# Virtualization

## The XEN Approach

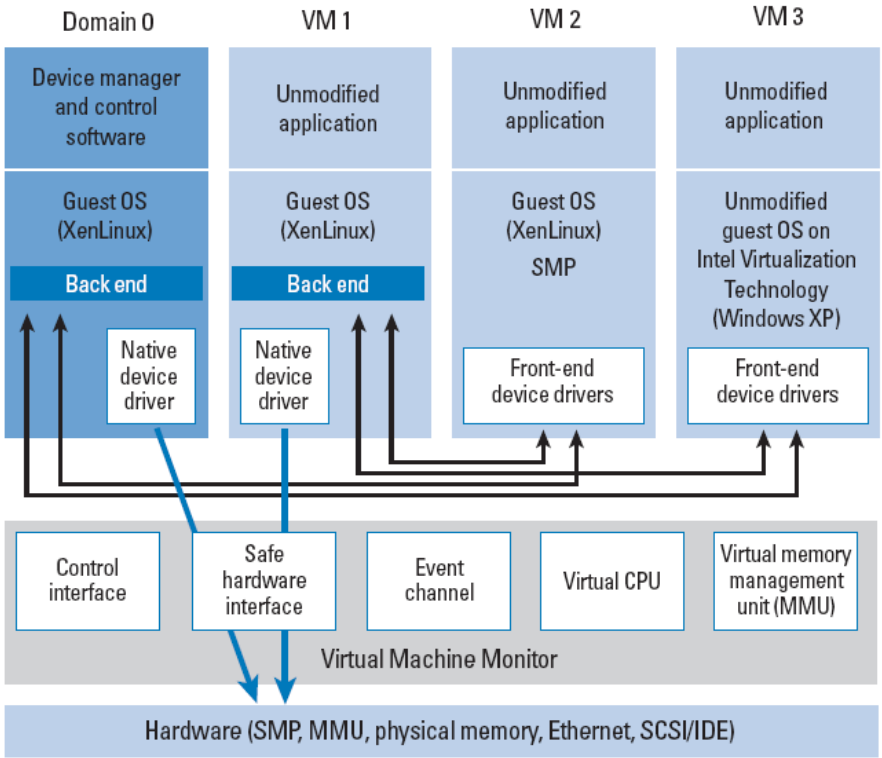
# XEN: *paravirtualization*

UNIVERSITY OF
CAMBRIDGE
Computer Laboratory


*References and Sources*

- Paul Barham, et.al., "Xen and the Art of Virtualization," Symposium on Operating Systems Principles 2003 (SOSP'03), October 19-22, 2003, Bolton Landing, New York.
- Presentation by Ian Pratt available at http://www.cl.cam.ac.uk/netos/papers/2005-xen-may.ppt

