MIPS

by Imagination

KVM on MIPS

KVM Forum

14th October 2014

James Hogan

james.hogan@imgtec.com

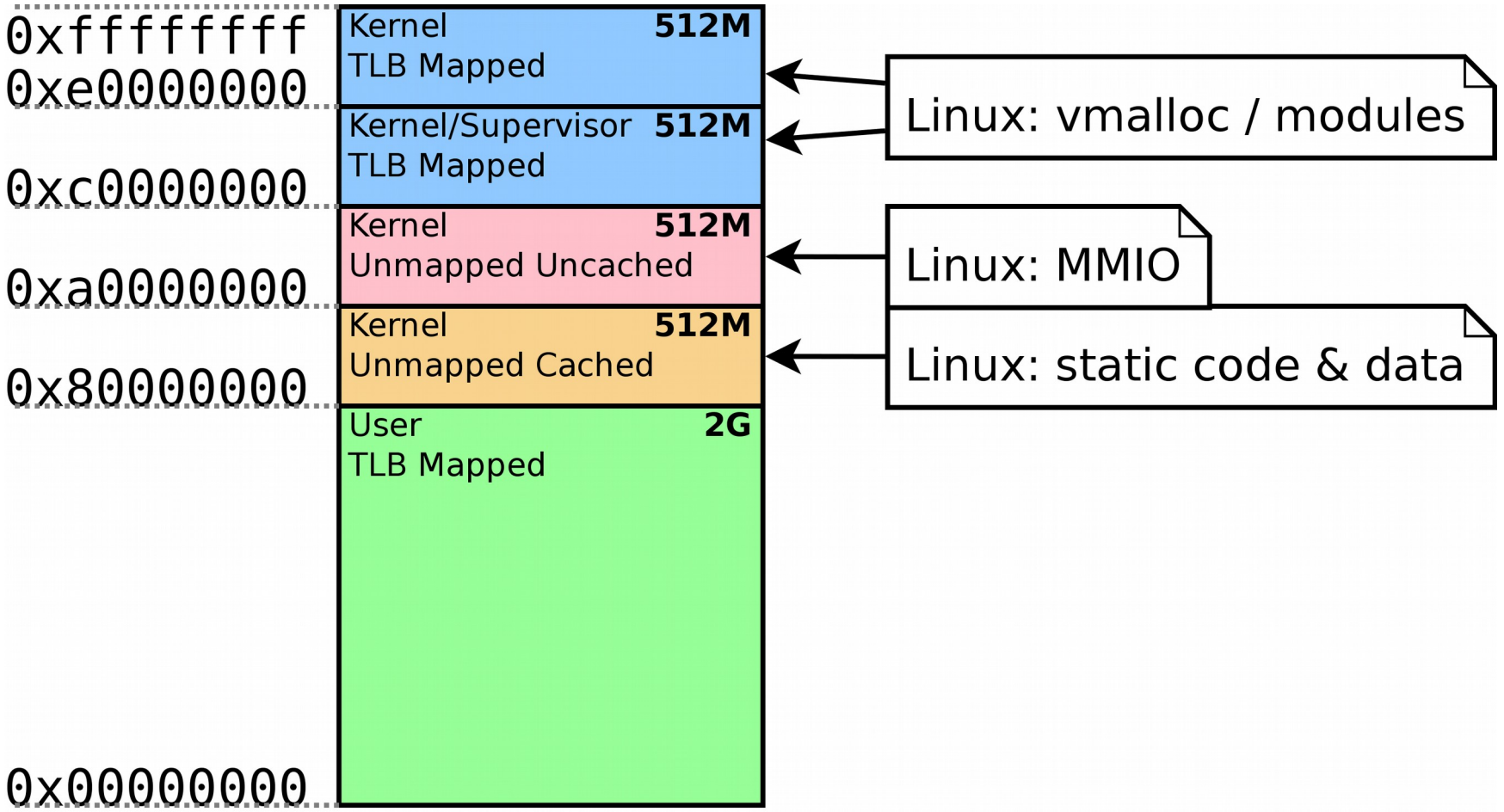
Overview

- Trap & Emulate
  - Virtual Address Space
  - Trap & Replace
- MIPS VZ
  - TLB Management
  - TLB Critical Sections
- Future work

