Shadow TLB Management on PowerPC 440
KVM Forum 2008

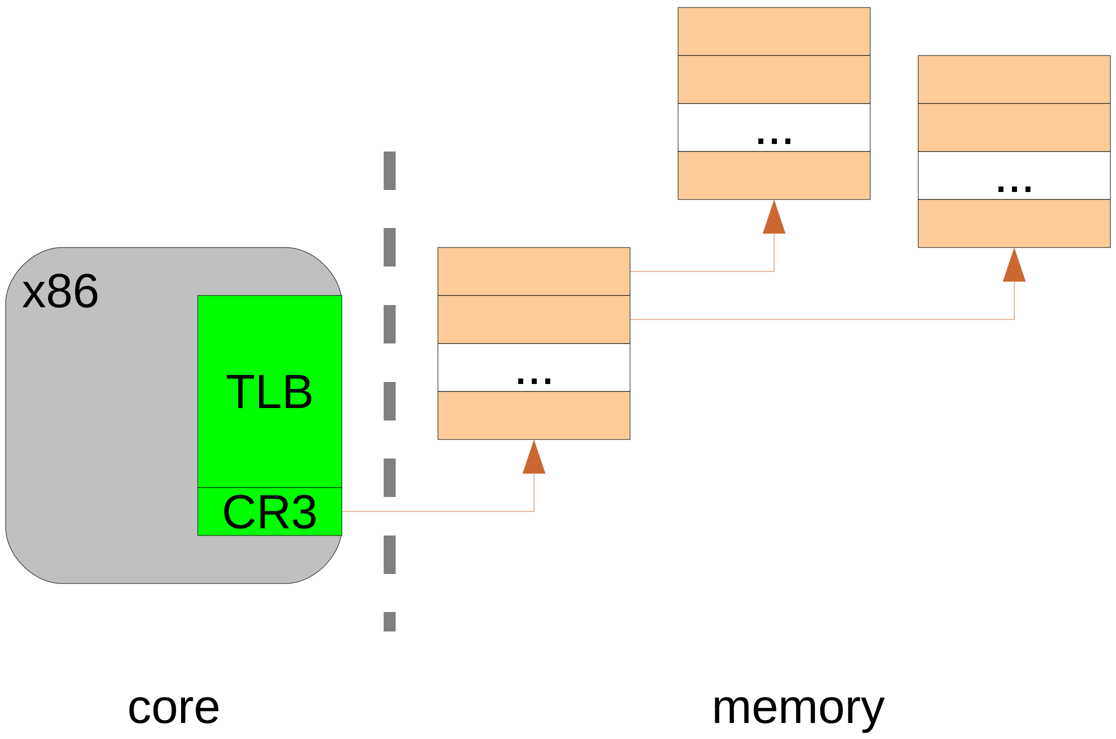
Hollis Blanchard
IBM Linux Technology Center

Agenda

- ➔Introduction to the software-controlled TLB
- •Virtualizing the software-controlled TLB
  - Replacement algorithm
  - Avoiding TLB misses
- •Summary

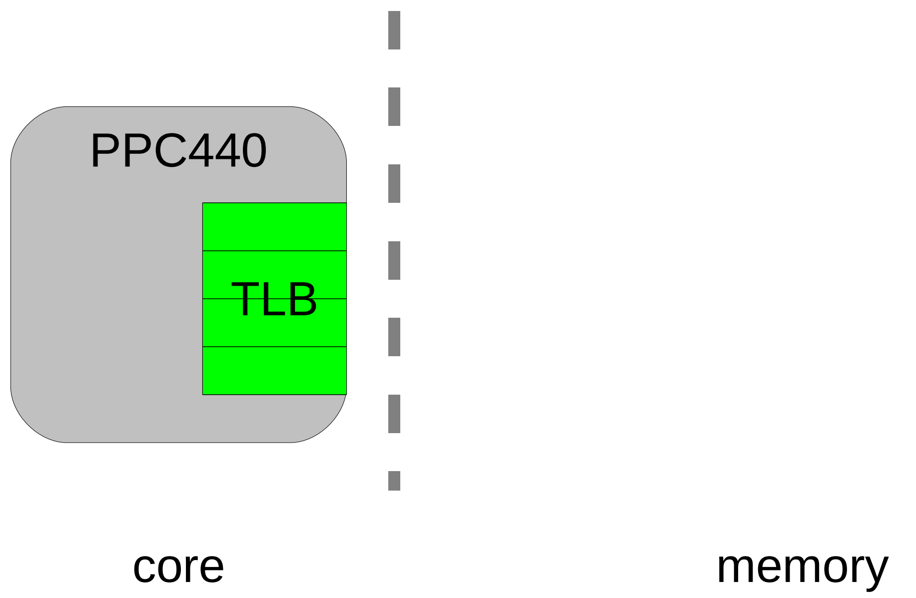
x86 Paging

- Hardware pagetable walker
  - Number of mappings limited only by available memory
    - 2-4MB to represent 1GB address space
  - Die space, power, and heat issues
  - Slow; need additional TLB