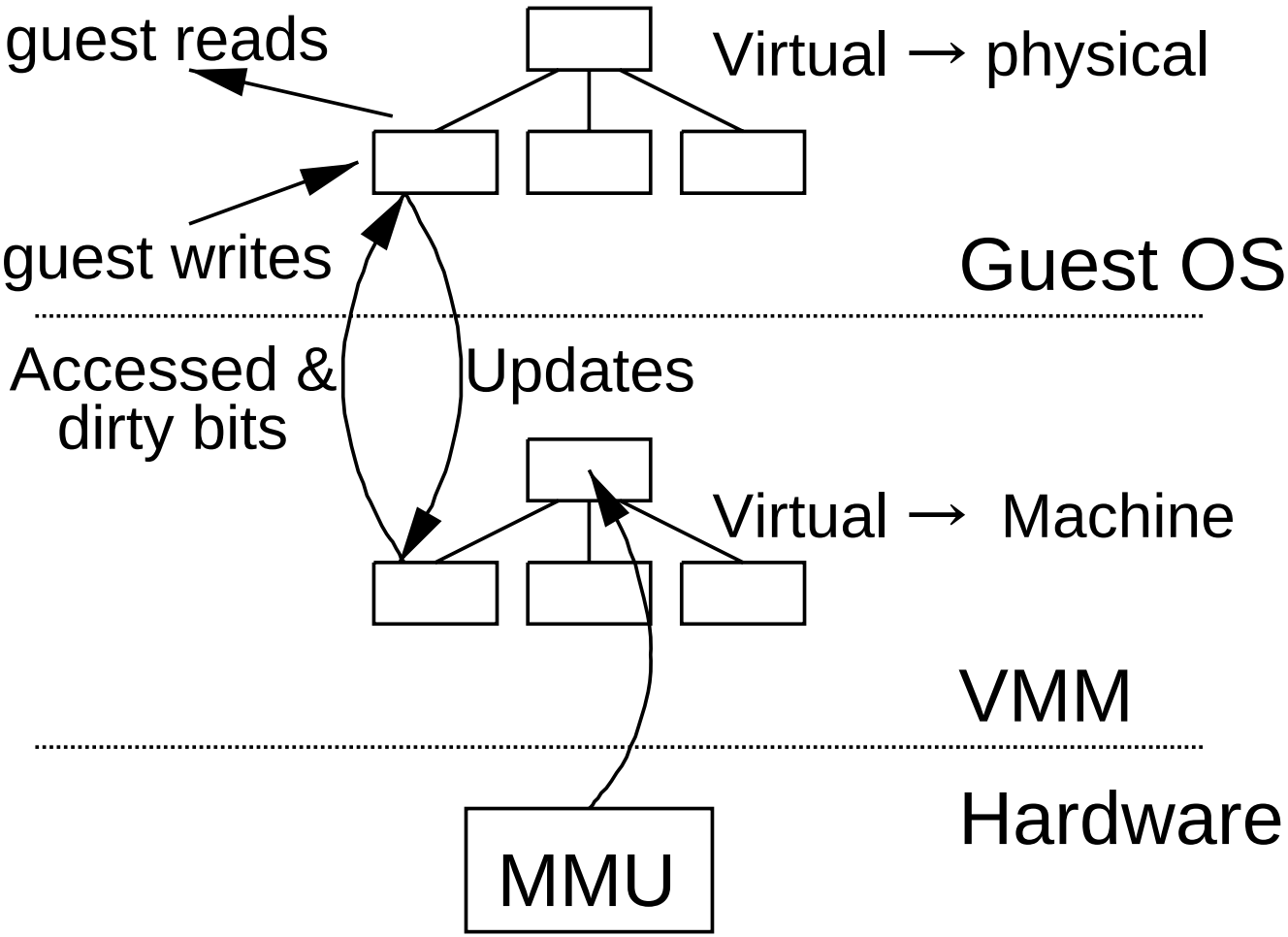
# Xen - Structure



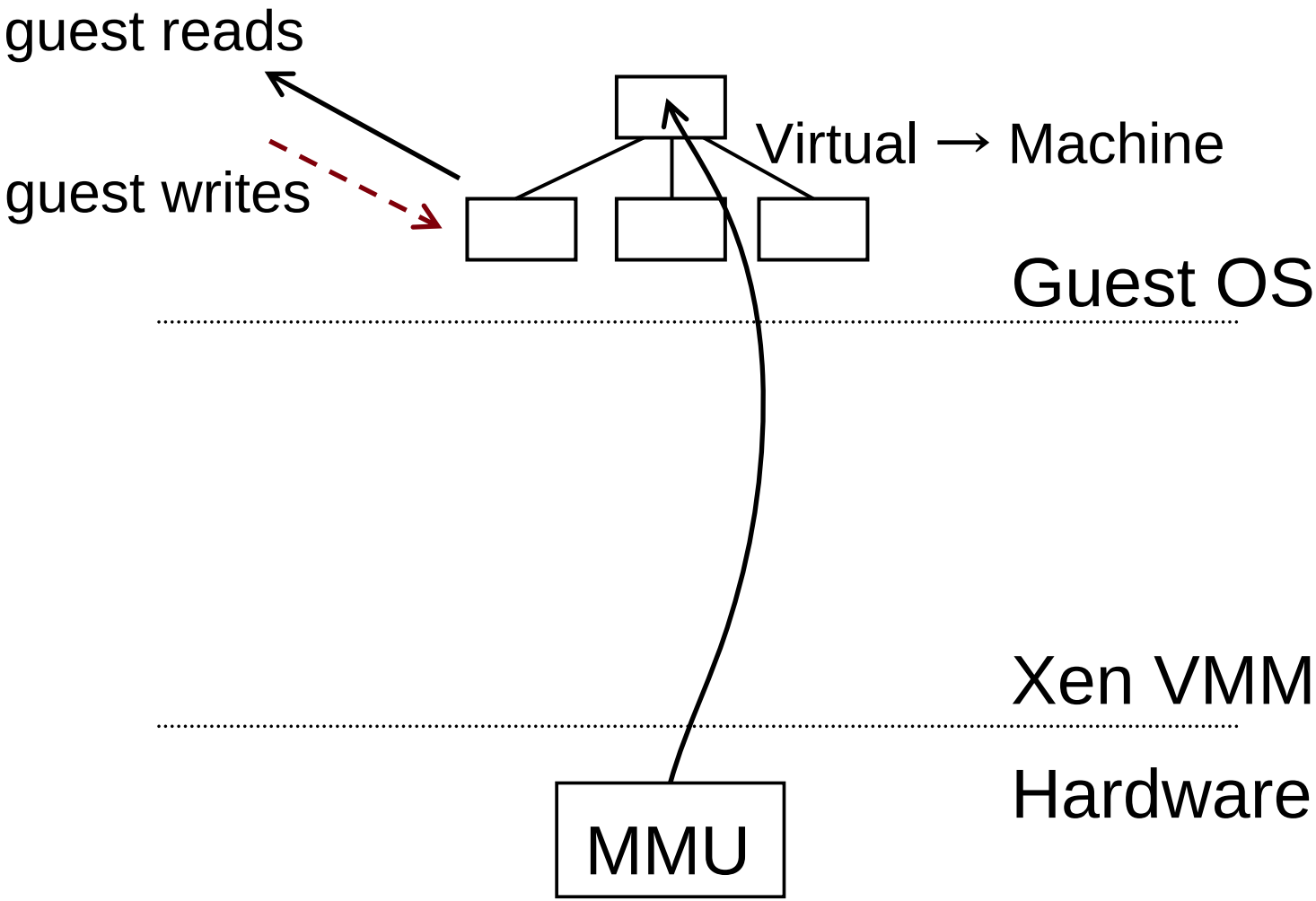

*Xen 3.0 Architecture*

- Employs paravirtualization strategy
  - Deals with machine architectures that cannot be virtualized
  - Requires modifications to guest OS
  - Allows optimizations
- "Domain 0"
  - has special access to control interface for platform management
  - Has back-end device drivers
- Xen VMM
  - entirely event driven
  - no internal threads

