KVM for z/VM Lovers

Christian Ehrhardt
IBM

5th August 2014
15752

#SHAREorg

SHARE in Pittsburgh 2014

Motivation

- There are a myriad of KVM introductions out there, so what is the point of another one?
  → KVM concepts explained based on known z/VM (6.3) components

Agenda

- CP interfaces
- Basic guest definition
- Virtualization basics
- Paging
- Dispatcher / Interruptibility
- Device virtualization

- Motivation - Better is not always the question
- Things not available in KVM
- Appendix
  - Disk setup options
  - Network setup options
  - CP Trace / TRSOURCE
  - Guest debugging with Trace

Terminology

- Things are sometimes called differently

System z	"Open world"
Storage	Memory
Direct access storage device (DASD)*	Storage, Disk
CPU, PU, IFL, SAP, ...	Processor, CPU
IPL	Boot
Central electronics complex (CEC)	Computer
Hipervisor	Hypervisor
...	

  - So there are plenty of terms that differ
    - Often it is also important who you ask even z/OS, z/VM and Linux on System z people talk differently
    - You will realize that often the difference is just that - terminology

*FCP doesn't count here as it came from the distributed systems anyway

Before diving into details – help me!

- Help me to find even more important areas that are missing
  - To create new charts helping us to understand each other better
  - Feel even more free than usual to ask about details, special options, ...

CP Interfaces

- z/VM
  - Host provides commands to query or change environment and configuration
  - One can use CMS for further functionality
- KVM
  - Host provides commands to query or change environment and configuration
  - Host is a full scale Linux system with all its tools

Basic guest definition

- z/VM
  - Eventually a guest is represented by a directory entry
    - In DirMaint or alternate solutions like VM:Secure
  - There is SMAPI as external interface
  - So the defintion of a guest could be done
    - Directly
    - Or by any interface exploiters of those like IBM Wave, xCAT or others
- KVM
  - Eventually a KVM guest is just an invocation of qemu with certain options
  - Core management is usually done by libvirt
  - So the “definition” of guests could be done any libvirt exploiter
    - Could be the command line based virsh with its xml files
    - Could also be driven by Openstack, virt-manager or others

Basic guest definition - example

- z/VM – directory entry

```
USER LINUX01 MYPASS 512M 1024M G
MACHINE ESA 2
IPL 190 PARM AUTOCR
CONSOLE 01F 3270 A
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
SPECIAL 500 QDIO 3 SYSTEM MYLAN
MDISK 191 3390 012 001 ONEBIT M
MDISK 200 3390 050 100 TWOBIT MR
```

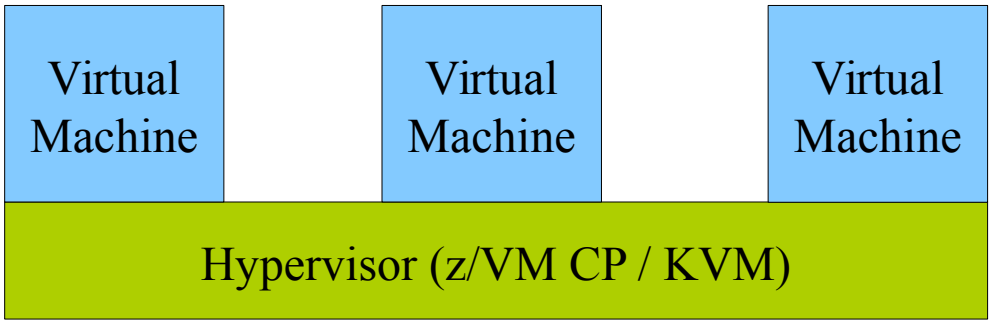
- KVM – virsh xml (shortened)

```
<domain type='kvm'>
  <name>LINUX01</name>
  <memory unit="MB">512</memory>
  <vcpu>32</vcpu>
  <os><type arch='s390x' machine='s390-ccw-virtio'>
  hvm </type></os>
  <devices>
    <console type='pty'>
      <target type='sclp'/>
    </console>
    <interface type='direct'>
      <mac address='de:ad:bb:12:35:01'/>
      <source dev='eth1' mode='bridge'/>
    </interface>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source dev='/dev/disk/by-path/ccw-0.0.c11a'/>
    </disk>
  </devices>
</domain>
```

Virtualization basics

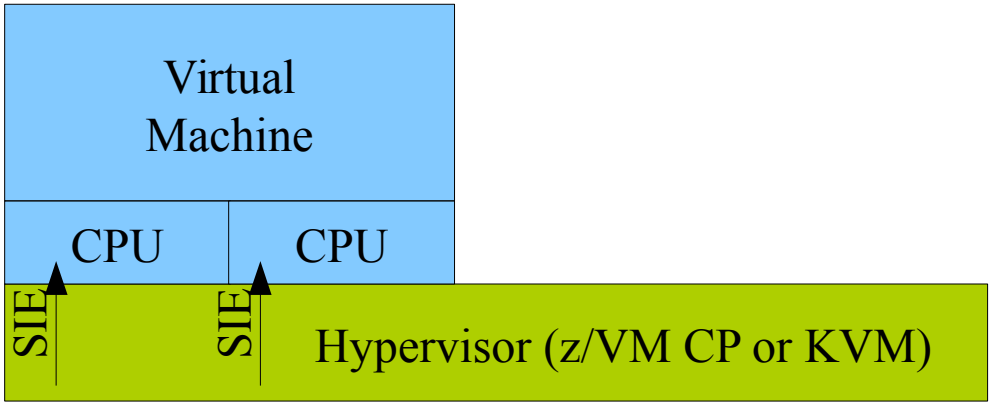
- z/VM
  - Replicates the z/Architecture Principles of Operations
  - Permits overcommitment of real hardware
  - Emulates or passes through z/Architecture I/O devices

- KVM/Linux
  - Replicates the z/Architecture Principles of Operations
  - Permits overcommitment of real hardware
  - Provides virtio based access to I/O devices



Virtualization basics

- Both
  - Use “Start Interpretive Execution” SIE instruction to “run” virtual processors
    - SIE uses a control block that describes the virtual processor state
    - SIE uses the Dynamic address translation (DAT) tables for the virtual machine
  - Hypervisor gets control back for various reasons
    - (Host) Page faults
    - I/O
    - Privileged instruction (including service calls like DIAGs)
  - Can kick another processor “out of SIE”



Paging

- z/VM
  - Historically there was demand paging (central ↔ expanded)
  - But since z/VM 6.3 there is only Block paging to disk left
    - Evacuation is done in groups of pages to keep IOPs down
    - Pages needed back from disk are also read in groups
  - Max 256 devices of 64GiB each

- KVM/Linux
  - Swapping is done on a per-page base
  - Page-out writes are done in so called page-clusters
    - The assumption is that pages aging together could be read together later as well
    - Avoid IOPs and increase efficiency
  - Page-in reads a dynamic amount of pages
    - A heuristic determines the success of reading ahead more than the faulting page
  - Max 30 devices, individual/overall limits out of reach
    - ~32 YiB per devices (by code limits), but I never found that device though

Paging – examples of attached paging space

- z/VM - 64 x 16GiB

```
vmcp "query alloc page"
                EXTENT      EXTENT  TOTAL  PAGES   HIGH    %
VOLID  RDEV      START         END  PAGES IN USE   PAGE USED
------ ---- ---------- ---------- ------ ------ ------ ----
102964 488F          8   16777214 16384K     29     63   1%
[...]
102901 4850          8   16777214 16384K     24     60   1%
                                 ------ ------        ----
SUMMARY                          1024M    628          1%
USABLE                           1024M    628          1%
```