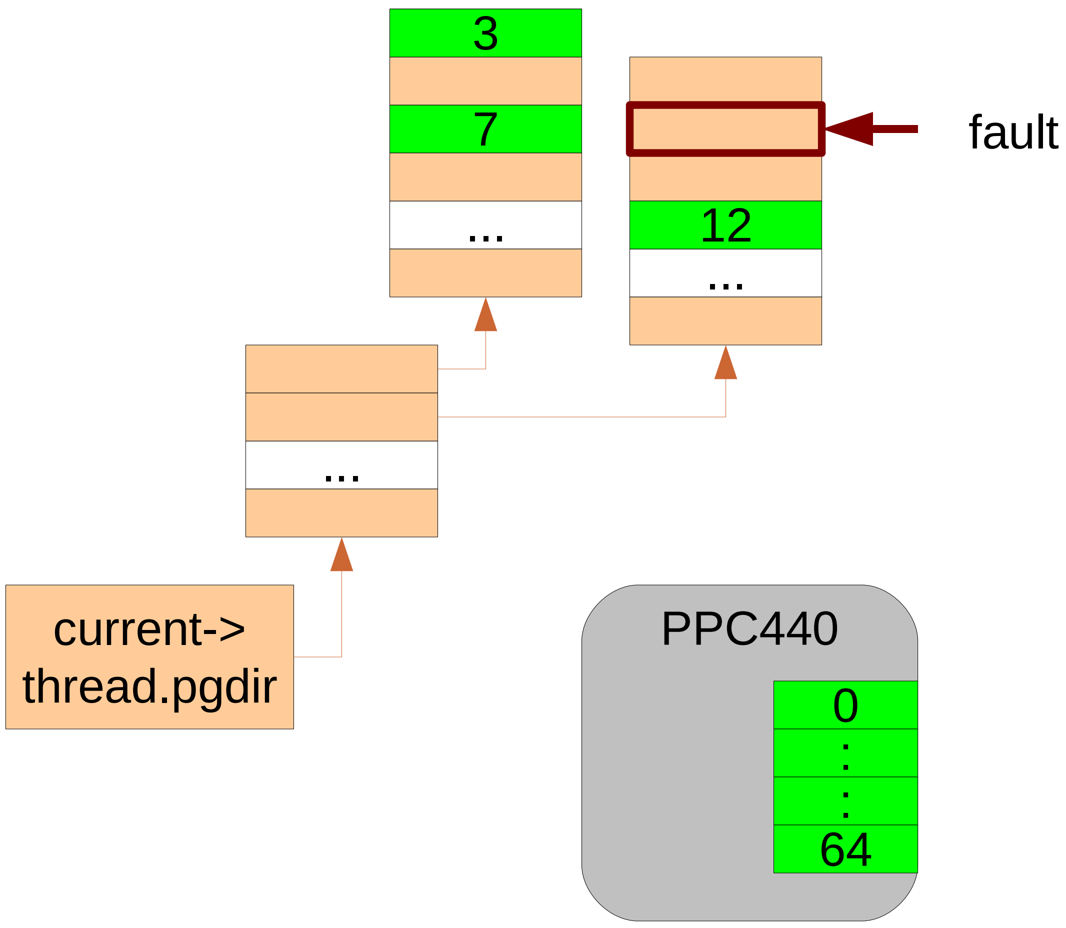
Software-controlled TLB

- No hardware pagetable walker
  - Avoid die costs
  - No memory required
- TLB
  - Limited number of simultaneous mappings (e.g. 64)
    - Large pages **very** important
  - Not in TLB? Invoke software TLB miss handler
    - Critical performance path



TLB Misses

- General-purpose operating systems need more than 64 mappings
  – Heavy TLB thrashing
  – Still need software-only “page table” structures
- Fast path
  – Miss in TLB, hit in page tables
  – Handler walks page tables in assembly
- Slow path
  – Miss in TLB, miss in page tables