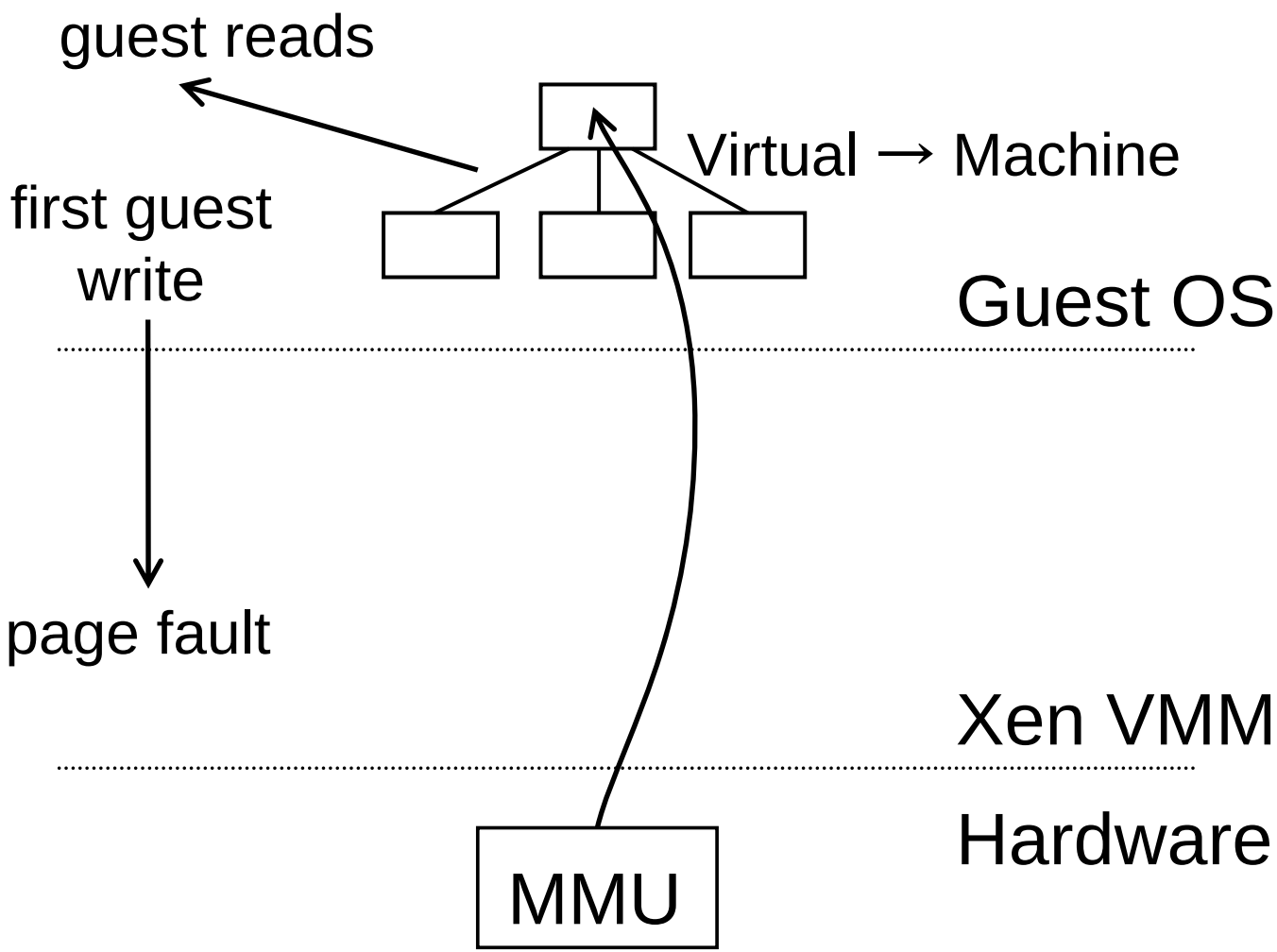
# MMU Virtualizion : Shadow-Mode



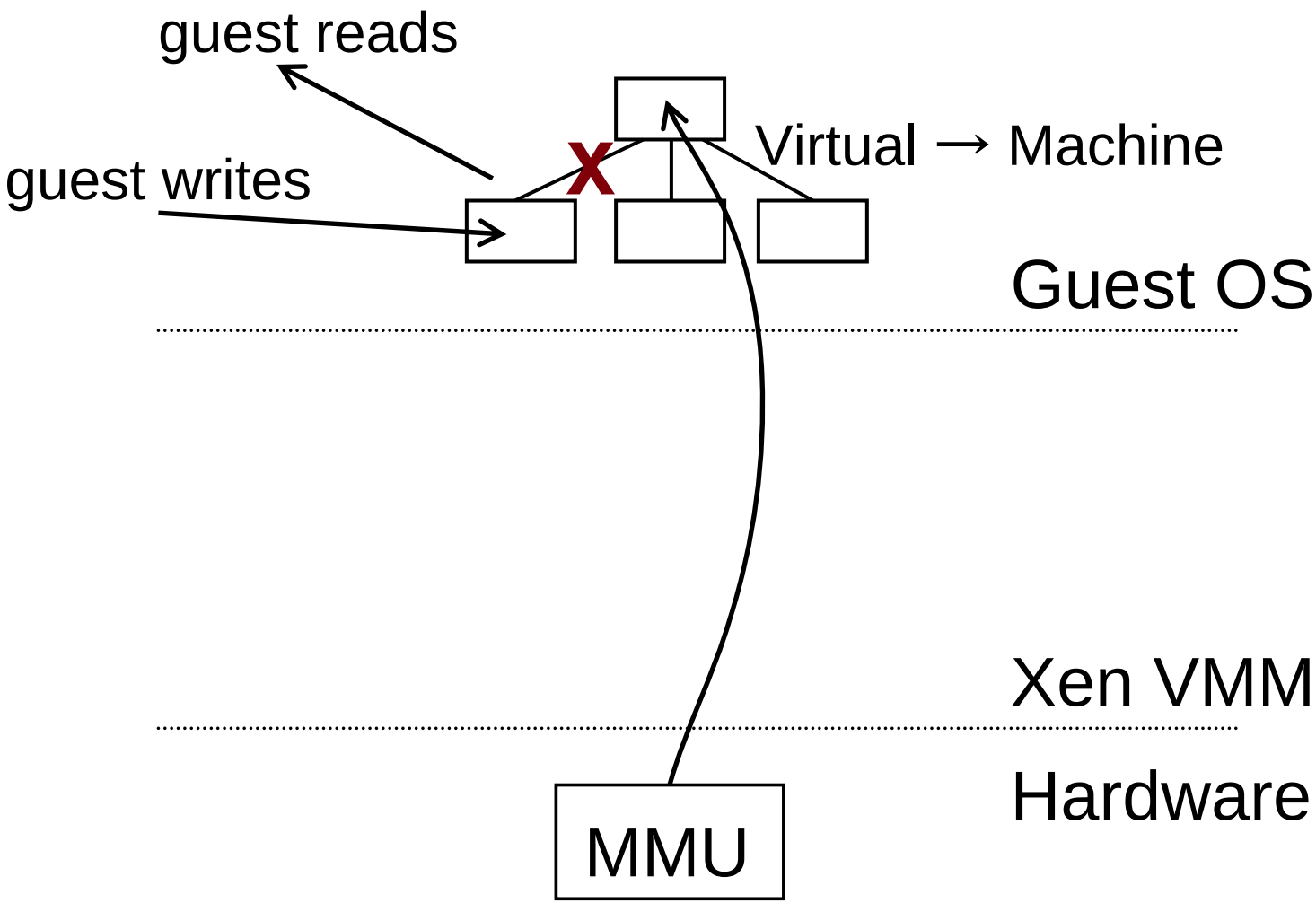
