KVM Forum
14th October 2014
James Hogan
james.hogan@imgtec.com
Overview

- Trap & Emulate
  - Virtual Address Space
  - Trap & Replace

- MIPS VZ
  - TLB Management
  - TLB Critical Sections

- Future work
Trap & Emulate (T&E)

- Run guest OS in user mode
- Existing hardware (no VZ, EVA, KScratch registers, etc)
- MIPS instruction set well suited
  - Sensitive instructions not exposed to user mode
  - Coprocessor 0 (privileged) instructions cause traps
  - Emulated by KVM
- Modified guest kernel
- By Kyma Systems, for MIPS Technologies
- Upstream in QEMU v2.1[1], Linux v3.10[2]
Traditional MIPS32 Virtual Address Space

0xffffffff: Kernel TLB Mapped, 512M
0xe00000000: Kernel/Supervisor TLB Mapped, 512M
0xc00000000: Kernel Unmapped Uncached, 512M
0xa00000000: Kernel Unmapped Cached, 512M
0x800000000: User TLB Mapped, 2G

Linux: vmalloc / modules
Linux: MMIO
Linux: static code & data
T&E Guest Mode
Virtual Address Space

```
0xffffffff  Kernel TLB Mapped  512M
0xe0000000   Kernel/Supervisor TLB Mapped  512M
0xc0000000   Kernel Unmapped Uncached  512M
0xa0000000   Kernel Unmapped Cached  512M
0x80000000   Guest Kernel TLB Mapped  512M
0x60000000   Guest Kernel TLB Mapped  512M
0x40000000   Guest User TLB Mapped  1G
0x00000000
```

- Linux: `vmalloc / modules`
- Linux: MMIO
- Linux: static code & data
- Guest Linux: `vmalloc / modules`
- Guest Linux: static code & data
Trap & Replace

• Replace trapping guest instruction
• mfc0/mtc0 (read/write control registers)
  – Many CP0 registers RO/RW, no immediate side effects
  – Replace with load/store
  – Map page at 0x00000000 while in guest kernel
  – Hard wired zero register for base
    \[
    \text{mtc0} \quad rt, \text{reg} \rightarrow \text{sw} \quad rt, (\text{reg} \times 4)(\text{zero}) \\
    \text{mfc0} \quad rt, \text{reg} \rightarrow \text{lw} \quad rt, (\text{reg} \times 4)(\text{zero})
    \]
MIPS VZ

- MIPS r5 architecture extension for hardware assisted virtualization
  - Guest CP0 state, guest mode
  - Minimum of traps to hypervisor
  - Virtualized guest physical memory
  - Runs unmodified guest OS
- VZ hardware (MIPS, Cavium, Broadcom)
- KVM ports
  - Sanjay Lal (Kyma) posted May 2013\(^3\)
  - David Daney (Cavium) posted June 2013\(^4\)
Normal TLB Management

Hardware

Virtual Address (VA)

TLB
VA -> PA

TLB Hit

Physical Address (PA)

Kernel Software

TLB Exception

Page Table
VA -> PA

Page Fault

TLB Miss
tlbwr
T&E TLB Management

**Hardware**
- Guest Virtual Address (GVA)
  - TLB GVA -> PA
    - TLB Hit
    - TLB Miss
    - Physical Address (PA)

**Kernel Software (KVM)**
- TLB Exception
  - Guest TLB GVA -> GPA
    - TLB Hit
    - guest_pmap GPA -> PA
    - tlbwr

**Guest Kernel Software**
- Guest TLB Exception
  - Guest Page Table GVA -> GPA
  - Page Fault
VZ TLB Management

Hardware

Guest Virtual Address (GVA)

Guest TLB
GVA -> GPA

TLB Hit

Guest Physical Address (GPA)

TLB GPA -> PA

TLB Miss

Physical Address (PA)

Kernel Software (KVM)

TLB Exception
guest_pmap
GPA -> PA
tlbwr

Guest Kernel Software

Guest TLB Exception

TLB Miss

tlbwr

Guest Page Table
GVA -> GPA

Page Fault
VZ War Story: Shrinking Pages

- Multiple guests soaking with crashme
- One guest eventually locks up
  - Guest page size (CP0_PageMask) reset to 4KB
  - Infinitely writes 4KB instead of 16KB page mapping
- Guest mode change: check CP0_PageMask
- PDTrace: capture control flow around change
PDTrace Analysis