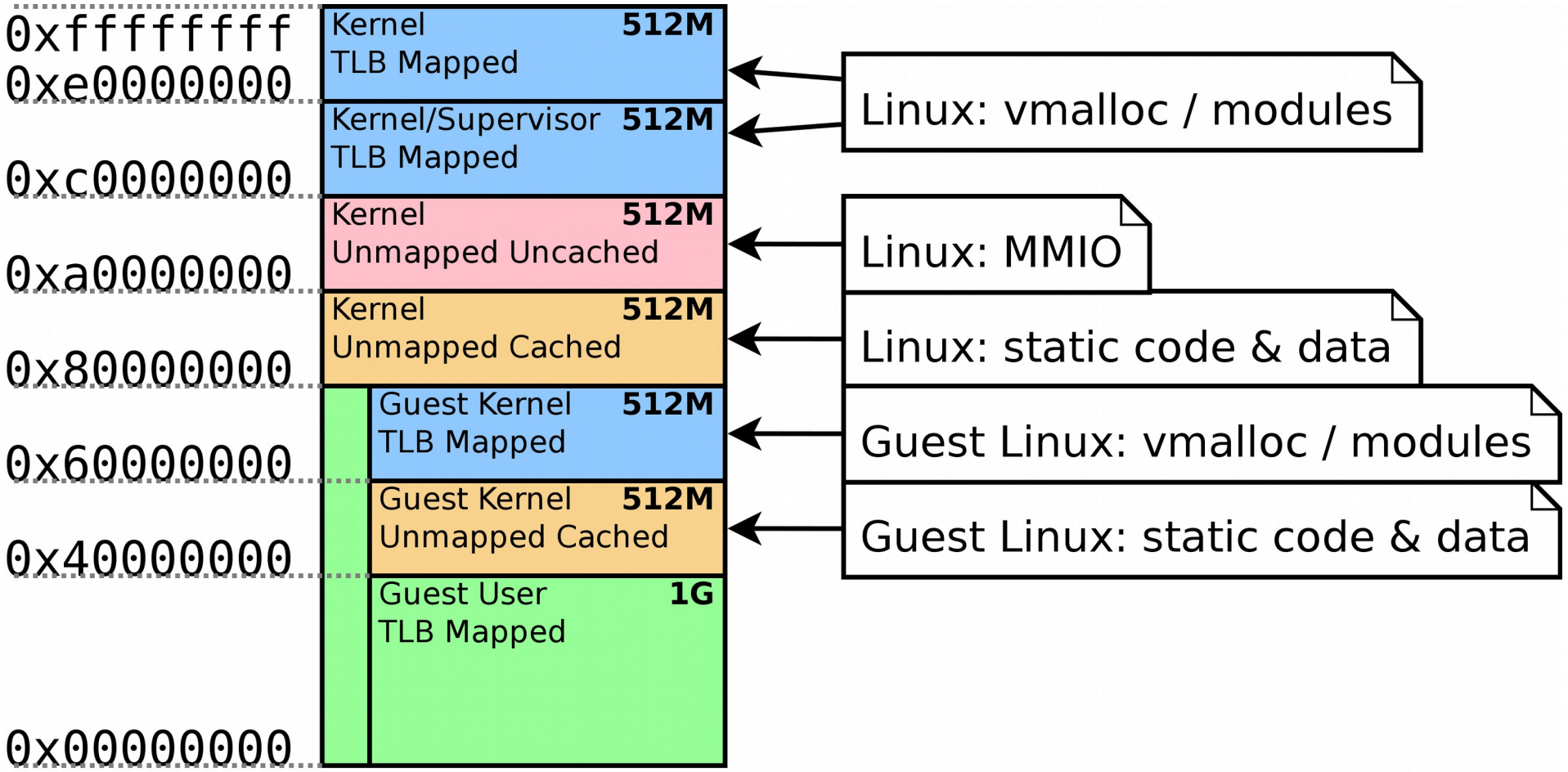
Trap & Emulate (T&E)