# MMU Virtualization : Direct-Mode



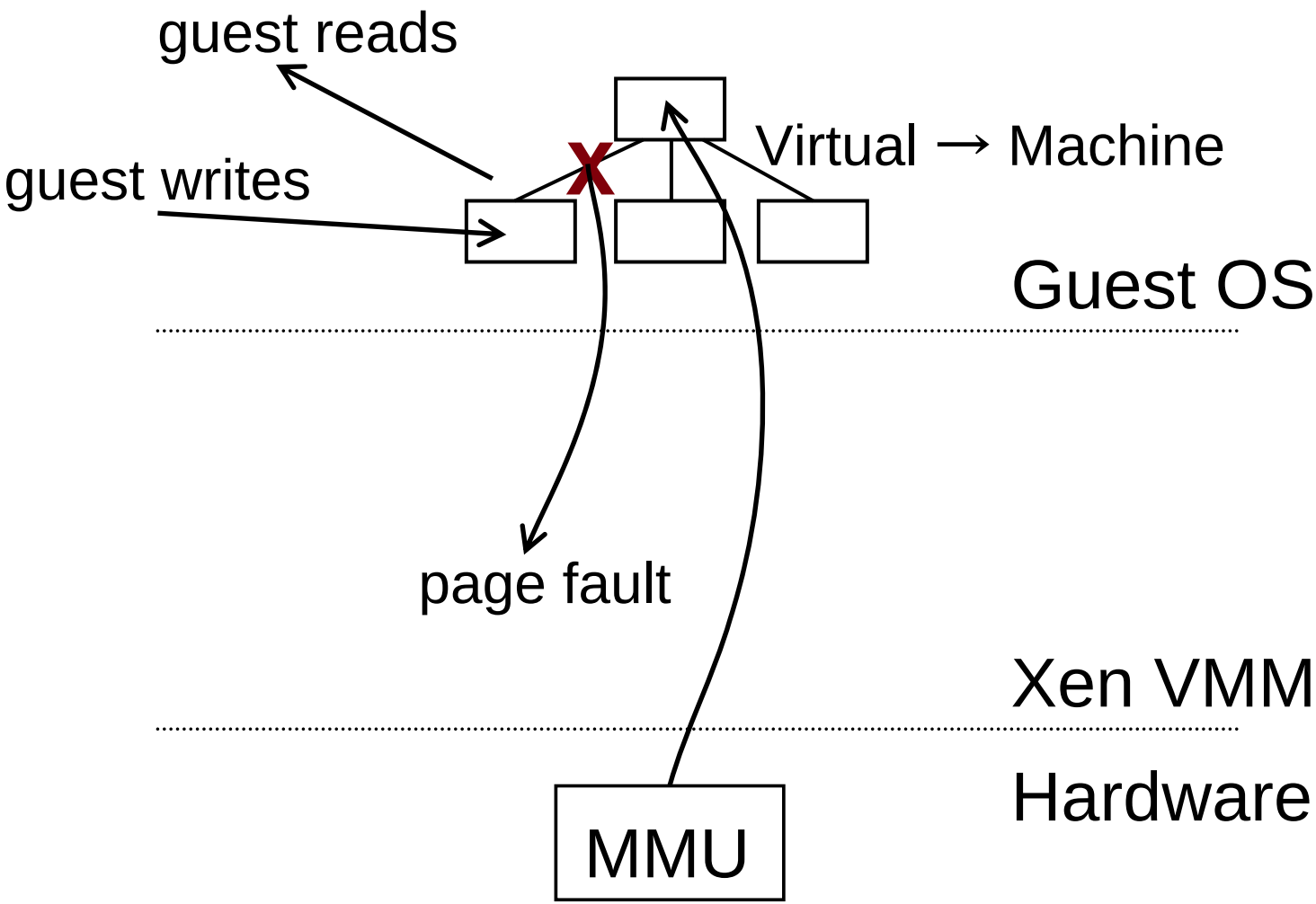
# Writeable Page Tables : 1 – write fault