Per disk usage info

- KVM/Linux – 16 x 256GiB

```
cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/dm-16                              partition       268434428       0       100
[...]
/dev/dm-29                              partition       268434428       0       100
# cat /proc/meminfo | grep -i swap
SwapCached:            0 kB
SwapTotal:     4288659392 kB
SwapFree:      4288659392 kB
```

Totals in the 4TiB range

Terminology – tricky parts

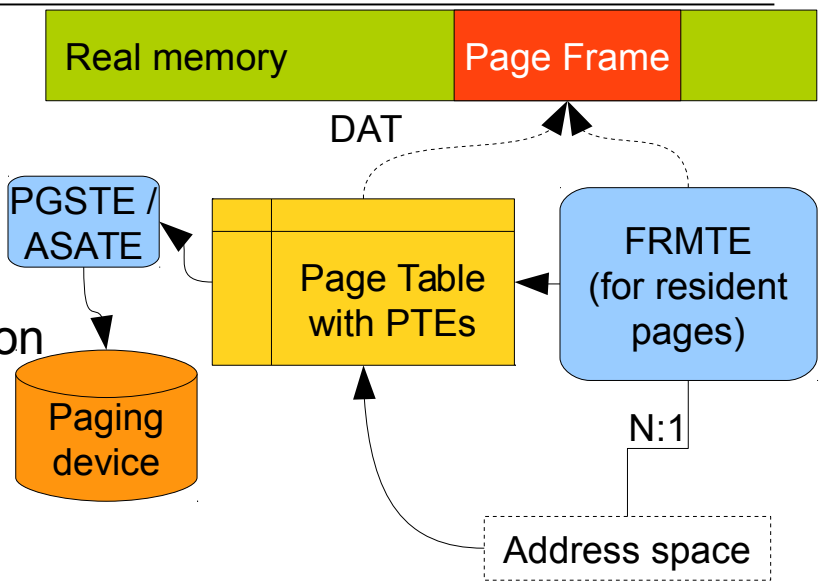
- Things are sometimes called the same, but they are not
  - be aware of this on the next pages about memory management
  - beg a pardon for me using these things interchangeably

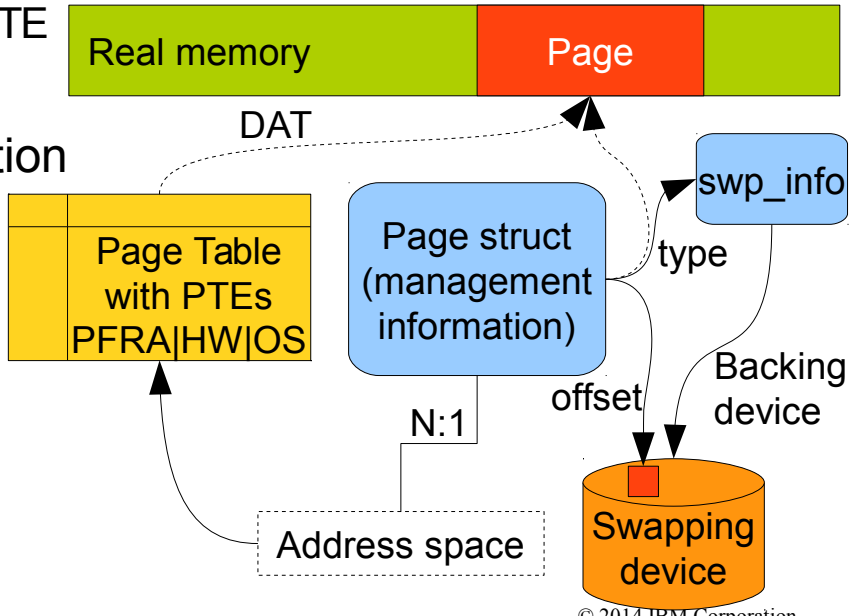
System z	"Open world"
(Page) frame	Page
Page Table	Page Table
FRMTE	Page struct often just called page too

*FCP doesn't count here as it came from the distributed systems anyway

Paging – a page

- z/VM
  - The page table holds the address resolution
  - For resident pages there will also be an FRMTE which maps real memory usage and holds extra information
  - Non resident pages have no FRMTE
    - From the PTE z/VM can reach the serialization information via PGSTE / ASATE
- KVM/Linux
  - The page table holds the address resolution
    - Also HW bits like “validity”
    - Also OS bits used to flag the swap device
  - A Page can be
    - In memory (anonymous)
    - On backing storage (swapped out)
    - On both (Swap cache)

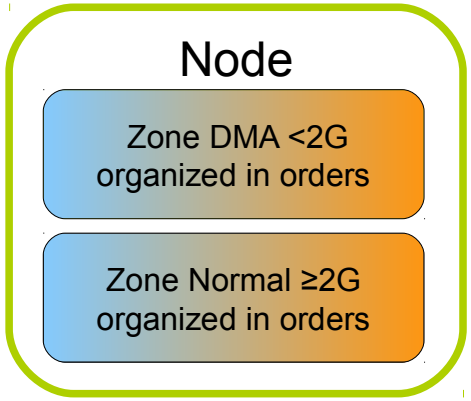




Paging – available list

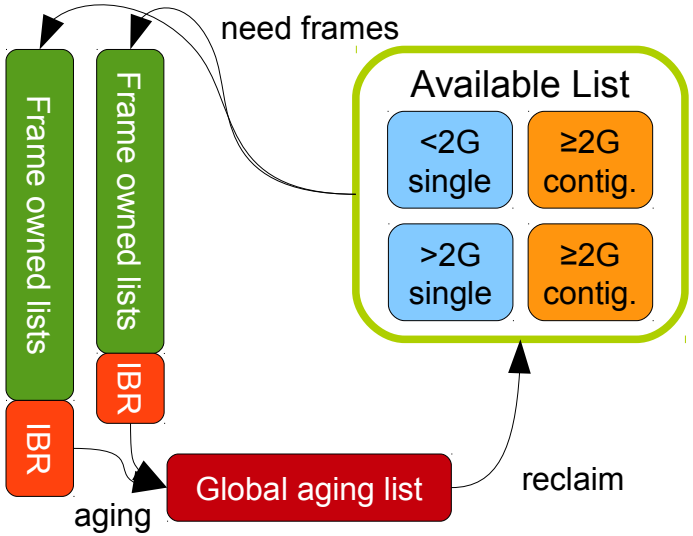
- z/VM
  - Free frames are on the available list
  - Holds certain categories
    - <2G, >2G, single or contiguous
    - Allocations are served from here

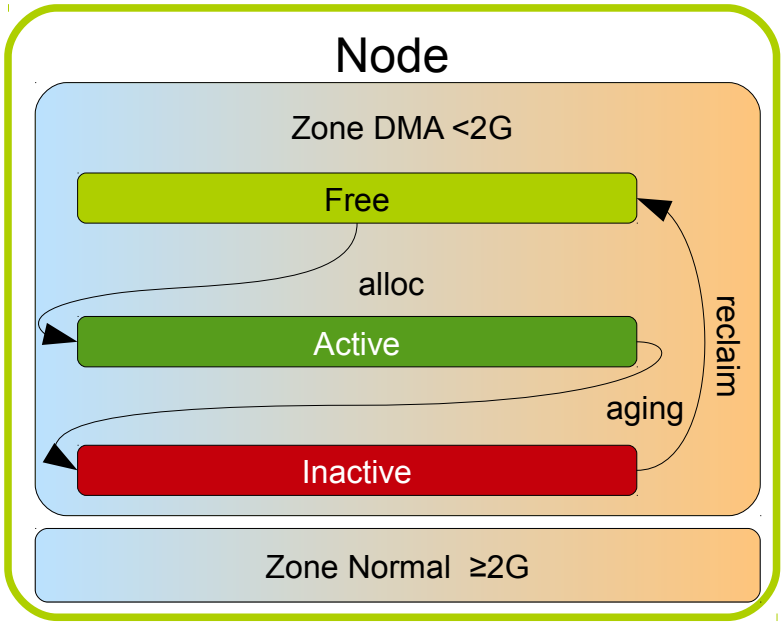
- KVM/Linux
  - Free pages are on the free list
  - Such a list is member of a zone
    - Zone DMA covers the 31bit range <2G
    - Zone Normal is ≥2G
    - Structured in “orders” for contiguity
      order 0 = 2^0 contiguous pages
      order 1 = 2^1 contiguous pages
      ...



