Shadow pagetables update

Tim Deegan, XenSource UK
Shadow (II) architecture

- Shadow pages on demand
  - #PF, MOV CR3
  - Revoke guest write access to shadowed page
  - Unshadow by refcount, memory pressure, or bogus contents

- Trap and emulate writes
  - Atomically update original and shadows

- Per-domain “shadow lock”
What’s changed?

- Don’t maintain shadows of PAE L3 tables
  - Problems with shared-use pages
  - Complex “sub-shadow” mechanisms
  - Snapshot on MOV CR3 instead
- Fast unshadow
  - Keep a single upward link in each shadow
- Fast write-access revocation
  - Heuristics: guest’s linear maps, 1–1 maps
What’s changed?

• Fast-path for MMIO, not-present
  – Use reserved bits in shadow PTEs
  – Handle these faults without taking the lock

• Prefetching PTEs

• Interface cleanup
  – Allow other “paging assistance” modes
  – Shadow lock now entirely private
  – p2m code split from shadow code
What’s changed?

• SysMark  
  (Office Productivity)  
  A: Not optimized  
  B: Optimized  

• Most of the benefit in this benchmark is from the fast-path and prefetch
• Cache guest virt-to-phys translations
  – Faster lookups in MMIO + emulation
  – Invalidate on guest TLB flush or invlpg

• Emulate-ahead in PAE mode
  – Guest that writes 32 bits to a PAE pagetable will soon write the “other half”
  – x86_emulate now complete enough to run through a few instructions in between
• Re-introduce out-of-sync shadows
  – Only for pages with heavy churn
  – Spot many (~8) writes to the same page
  – Keep trap-and-emulate for demand-paging

• Finer-grained shadow locking
  – Per-page locks and top-down ordering
  – Not until we know the shadow lock is the bottleneck
Questions?
Paging interface

- Init/teardown of domains, vcpus.
- Domctl hypercall
  - Mem alloc’n, mode changes, log-dirty ops
- Calls for MMU programming events
  - #PF, MOV CR3, MOV CR0/4, INVLPG
  - Writes/cmpxchgs of pagetables
- Writes to p2m
- Reading/walking guest pagetables