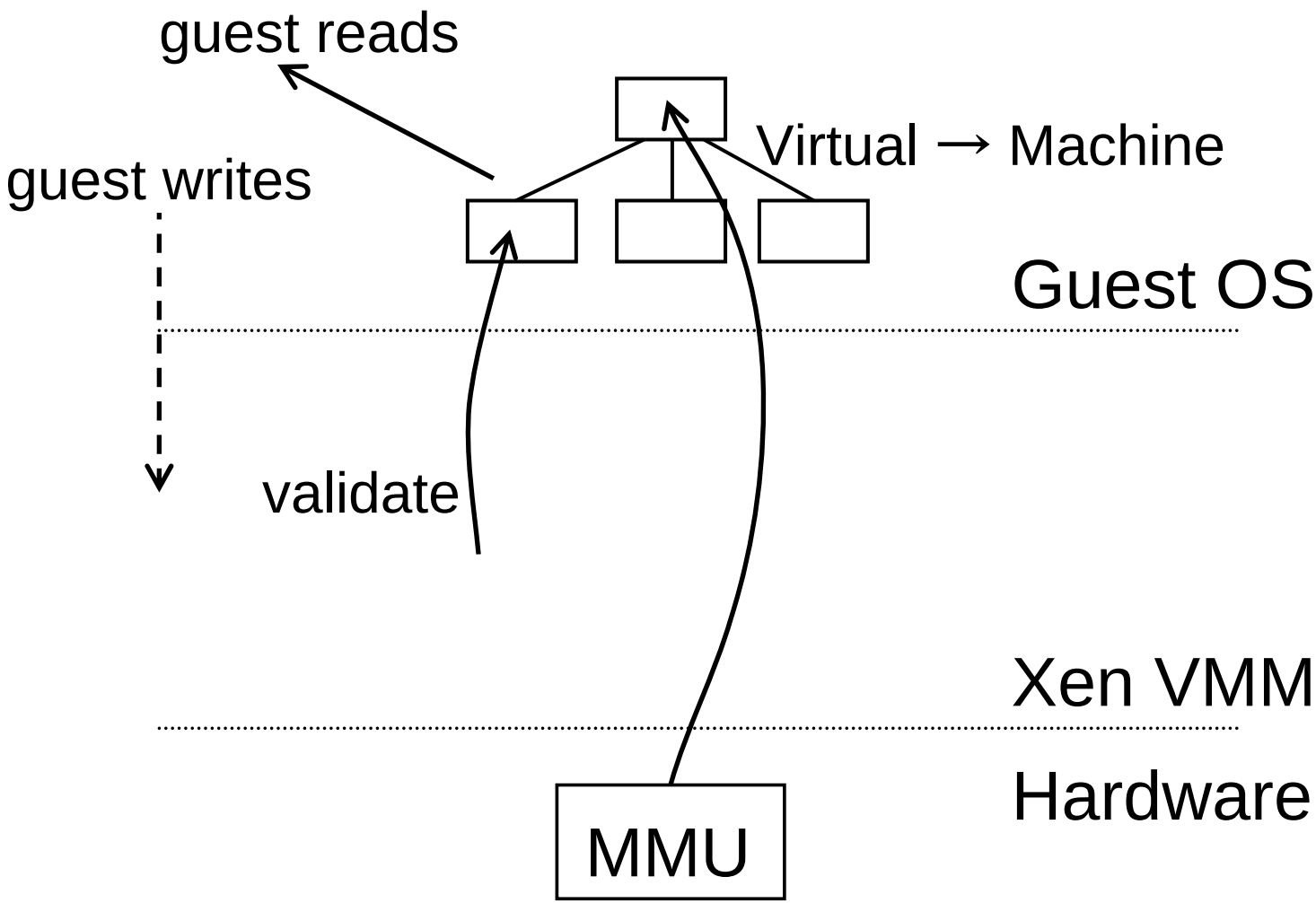
# Writeable Page Tables : 2 - Unhook



# Writeable Page Tables : 3 - First Use

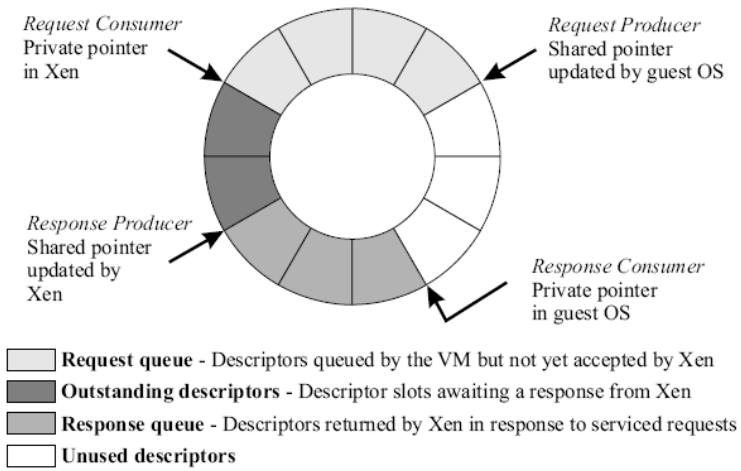


# Writeable Page Tables : 4 – Re-hook



# I/O

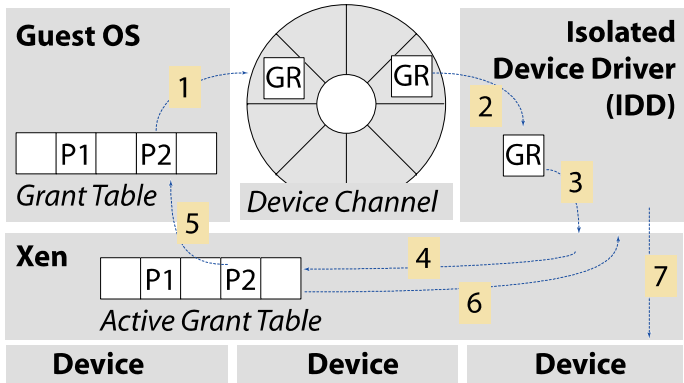
- Safe hardware interfaces
  - I/O Spaces
    - Restricts access to I/O registers
  - Isolated Device Drive
    - Driver isolated from VMM in its own “domain” (i.e., VM)
    - Communication between domains via device channels
- Unified interfaces
  - Common interface for group of similar devices
  - Exposes raw device interface (e.g., for specialized devices like sound/video)
- Separate request/response from event notification
- I/O descriptor rings