- 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



T&E Guest Mode Virtual Address Space



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

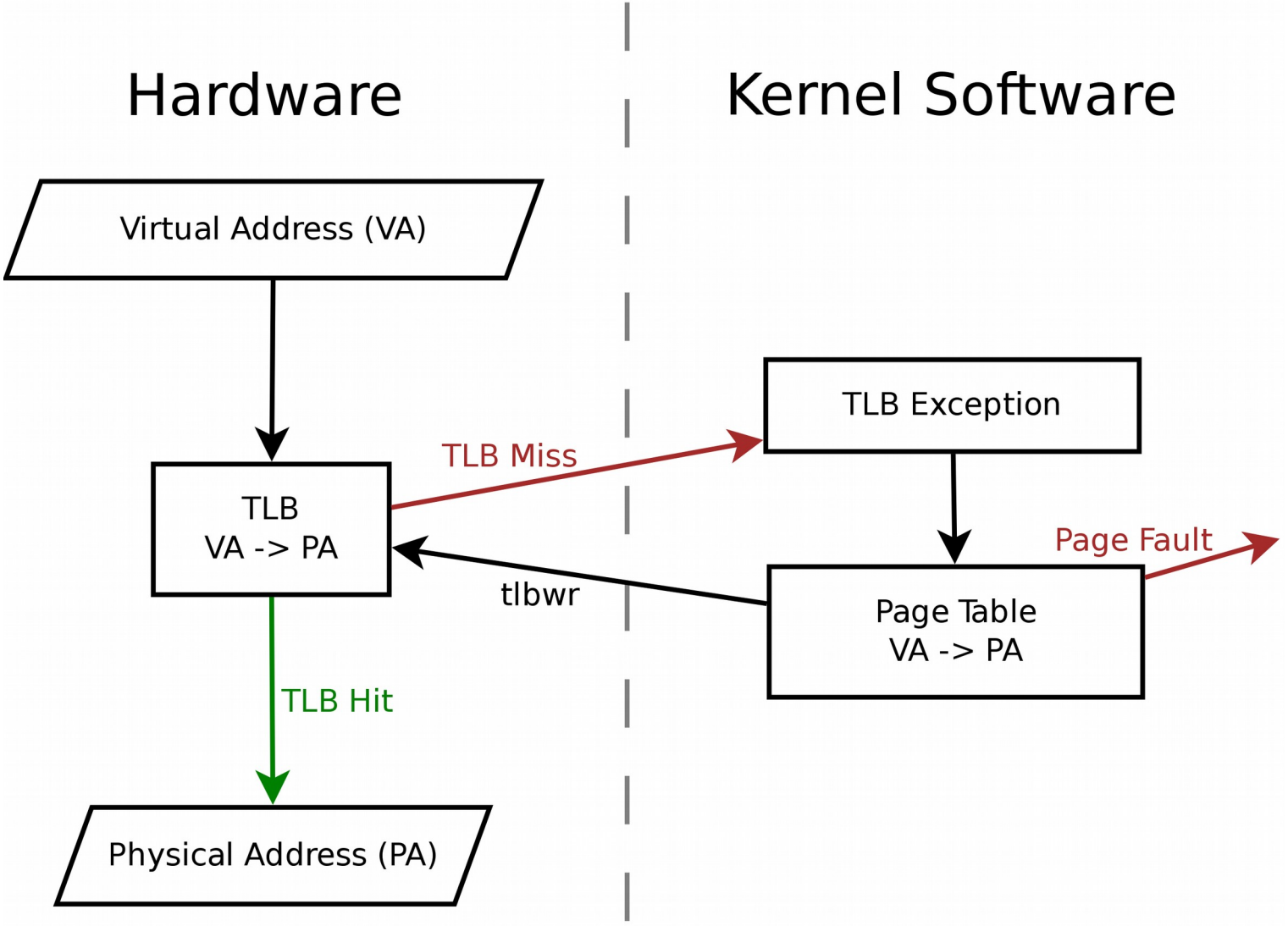
    `mtc0 rt,reg` → `sw rt,(reg*4)(zero)`

