Block Device for Swapping

- **Swapping to DCSS does not cause CP overhead**
  - The MVC is done in SIE when both pages are available
    - Results in low T:V Ratio
    - Swap rate measured 50% higher than with VDISK
    - Swapping still consumes CPU resources – avoiding is better
- **Fragmentation of swap space is still a concern**
  - Consider to implement a hierarchy of swap devices

```
                        <---CPU time--> <--------Main Storage (pages)--------->
         UserID   <(seconds)> T:V <Resident>  Lock <-----WSS----->
Time     /Class   Total  Virt Rat Total Activ  -ed Total Actv  Avg Resrvd
-------- -------- ----- ----- --- ----- ----- ---- ----- ---- ---- ------
04:30:27 LINUY03  40.56 25.72 1.6 13216 13216   10 13233  13K  13K      0
         LINUY02  40.48 40.42 1.0 15166 15166   10 15982  15K  15K      0
```


DCSS Block Device for Swapping

- **Restrictions and limitations**
  - Swap DCSS must be defined above virtual machine size
    - As low as possible to reduce cost of tables in Linux
      - Maybe define multiple segments at different starting address
    - Total of virtual machine size and DCSS is < 2G
  - Address space used for swap can not be used for xip2

Possible savings

- These are no absolute values
- Numbers are based on comparison between servers
- Not all savings can be achieved at the same time

- In this configuration allowed for running 300 servers rather than only 100.

Virtual NIC rather than dedicated OSA	**1820**
Avoid useless work where possible	**1500**
Shared kernel in NSS	**400**
Built-in drivers rather than modules	**100**
Idle server with xip2 file system	**1850**
Stick to “magic size” boundaries	**150**
Web server with xip2 file system	**1050**
	6870

Conclusion

- **Exciting new development**
  - Essential for running lots of Linux guests on z/VM
- **Execute in Place file system**
  - For general use the software management issues must be solved
- **Collaborative Memory Management**
  - Will need a lot of experimentation to get this tuned
- **Combination of xip2 and cmm has extra value**