  - Used to communicate I/O requests and responses
  - For bulk data transfer devices (DMA, network), buffer space allocated out of band by GuestOS
  - Descriptor contains unique identifier to allow out of order processing
  - Multiple requests can be added before hypercall made to begin processing
  - Event notification can be masked by GuestOS for its convenience



# Device Channels

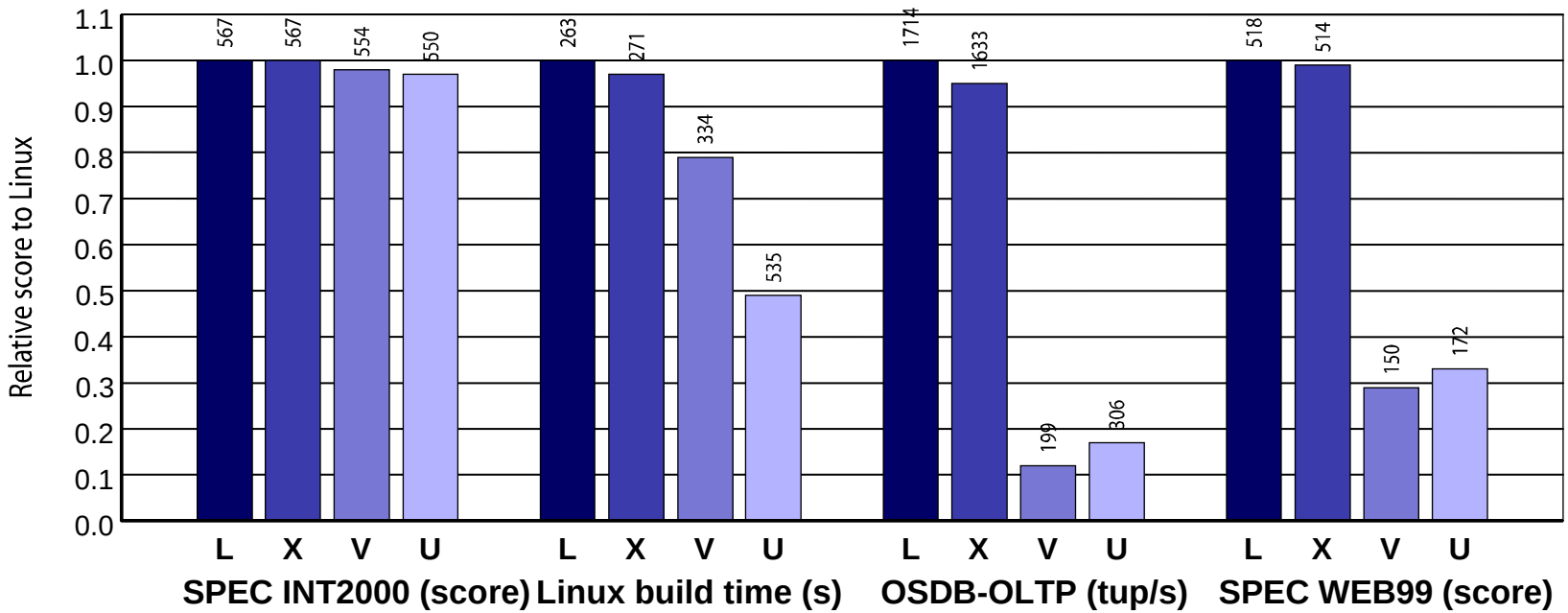
- Connects “front end” device drivers in GuestOS with “native” device driver
- Is an I/O descriptor ring
- Buffer page(s) allocated by GuestOS and “granted” to Xen
- Buffer page(s) is/are pinned to prevent page-out during I/O operation
- Pinning allows zero-copy data transfer