Guest 1

lw a1, 0x14(t0)

mtc0 at, CP0_KScratch0

TLB mapping invalid: Guest TLB Invalid Exception

Guest 1 Register State

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>t0</td>
<td>0x0123C000</td>
</tr>
<tr>
<td>CP0_BadVAddr</td>
<td>0x0123C014</td>
</tr>
<tr>
<td>CP0_PageMask</td>
<td>0x0FFFF9000 (16K)</td>
</tr>
</tbody>
</table>

Guest TLB Entries

<table>
<thead>
<tr>
<th>Index</th>
<th>GuestID</th>
<th>GVA</th>
<th>GPA0</th>
<th>GPA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>1</td>
<td>0x01238xxx</td>
<td>0x08228xxx</td>
<td>invalid</td>
</tr>
</tbody>
</table>
PDTrace Analysis

Guest 1

lw a1, 0x14(t0)

Pre-emption: Guest 2 runs

mtc0 at, CP0_KScratch0

... TLB Write Random: Replaces Guest 1's TLB Entry

Guest 2

... tlbwr ...

Guest TLB Entries

<table>
<thead>
<tr>
<th>Index</th>
<th>GuestID</th>
<th>GVA</th>
<th>GPA0</th>
<th>GPA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>2</td>
<td>0x3FF80xxx</td>
<td>0x12BC8xxx</td>
<td>0x13BF0xxx</td>
</tr>
</tbody>
</table>
PDTrace Analysis

Guest 1 Register State

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP0_BadVAddr</td>
<td>0x0123C014</td>
</tr>
<tr>
<td>CP0_Index</td>
<td>0xFFFFFFFF</td>
</tr>
<tr>
<td>CP0_PageMask</td>
<td>0x0FFF9000 (16K)</td>
</tr>
</tbody>
</table>

Guest TLB Entries

<table>
<thead>
<tr>
<th>Index</th>
<th>GuestID</th>
<th>GVA</th>
<th>GPA0</th>
<th>GPA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>2</td>
<td>0x3FF80xxx</td>
<td>0x12BC8xxx</td>
<td>0x13BF0xxx</td>
</tr>
</tbody>
</table>
**PDTrace Analysis**

**Guest 1**

```
srl    k0, k0, 12
...

tlbps
andi   at, k0, 0x1
beqz   at, 0x803604a4
andi   at, k0, 0x80
beqz   at, 0x8036046c
nop

tlbr
```

**TLB Probe result (CP0_Index) not checked**

**Guest 1 Register State**

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP0_BadVAddr</td>
<td>0x0123C014</td>
</tr>
<tr>
<td>CP0_Index</td>
<td>0xFFFFFFFF</td>
</tr>
<tr>
<td>CP0_PageMask</td>
<td>0x00000000 (4K)</td>
</tr>
</tbody>
</table>

**TLB Read:** TLB registers reset to invalid

**Guest TLB Entries**

<table>
<thead>
<tr>
<th>GVA</th>
<th>GPA0</th>
<th>GPA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3FF80xxx</td>
<td>0x12BC8xxx</td>
<td>0x13BF0xxx</td>
</tr>
</tbody>
</table>

**Guest 2**

```
...tlbwr...
...```
TLB Critical Sections

• Context switch must preserve critical TLB entry
  – Detect based on exception level, exception cause
  – Preserve TLB entry matching CP0_BadVAddr

• Trap & Emulate
  – Guest TLB stored in memory, not as volatile
  – Still affects savevm/loadvm/migration
    • Harder to hit
TLB Critical Sections

- Code assuming TLB entry exists/preserved
  - TLB Invalid exception (valid bit clear)
  - TLB Modified exception (write disallowed)
  - TLB Read/Execute Inhibit exception (read/execute disallowed)
  - Potentially anywhere CP0_Index points to valid entry (interrupts disabled)
    - Between TLB probe (tlbp) and TLB read (tlbr)
Future Work

• General
  – Expose FPU, MSA etc to guest
  – SMP

• Trap & Emulate
  – Further optimisation & fixes

• VZ
  – Unify implementations
  – Upstream
  – Device assignment
    • IOMMU
    • MIPS GIC & IRQ pass through
References

- Qemu:
  1. [v5] “Qemu: KVM Support for MIPS32 Processors”

- KVM:
  2. [v2] “KVM for MIPS32 Processors”