Paging – page-out selection

- z/VM
  - Demand scan pushes frames from
    - Owned valid frames, down to
    - Owned “invalid but resident” frames (invalidation tracks access), down to
    - Global aging list
    - There happens page write
    - After being evacuated frame reclaim moves it over to the available list
- KVM/Linux
  - Reclaim code pushes pages down from
    - The active list, down to
    - The inactive list (invalidation to track access)
    - At the end of the inactive list swap-out and reclaim takes place





Paging – scans and position on disk

- z/VM
  - Optional prewriting allows eviction in case of demand
    - If that is not enough, demand scan kicks in
  - A page almost always goes back to its same DASD slot
  - A page not changed since last read from DASD is almost never rewritten
- KVM/Linux
  - Prewriting via watermarks once free pages drop below the high watermark
    - Below the low watermark allocating processes contribute time (direct reclaim)
  - A page “owns” its swap slot which never changes
  - There is no swap rewrite until a page is made dirty
  - Reclaim work at the end of the inactive list until it freed enough pages
    - Anon pages get swap slots assigned and written asynchronously
    - Dirty are pages written
    - Clean pages are discarded

Paging – some counters to compare

- z/VM

```
(from perf toolkit)
Main storage utilization:
 Total real storage          393'216MB
 Total available             393'216MB
 Offline storage frames            0
 SYSGEN storage size         393'216MB
 Shared storage               24'372KB
 FREE stor. subpools           5'612KB
 Subpool stor. utilization       88%
 Total DPA size              390'060MB
 Locked pages                  47986
 Reserved user storage             0KB
 Set reserved SYSMAX               0KB
 Trace table                   9'700KB
 Pageable                    389'863MB
 Storage utilization               0%
 Tasks waiting for a frame         0
 Tasks waiting for a page          0/s
 Standby real stor. size           0KB
 Reservd real stor. size           0KB
```

- KVM

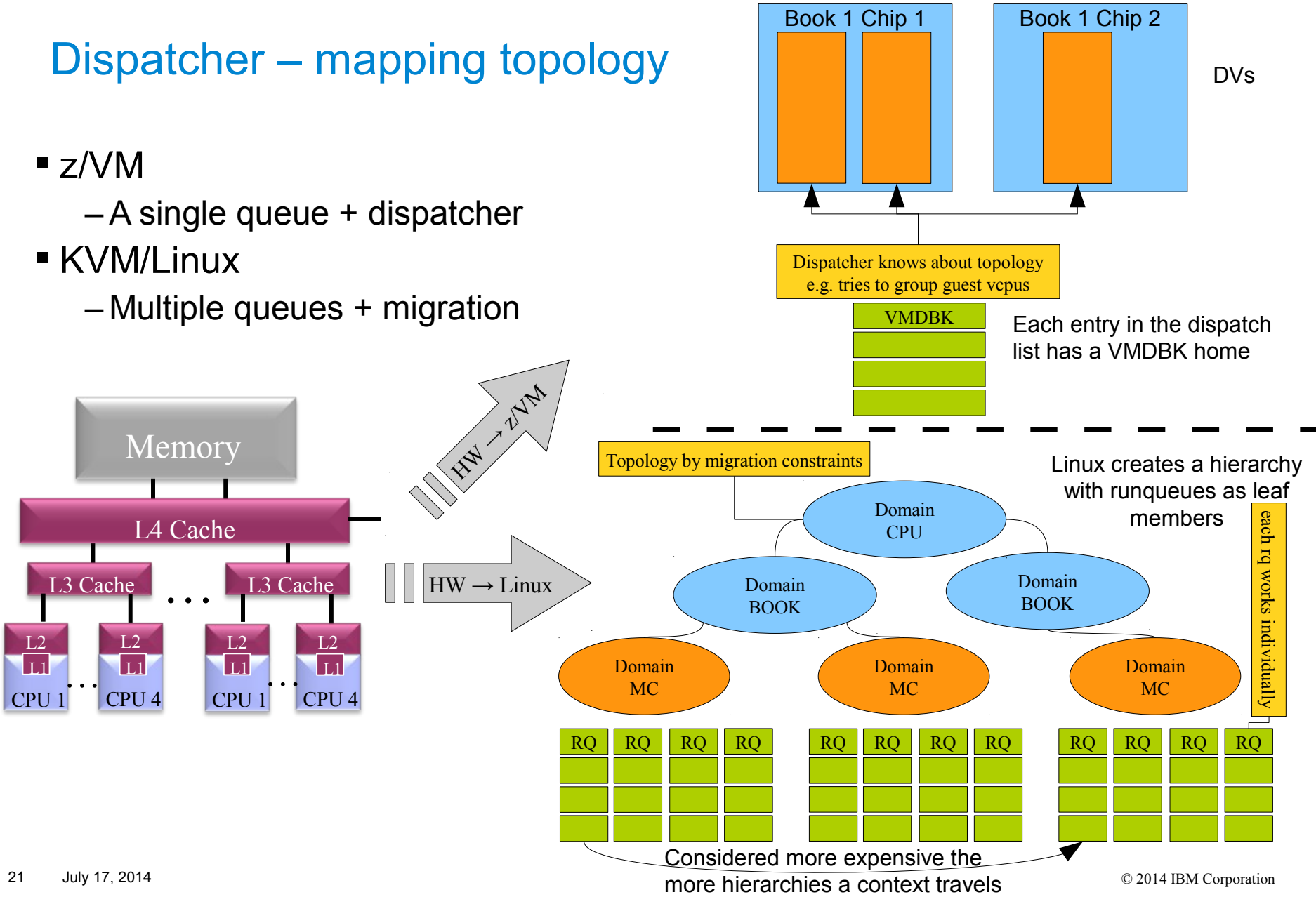
```
cat /proc/meminfo
MemTotal:        396209632 kB
MemFree:         389635216 kB
Cached:            204444 kB
SwapCached:             0 kB
Active:           3585476 kB
Inactive:          219756 kB
[…]
Mlocked:             3528 kB
SwapTotal:       4288659392 kB
SwapFree:        4288659392 kB
[…]
cat /proc/zoneinfo
Node 0, zone      DMA
  pages free     127285
        min      29
        low      36
        high     43
[...]
    nr_inactive_anon 1
    nr_active_anon 29
[...]
```