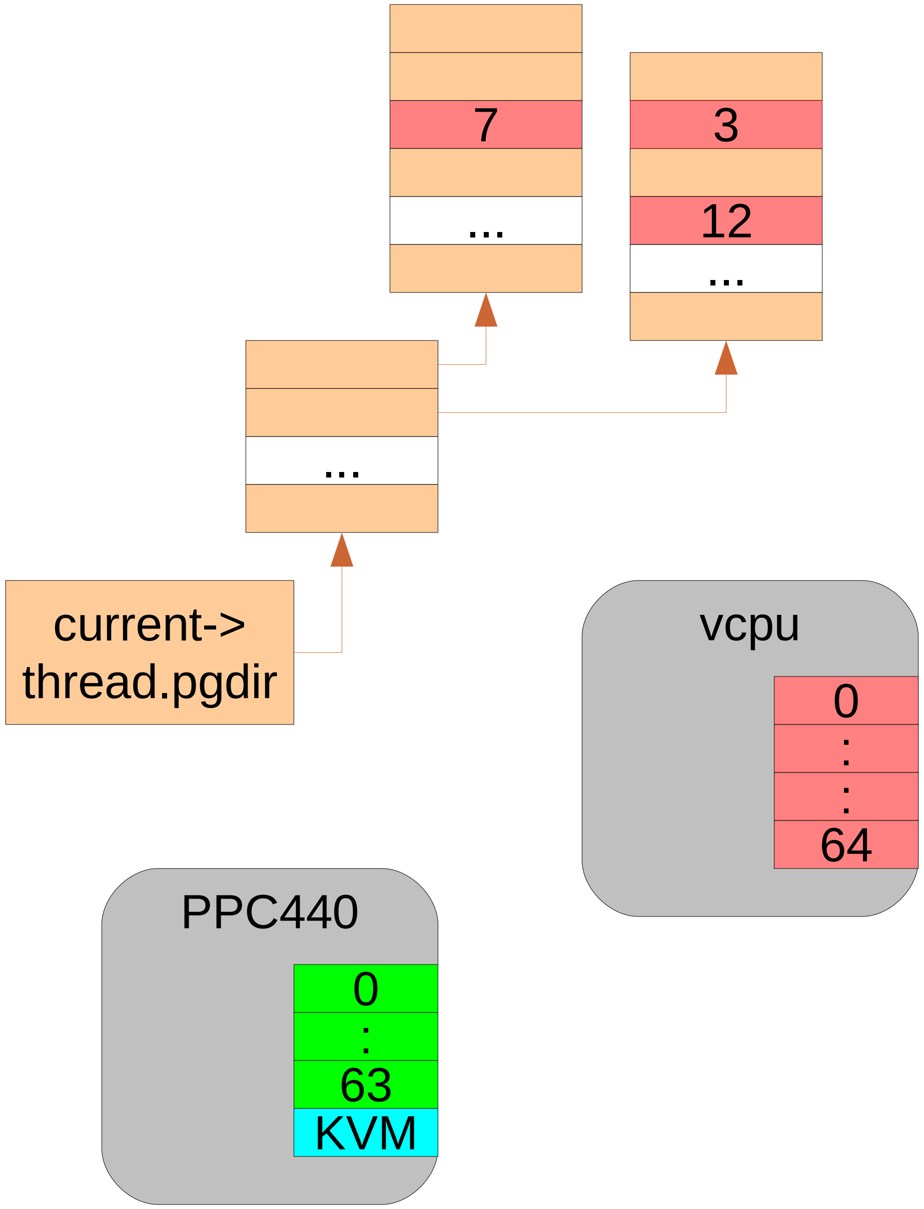


Agenda

- Introduction to the software-controlled TLB
- ➔Virtualizing the software-controlled TLB
  - Replacement algorithm
  - Avoiding TLB misses
- Summary