    `mfc0 rt,reg` → `lw rt,(reg*4)(zero)`

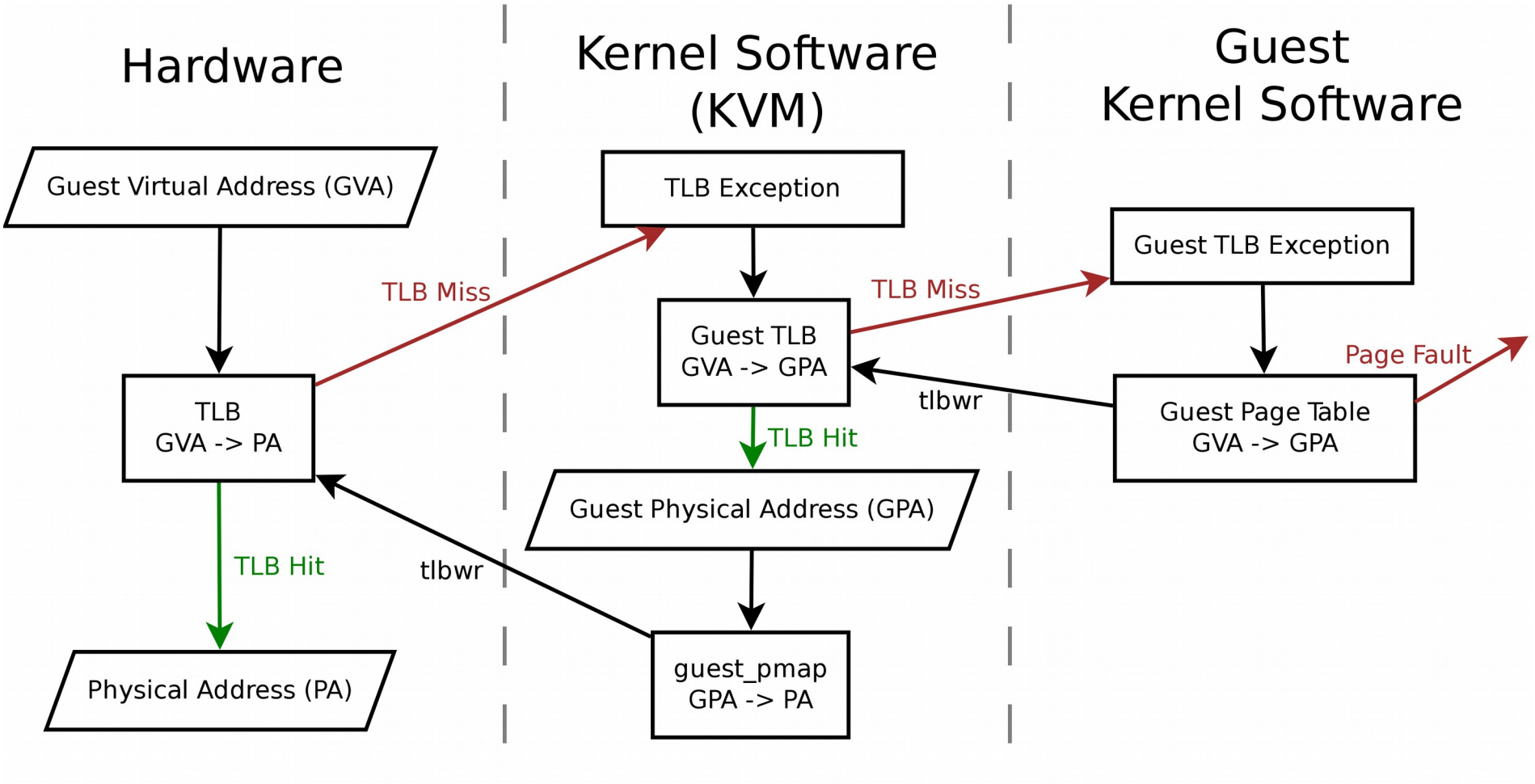
MIPS VZ

- 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]