Dispatcher

- z/VM
  - Scheduler determines priorities based on shares and other factors
  - Dispatcher runs a virtual processor on a logical processor
  - Uses a central dispatch list
    - Queue of runnable VMDBKs kept in order by urgency
    - Tries to preserve VMDBK homes when assigning to dispatch vectors
    - Tries to keep VMDBKs of a guest topologically close
  - Single entity (dispatcher): good for central decisions
- KVM/Linux
  - Scheduler handles prioritization and dispatching of processes
  - Uses one runqueue per CPU
    - Holds task structs for each context (process/thread) ordered by priority/fairness
    - Processes are pulled to or migrated off a runqueue but they always belong to one
    - The more hierarchies a migration travels the more expensive it is considered to be
    - Tries to group related processes to simplify IPC
  - Spread entity (scheduler): good for scaling

Dispatcher – mapping topology

- z/VM
  - A single queue + dispatcher
- KVM/Linux
  - Multiple queues + migration



Interruptibility / Concurrency

- z/VM
  - Some tasks require a master cpu
  - Some core parts of z/VM run with interrupts disabled
    - Eases data handling

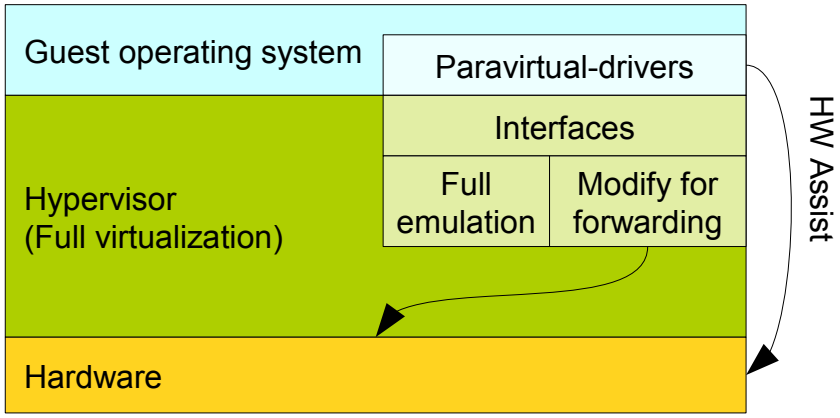
  → Optimize for efficiency

- KVM
  - No master cpu concept
  - Runs almost always with interrupts enabled
    - Explicit/Implicit preemption
    - Code needs to be safe against concurrency effects

  → Optimize for low latencies

Device virtualization – Basics

- z/VM
  - Utilizes hardware features to pass through devices and let the guest drive I/O without hypervisor exit
    - Initial setup still requires Host involvement
  - Trap instructions to emulate I/O
    - The architecture provides certain instructions (diags) to para-virtualize I/O
- KVM
  - Trap instructions to emulate I/O
    - The architecture provides certain instructions (diags) to para-virtualize I/O



Device virtualization – Enumeration

- z/VM
  - Devices enumerated via channel subsystem
  - Data transport either via start subchannel+ccw
  - Or data transport via qdio
    - Initial setup still via start subchannel
    - Several device types can use the same transport scheme
    - Support Adapter Interrupts for IRQ avoidance by grouping several devices
- KVM
  - Devices enumerated via channel subsystem
  - Device Communication via virtio
    - Initial setup still via start subchannel
    - Several device types can use the same transport scheme
    - There is no support for “classic” system z I/O devices

Device virtualization – Enumeration

- z/VM
  - Presents devices via channel subsystem
  - guest can sense id to get details
    - For example ControlType 1732/01 is a qdio based OSA Adapter
  - All kind of emulated and direct attached classic System z I/O devices

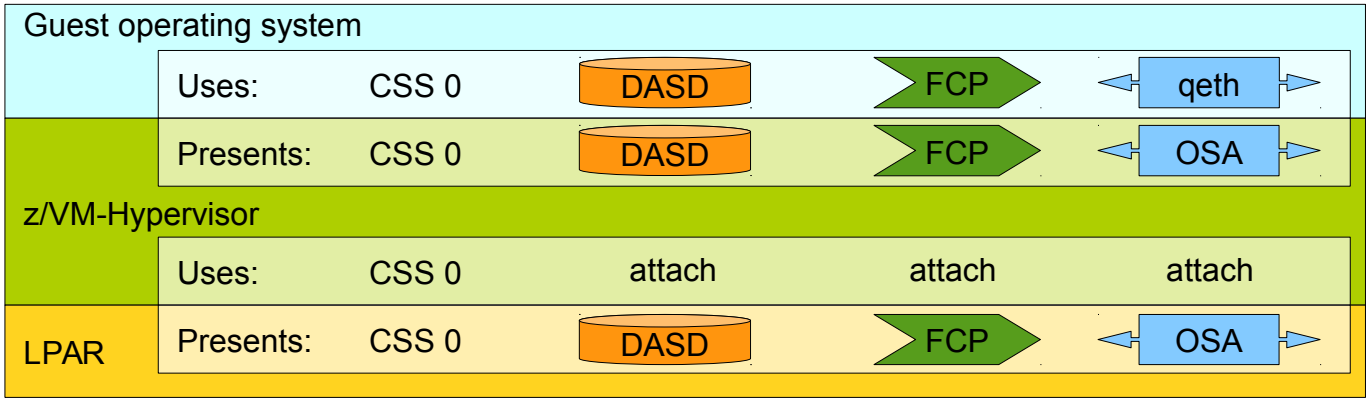
```
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.1000 0.0.0005  1732/01 1731/01 yes  80  80  ff   02000000 00000000
0.0.c116 0.0.000e  3390/0c 3990/e9 yes  ff  ff  ff   30313233 34353637
```

- KVM
  - Presents devices via channel subsystem
  - guest can sense id to get details
    - For example ControlType 3832/01 is a virtio-net based virtual network card
    - Note that only a virtual channel path with id 0 is supported

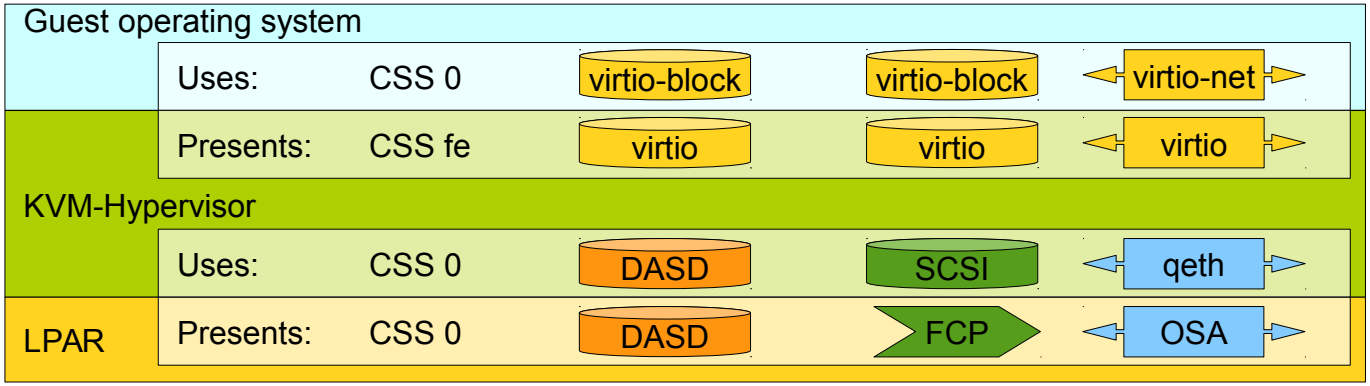
```
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0002 0.0.0000  0000/00 3832/02 yes  80  80  ff   00000000 00000000
0.0.0001 0.0.0002  0000/00 3832/01 yes  80  80  ff   00000000 00000000
```