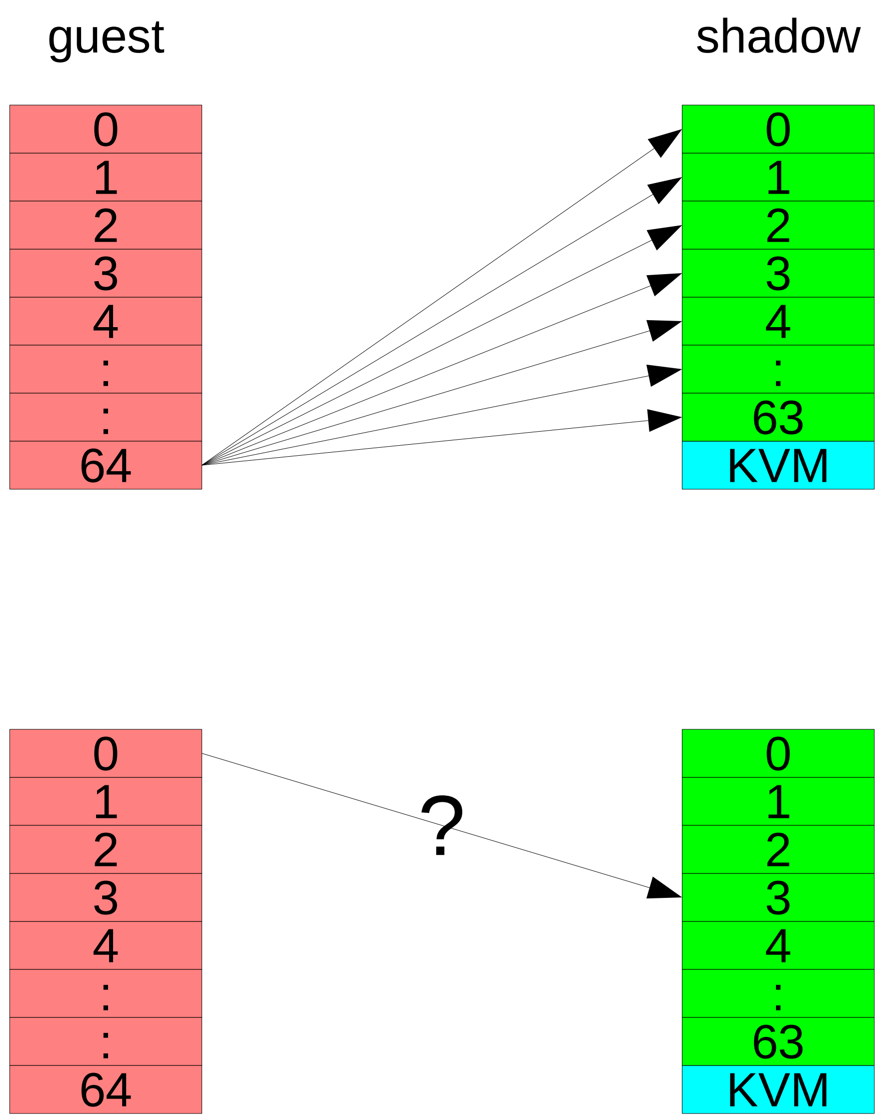
The Shadow TLB

- (Virtual) TLB holds guest physical addresses
  - Shadow TLB holds host physical addresses
  - Only 64 host pages at a time must be pinned
- Some guest mappings will be omitted from the shadow
  - MMIO mappings
  - Mappings we couldn't fit because we stole a TLB entry
- Emulate large guest pages with multiple small host pages



Shadow TLB Replacement Policy

- Single 256MB mapping covers 440 guest kernel
  - Could fill entire shadow TLB with 4KB mappings
- Break correlation between guest and host TLB entry indexes
  - Assumes full associativity
  - But we must propagate guest TLB changes to all corresponding shadow entries!
- Host implements its own TLB replacement policy



Shadow TLB Replacement Observations

- Currently, Linux (guest) TLB replacement is round-robin
  - We can start there too
- The instruction pointer and stack pointer are heavily used virtual addresses
  - If our policy selects the shadow entry mapping either, let's try again
  - Stack pointer is an ABI convention, not an architected register
    - Luckily, most (all?) PowerPC ABIs use GPR1 as the stack pointer