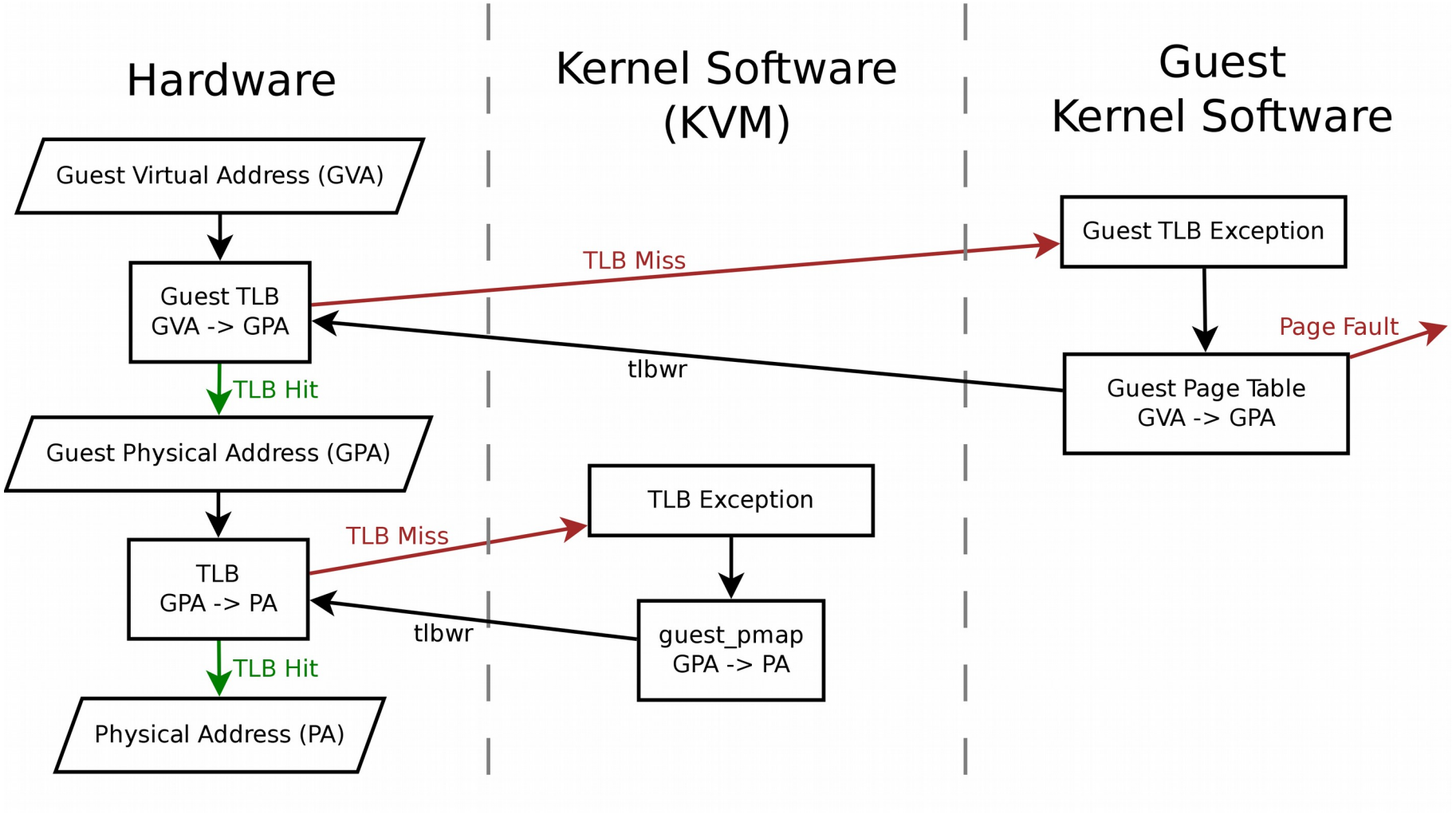
Normal TLB Management



T&E TLB Management



VZ TLB Management



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)
```

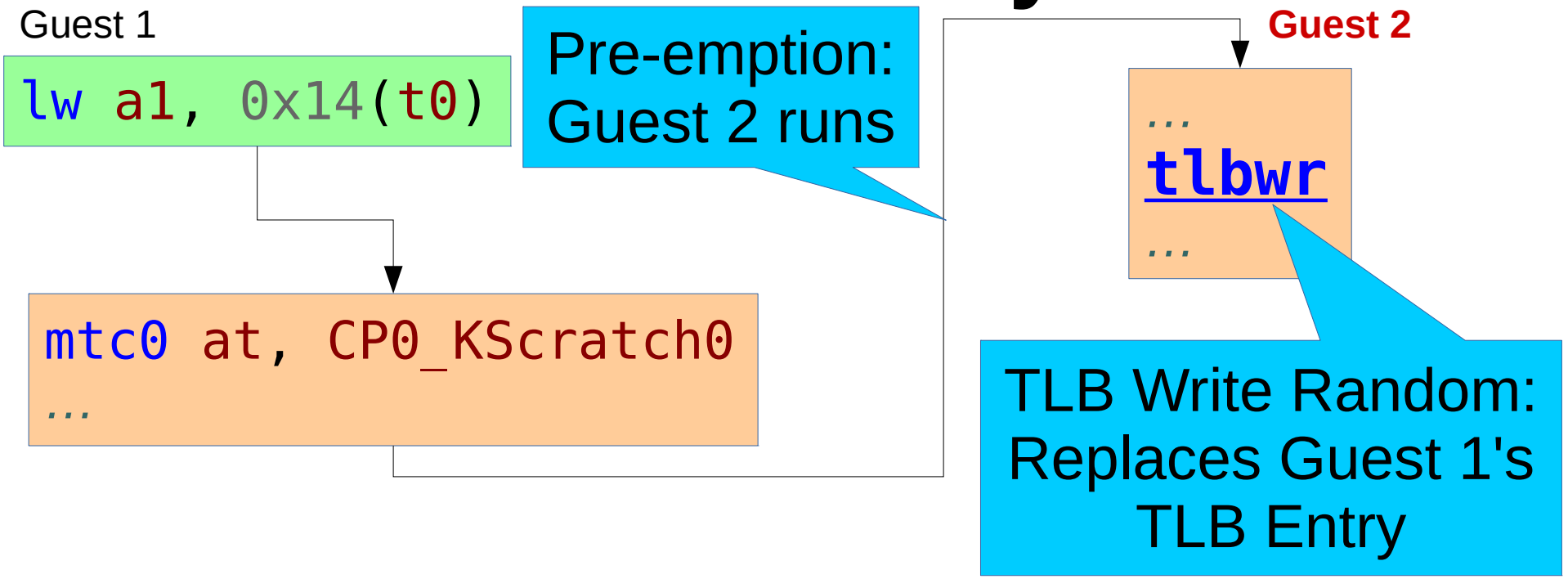
TLB mapping invalid:
Guest TLB Invalid Exception

```
mtc0 at, CP0_KScratch0
...
```

Guest 1 Register State	
Register	**Value**
t0	0x**0123C**000
CP0_BadVAddr	**0x0123C014**
CP0_PageMask	0x0FFF9000 (16K)

Guest TLB Entries				
Index	**GuestID**	**GVA**	**GPA0**	**GPA1**
34	1	0x**01238**xxx	0x08228xxx	***invalid***

PDTrace Analysis



Guest TLB Entries				
Index	**GuestID**	**GVA**	**GPA0**	**GPA1**
34	**2**	**0x3FF80xxx**	**0x12BC8xxx**	**0x13BF0xxx**

PDTrace Analysis

Pre-emption: Guest 1 runs again

Guest 1

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

TLB Probe: No matching TLB entry

Guest 2

```
...
tlbwr
...
```

Guest 1 Register State	
Register	**Value**
CP0_BadVAddr	**0x0123C014**
CP0_Index	**0xFFFFFFFF**
CP0_PageMask	0x0FFF9000 (16K)

Guest TLB Entries				
Index	**GuestID**	**GVA**	**GPA0**	**GPA1**