Device virtualization – Enumeration

- z/VM



- KVM

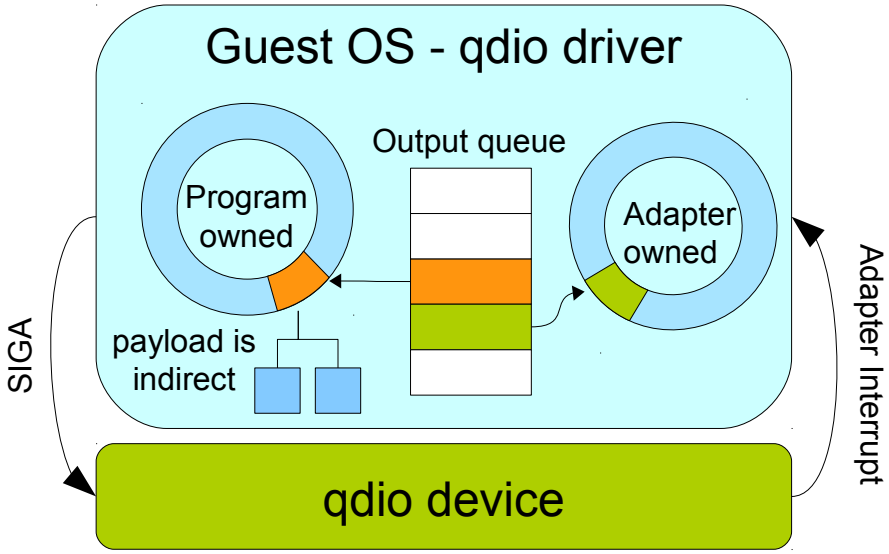


Device virtualization – Data transfer

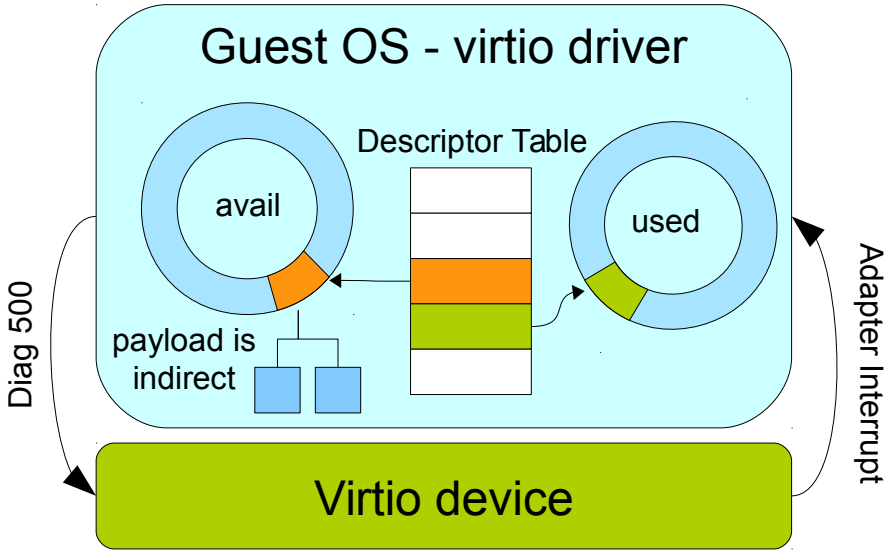
- z/VM
  - Subchannel based I/O
    - Submitting channel command words to be processed
  - QDIO based I/O
    - Initial setup via start subchannel
    - There are HW assists to avoid Hypervisor exits
    - Implements the Adapter interrupt architecture
- KVM
  - Provides neither classic subchannel+ccw nor qdio I/O devices
  - Virtio-transport based I/O
    - structural similarities to qdio
    - Initial setup via start subchannel
    - There are no HW assists for virtio, but several latency optimizations in KVM like data-plane, vhost-net, eventfd
    - Implements the Adapter interrupt architecture

Device virtualization – Data transfer (only one of many queues)

- z/VM (qdio)



- KVM (virtio)



Motivation – Better is not always the question

- Some things are just too special to be clearly better/worse
  - One needs to evaluate them for himself
  - Without all the context and details any comparison will be misleading at best

- And on top of that you just learned that they are not as different as you might have thought an hour ago

Things not available in KVM (yet)

- Well, some things just don't exist or are not mature enough yet
  - z/VM league weight management
  - Set Reserve
    - mlock too heavyweight and fixed on specific pages instead of an amount
    - Currently there is a lack of an easy page->process path
  - LDBUF style memory overcommit trashing prevention
  - Discontiguous Saved Segments
    - Although people are working on shared memory for KVM in general
  - Vertical CPU management
    - functionally available but no infrastructure like excess projection and management
  - System reset

- Again: Help me to find other areas important to you!
  - To extend the list above
  - To create new charts helping us to understand each other better

Q&A and optional parts

- Q&A
- Appendix
  - Disk setup options
  - Network setup options
  - CP Trace / TRSOURCE
  - Guest debugging with Trace

Questions

- Further information is at
  - Linux on System z – Tuning hints and tips
    http://www.ibm.com/developerworks/linux/linux390/perf/index.html
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

Christian Ehrhardt
Linux on System z
Performance Evaluation

Research & Development
Schönaicher Strasse 220
71032 Böblingen, Germany

ehrhardt@de.ibm.com

Paging – page-out selection

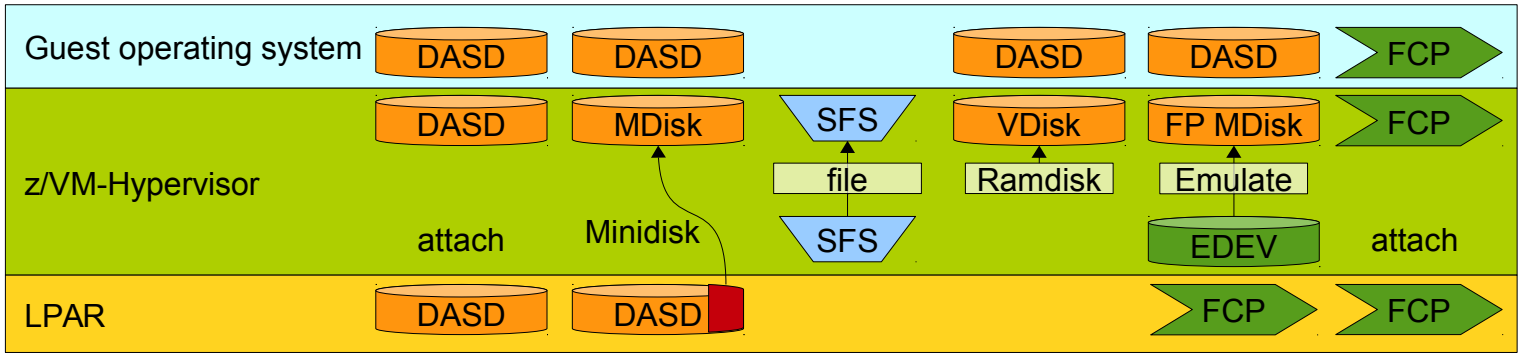
- KVM/Linux – extras that are good to know
  - As in the IBR, access revalidates pages
    - And puts them back to the active list
  - Pages are of two types
    - Anonymous pages which have no storage assigned
    - File backed (could be written to that file)
    - A page that got swap slot assigned is effectively file-backed (swap cache) pages can reside on disk and in memory
  - Page selection is more or less owner (process) agnostic
    - These (active / inactive) lists are per zone
  - Background Reclaim processes exist per Numa node (kswap)
  - Various other page or its mappings attributes are considered for swapping decisions
    - Prefer: Pages brought in by file reads start at the inactive list
    - Defer: Executable pages are less likely to be swapped
    - Balance: File pages are preferred before anon pages (swappiness tunable)