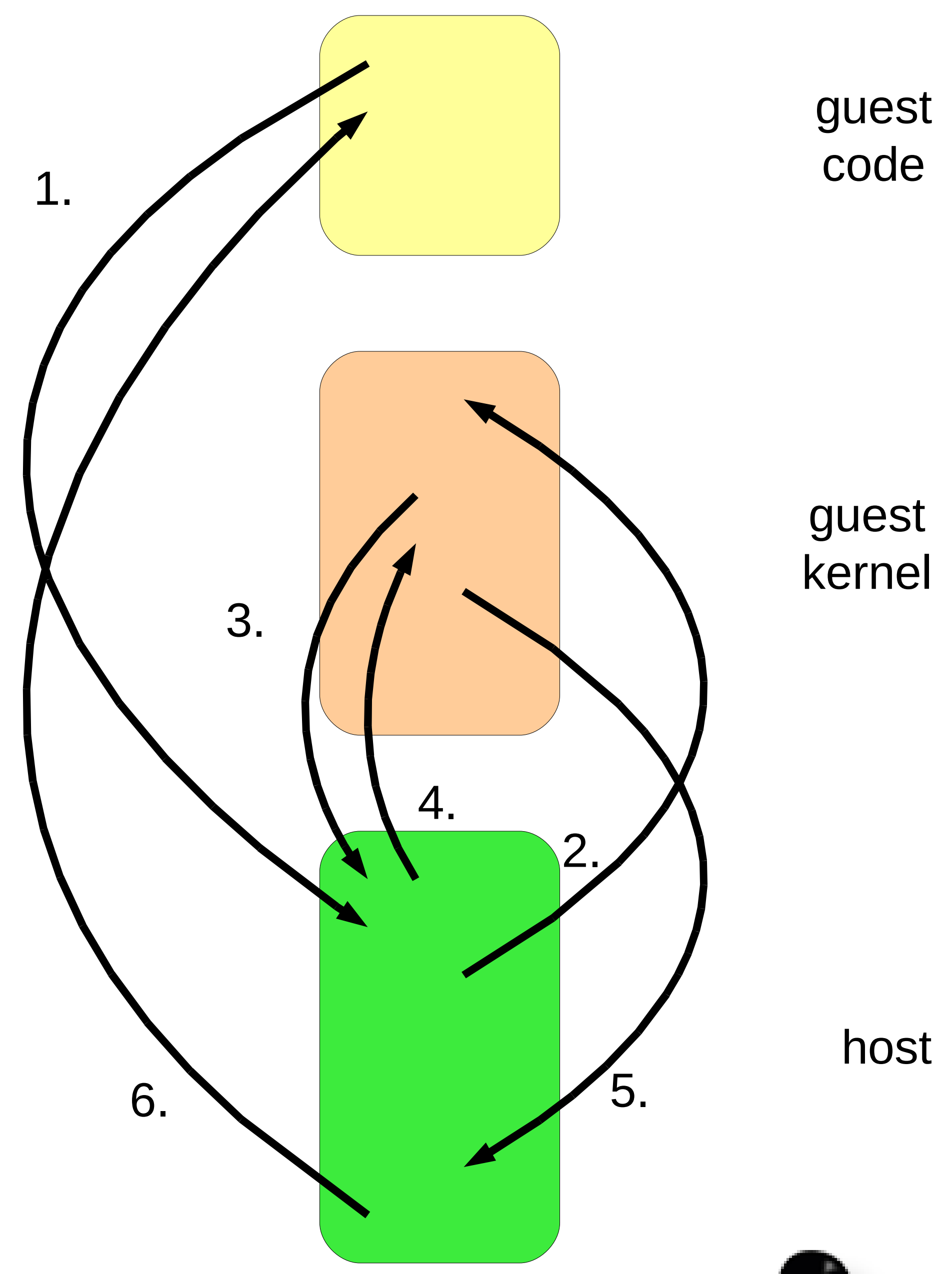
Shadow TLB Misses

- Fast path: shadow TLB miss, guest TLB hit
  - KVM opaquely handles fault without guest involvement
- Slow path: shadow TLB miss, guest TLB miss, guest page table hit
  - Must invoke guest to fill its TLB
  - But we'd rather not because this requires many context switches
  - Remember how this is the most performance-critical path?
- (Slowest path: shadow TLB miss, guest TLB miss, guest page table miss)
  - Nothing we can do here

Shadow TLB Misses: Context Switches

- Context switches
  1. Shadow and guest TLB miss
  2. Host invokes guest handler*
  3. Guest inserts new entry (hcall or instruction emulation*)
  4. Host returns from TLB insert
  5. Guest returns to host
  6. Host returns to interrupted code

* 440-specific problem: these may require multiple traps



Shadow TLB Miss Performance Mitigation

- Have host walk guest page tables
  - Brittle host code
  - These are software constructs, and very kernel-specific
- Define a new “hardware” page table format (and have host walk that)
  - Invasive to guest memory management code
- Advertise a larger TLB than hardware really has
  - After all, the goal is simply for host visibility into more guest mappings
  - Probably requires slight guest modification (guest TLB size may be a build-time constant)
  - Will want “TLB invalidate all” instruction/hypercall too (not all processors have this)

Agenda

- Introduction to the software-controlled TLB
- Virtualizing the software-controlled TLB
  - Replacement algorithm
  - Avoiding TLB misses
- ➔Summary

Summary

- The TLB miss interrupt handler is a critical performance path on systems with a software-controlled TLBs
- A shadow TLB is analogous to and simpler than shadow page tables
- Shadow TLBs exacerbate the TLB thrashing performance problem by increasing the number of context switches