34	**2**	0x**3FF80**xxx	0x12BC8xxx	0x13BF0xxx

PDTrace Analysis

Guest 1

```
srl   k0, k0, 12
...
tlbp
andi  at, k0, 0x1
beqz  at, 0x803604a4
 andi at, k0, 0x80
beqz  at, 0x8036046c
 nop
tlbr
```

TLB Probe result (CP0_Index) not checked

Guest 2

```
...
tlbwr
...
```

Guest 1 Register State	
Register	**Value**
CP0_BadVAddr	0x0123C014
CP0_Index	**0xFFFFFFFF**
CP0_PageMask	**0x00000000 (4K)**

TLB Read: TLB registers reset to invalid

Guest TLB Entries		
GVA	**GPA0**	**GPA1**
0x3FF80xxx	0x12BC8xxx	0x13BF0xxx

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” https://lists.gnu.org/archive/html/qemu-devel/2014-06/msg04074.html
- KVM:
  2. [v2] “KVM for MIPS32 Processors” http://www.linux-mips.org/archives/linux-mips/2012-11/threads.html#00240
  3. Kyma: “KVM/MIPS32: Support for the new Virtualization ASE (VZ-ASE)” http://www.linux-mips.org/archives/linux-mips/2013-05/threads.html#00144
  4. Cavium: “KVM/MIPS: Implement hardware virtualization via the MIPS-VZ extensions.” http://www.linux-mips.org/archives/linux-mips/2013-06/threads.html#00132