Device Virtualization – Disk options

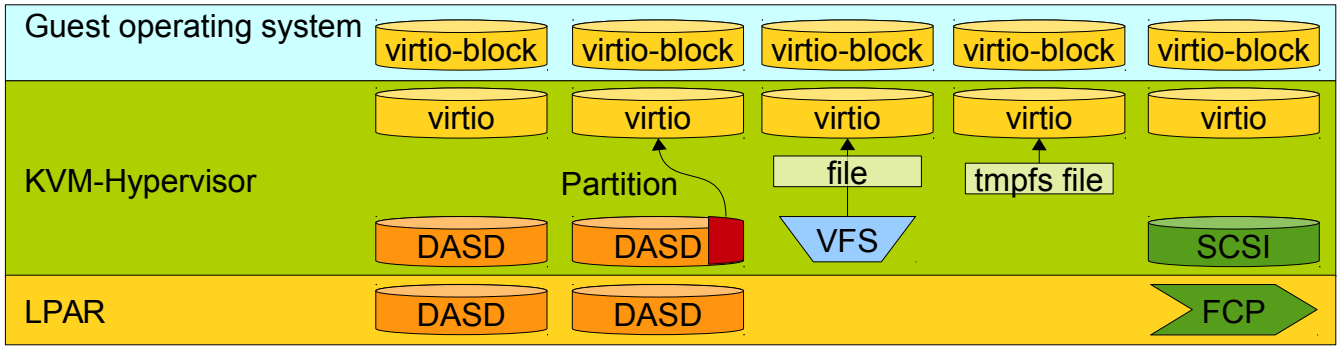
- z/VM
  - Can attach a full device as dasd
  - Can provide FCP devices via FBA emulation as dasd
  - Minidisks carve a device into logical subparts
    - Write-through cache for non dedicated disks
    - Option for fully virtual (only memory backed) minidisks
  - Filesystem based service via SFS (no Linux exploitation)
  - Can pass through FCP adapters
- KVM
  - Can pass any Host block device as virtio-block based to guest, such as
    - Full disks or their Partitions (max 3 with dasd, more with FCP)
    - LVM, any device mapper target, ...
  - Image Files are more flexible and also appear as virtio-block in the guest
    - The trade-off for this is extra overhead for the File system in the Host
    - Could be any File system, even NFS or cluster file systems
  - Caching could be used, but discouraged

Device Virtualization – Disk options

- z/VM



- KVM



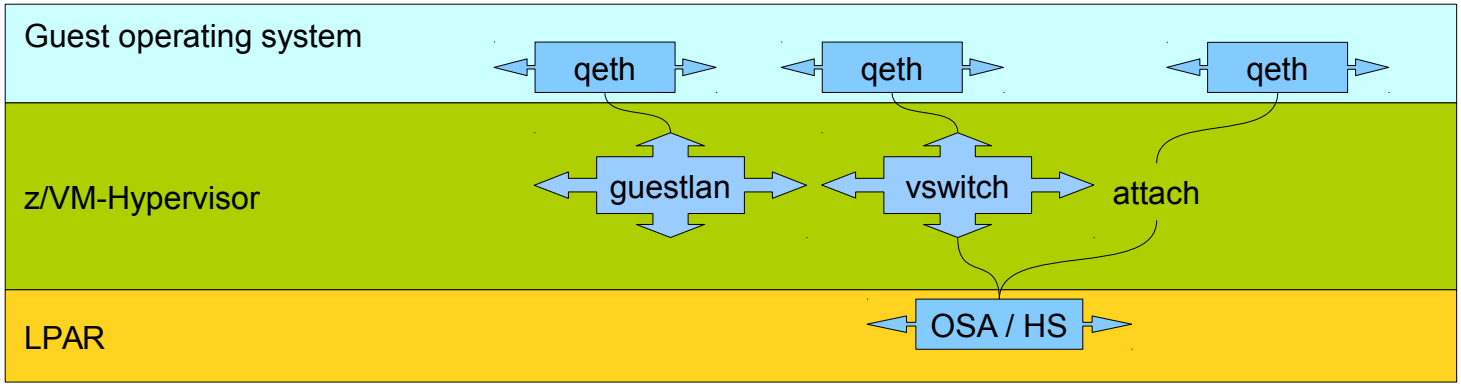
Device Virtualization – Networking options

- z/VM
  - Guest devices can be direct attached real devices
  - Virtual guest devices can be
    - Connected to a guestlan
    - Connected to a vswitch
  - Host real devices can be connected to a vswitch

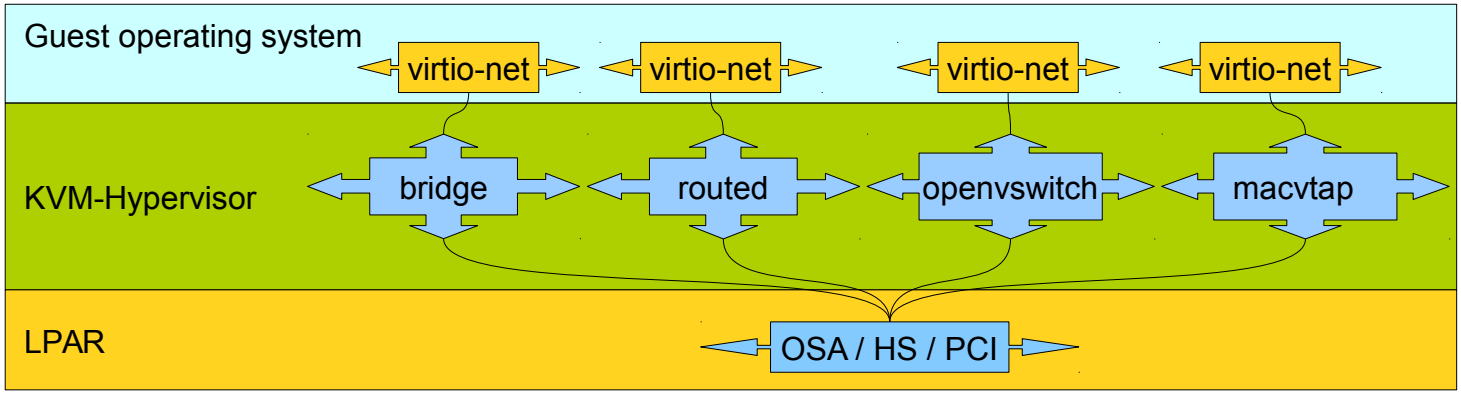
- KVM
  - Virtual guest devices can be
    - Associated with a host device via macvtap
    - Associated with a bridge in the Host
    - Routed in the Host (similar to L3 vswitch)
    - Associated with a openvswitch in the Host (Work in Progress)
  - Host real devices can be connected to a bridge or openvswitch

Device Virtualization – Networking options

- z/VM



- KVM



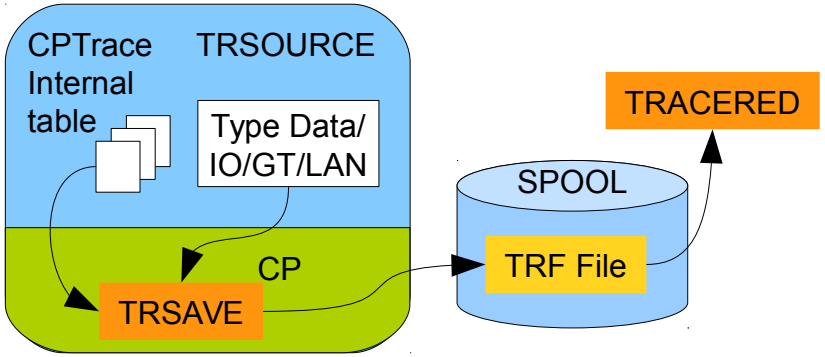
CPTrace / TRSOURCE

- z/VM
  - Nucleus maintains internal CP TRACE table
  - Can be trapped and saved to a TRF file with TRSAVE
    - Tracepoints provide a log of what happened
    - Data is captured in buffers
  - Can be enabled and disabled via SET CPTRACE – several default on
    - Due to that available in dumps for debugging purpose
  - Further trace types can be tapped via TRSOURCE

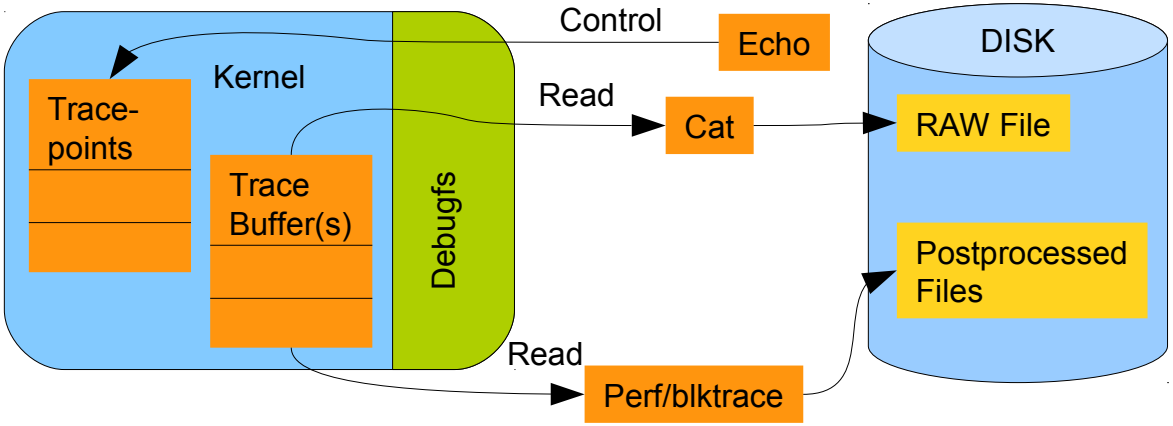
- KVM
  - Kernel provides a lot of Tracepoints
  - Can be controlled via virtual file system – default off
    - Sophisticated rewrite mechanism for max performance when disabled
    - Available in dumps for debugging purpose if enabled
  - A lot of on top evaluation tools like blktrace, perf sched, …
    - Profiling on any tracepoint

CPTrace / TRSOURCE - examples

- z/VM
  - Co-work of CPTrace, TRSOURCE, TRACERED depending on the case



- KVM
  - Tracepoints use internal wraparound buffers that can be tapped via debugfs



CPTrace / TRSOURCE – comparison example I

- z/VM – CPTrace

```
HCSTSM084I PROCESSING COMPLETE - 85 TRACE ENTRIES FORMATTED
-------------------------- 02/20/08 01:12:05.754151 --------------------------
8A00 CPU 0000 RUN USER (z/Arch)                        TIME 01:12:05.756698
     VMDMMODE   VMDBK ADDR   GUEST PSW
     82         01A39000     000C2000 810E99AC
B504 CPU 0000   INTERCEPTION, INSTRUCTION (z/Arch)     TIME 01:12:05.756698
     VMDGPR1    VMDBK        GUEST PSW
     07EF2668   01A39000     000C0000 810A5C54
     VMDMMODE   SIEICFLG     SIEINST SIEIPB
     82         80           B2F0    10000000
8A00 CPU 0000 RUN USER (z/Arch)                        TIME 01:12:05.756698
     VMDMMODE   VMDBK ADDR   GUEST PSW
     84         019C0000     00080000 80F22F50
```

- KVM – perf sched

```
    perf 12842 [000]   239.711465: sched_migrate_task: comm=awmtcpcl pid=12759 prio=120 orig_cpu=1 dest_cpu=3
    perf 12842 [000]   239.711555: sched_migrate_task: comm=awmtcpcl pid=12756 prio=120 orig_cpu=3 dest_cpu=1
 swapper     0 [001]   239.711734: sched_stat_wait: comm=awmtcpcl pid=12756 delay=0 [ns]
 swapper     0 [001]   239.711737: sched_switch: prev_comm=kworker/0:0 prev_pid=0 prev_prio=120 prev_state=R ==>
next_comm=awmtcpcl next_pid=12756 next_prio=120
    perf 12842 [000]   239.711739: sched_migrate_task: comm=awmtcpcl pid=12761 prio=120 orig_cpu=2 dest_cpu=3
awmtcpcl 12756 [001]   239.711746: sched_stat_wait: comm=awmtcpcl pid=12755 delay=7640 [ns]
```

CPTrace / TRSOURCE – comparison example I

- KVM – CPU context switch and migration mapping

```
D0   .   M0 *B0          240.248844 secs
D0   .   M0 *P0          240.248858 secs P0 => kworker/3:1:41
  *.   .   M0  P0        240.248860 secs
   .   .   M0 *.         240.248865 secs
```

- KVM – Latency overview

```
Task                  |   Runtime ms  | Switches | Average delay ms | Maximum delay ms | Maximum delay at   |
-----------------------------------------------------------------------------------------------------------------
:12842                |    982.372 ms |       67 | avg:    0.097 ms | max:    0.851 ms | max at: 241.389740 s
:3                    |      0.413 ms |        7 | avg:    0.056 ms | max:    0.376 ms | max at: 239.779265 s
:12763                |    424.249 ms |    15546 | avg:    0.034 ms | max:   90.719 ms | max at: 242.495018 s
```

CPTrace / TRSOURCE – comparison example II

- z/VM – I/O Trace

```
HCSTSM084I PROCESSING COMPLETE - 23 TRACE ENTRIES FORMATTED
TRACE TYPE IO, CPU 0001              TIME 14:16:49.766668
      TRACEID = IOTRACE, TRACESET = NULL, IODATA = 200
      USER = MAINT, I/O OLD PSW = 07041000 80000000 00000000 00131BB6
      DEVICE = 0A00, SCSW = 00C04007 1F59DCE0 0C000000, ESW = 00800000
      I/O PRIORITIES: CHANNEL = 0, CURRENT = 0, ORIGINAL = 0, OUT-PRIORITIZED COUNT = 0
   -> CCW(1) = 63400010 1F59A878, CCW ADDRESS = 1F59DCC8
      DATA = 00C00000 00000000 01FF0000 01FF0000 *................*
   -> CCW(2) = 47400010 1F59DDF8, CCW ADDRESS = 1F59DCD0
      DATA = 06000001 01FF0000 00000000 03FF0000 *................*
```

- KVM – blktrace / blkparse

```
94,8    0       32     0.000018062  5371  A   R 65824 + 8 <- (94,9) 65632
94,9    0       33     0.000018281  5371  Q   R 65824 + 8 [dd]
94,9    0       34     0.000018812  5371  G   R 65824 + 8 [dd]
94,9    0       36     0.000020000  5371  I   R 65824 + 8 (    1188) [dd]
94,9    0       37     0.000020250  5371  U   N [dd] 2
94,9    0       39     0.000022750  5371  D   R 65824 + 8 (    2750) [dd]
94,9    1        1     0.000526186     0  C   R 65728 + 80 (  505530) [0]
```

CPTrace / TRSOURCE – comparison example II

- KVM – blktrace / blkparse postprocessing
  - Statistical breakdowns per application / per cpu
  - Can be used to replay I/O with fio

```
logchecker (0)
 Reads Queued:           0,        0KiB  Writes Queued:           0,        0KiB
 Read Dispatches:        2,    1,008KiB  Write Dispatches:        0,        0KiB
 Reads Requeued:         0               Writes Requeued:         0
 Reads Completed:        9,    4,100KiB  Writes Completed:        0,        0KiB
 Read Merges:            0,        0KiB  Write Merges:            0,        0KiB
 IO unplugs:             0               Timer unplugs:           0
 Allocation wait:        0               Allocation wait:         0
 Dispatch wait:          0               Dispatch wait:           0
 Completion wait:        0               Completion wait:         0

CPU0 (dasdc1):
 Reads Queued:          18,    4,100KiB  Writes Queued:           0,        0KiB
 Read Dispatches:        8,    3,340KiB  Write Dispatches:        0,        0KiB
[...]
```

Debugging

- z/VM
  - Creates traps via TRACE on data, addresses, mem writes
    - Can also trap on instructions
    - Can include expressions to check conditions
  - Can show / modify memory & registers
  - Can control execution by SKIP/PASS/STOP
  - Can search memory with LOCATEVM (Host global search)
  - Trace points can be used to trigger other commands
- KVM
  - GDB can break on data, addresses, mem writes, source line
    - Can also interrupt on key press
    - can include expressions to check conditions
  - Can show / modify memory & registers
  - Can navigate through execution with skip/continue/step
  - Can search memory with find
  - Can trigger further non interactive actions

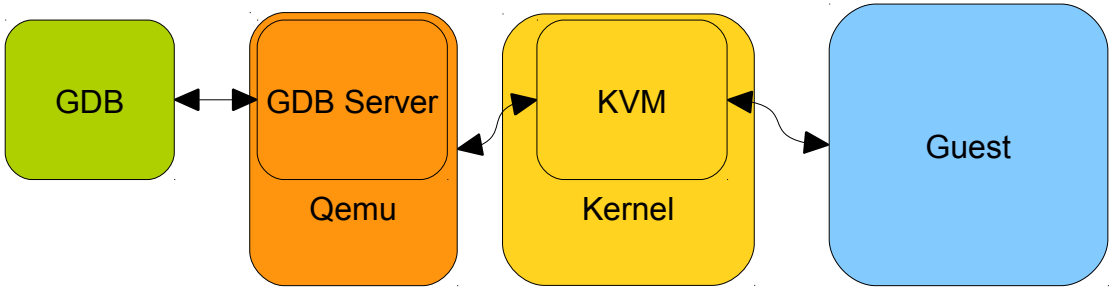
Debugging

- z/VM
  - TRACE... is the major frontend
  - Based on Program Event Recording

- KVM
  - Qemu provides a GDB server (option `-s`)
    - GDB attaches to qemu and can be augmented by debug and source info

```
gdb <image> -ex "target remote localhost:1234" -tui -d <path>
```

  - Provides basics to allow exploitation with all the features of gdb
    - SW breakpoints alter the guest (=guest visible, but fast)
    - HW breakpoints use the Program-Event-Recording (invisible, but can slow down)



Debugging - example

- z/VM

```
CPU ALL TRACE INST PSWA 663054
D T100.10
R00000100  00000000 00000000 00000000 00000000 06 *................*
STORE U100 DEAD BEEF
Store complete.
D T100.10
R00000100  C4C5C1C4 40C2C5C5 C6000000 00000000 06 *DEAD BEEF.......*
```

- KVM

```
b *0x663054
Breakpoint 4 at 0x663054: file arch/s390/kernel/entry64.S, line 643.
x /8 0x100
0x000100:       0x1f    0x8b    0x08    0x00    0x70    0xd8    0x9a    0x53
0x000008:       0x02    0x03    0xec    0x7d    0x77    0x40    0x53    0x49
set {int}0x000100 = 0xDEADBEEF
x /8 0x100
0x000100:       0xde    0xad    0xbe    0xef    0x70    0xd8    0x9a    0x53
0x000008:       0x02    0x03    0xec    0x7d    0x77    0x40    0x53    0x49
```

Debugging - example

- z/VM
  - Can print and convert strings
  - Can print structured z/VM data and access subfields
    - Via block in dumps, would need an auxiliary CMS setup print structures in traces
  - Way more functions to go into detail …
- KVM
  - Can print strings and has EBCDIC conversion if needed
  - Features like function and struct resolution can ease a lot of tasks
    - Can work on anything that has debuginfo
    - All working on life systems
  - Seems to have more functions than I can grasp …

Debugging

- z/VM

```
BLOCK PFXPG 0
[…]
+0058 PFXEXTNP        000C0000 803259E0
EXTERNAL NEW PSW
+0060 PFXSVCNP        000C0000 806247F8
SVC NEW PSW
+0068 PFXPRGNP        000C0000 8052A490
PROGRAM NEW PSW
```

- KVM

```
(gdb) set print pretty on
(gdb)  p /x (struct _lowcore)*0x0000
[…]
external_new_psw = {
    mask = 0x404c00180000000,
    addr = 0x662ef0
  },
  svc_new_psw = {
    mask = 0x704c00180000000,
    addr = 0x6628c4
  },
  program_new_psw = {
    mask = 0x404c00180000000,
    addr = 0x662b0c
  },
```

Page In/Out – some counters to compare

- z/VM

```
Paging / spooling activity:
Page moves <2GB for trans.    ..../s
Fast path page-in rate        ..../s
Long path page-in rate        ..../s
Long path page-out rate       ..../s
Page read rate                   0/s
Page write rate                  0/s
Change page rewrites             0/s
[...]
Paging SSCH rate                 0/s
SPOOL read rate                  0/s
SPOOL write rate                 0/s

Agelist:
Target size                     0KB
Actual size                     0KB
Revalidation rate                0/s
Reval post-write rate            0/s
Steal rate                       0/s
Pages Evaluated                  0%
Writes rate                      0/s
```

- KVM

```
[root@p10lp35 ~]# cat /proc/vmstat
nr_free_pages 98418244
nr_inactive_anon 1003
nr_active_anon 5176
[...]
nr_dirty 27
[...]
pswpin 0
pswpout 0
[...]
pgactivate 17942
pgdeactivate 0
[...]
pgscan_kswapd_dma 0
pgscan_kswapd_normal 0
[...]
```

Contiguous Pages – some counters to compare

- z/VM

```
FCX294      CPU 2827  SER EAA24  Interval 14:04:53 - 14:20:53    Perf. Monitor

          <------------ Storage -------------> <--Times--> <Frame Thresh>
Interval  <Available> <Requests/s> <Returns/s> <-Empty/s-> <- Singles -->
End Time   Sing  Cont  Sing   Cont  Sing  Cont  Sing  Cont   Low     Prot
>>Mean>>    19K  438M  1071  82296 11187 2055K    .0    .0    27       27
14:17:53    80K 1860M 17135  1286K  175K   32M    .0    .0   115      115
14:18:53    80K 1860M    .0     .0    .0    .0    .0    .0   115      115
```

- KVM

```
[root@p10lp35 ~]# cat /proc/pagetypeinfo
Page block order: 8
Pages per block:  256
Free pages count per migrate type at order     0      1      2      3      4      5      6      7      8
Node    0, zone      DMA, type    Unmovable    7      3      7      2      1      0      0      0      0
Node    0, zone      DMA, type  Reclaimable    0      0      0      0      1      1      1      0      0
[...] more types and zones
```