**Guest Requests DMA:**
1. Grant Reference for Page P2 placed on device channel
2. IDD removes GR
3. Sends pin request to Xen




4. Xen looks up GR in active grant table
5. GR validated against Guest (if necessary)
6. Pinning is acknowledged to IDD
7. IDD sends DMA request to device

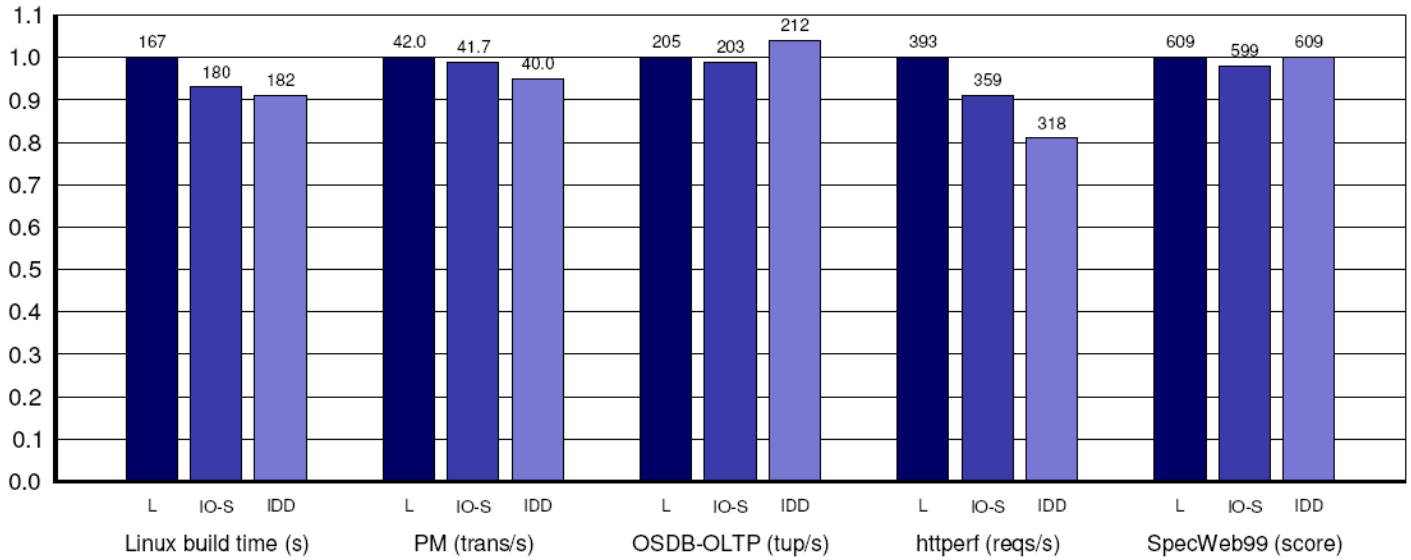
# System Performance




**Benchmark suite running on Linux (L), Xen (X), VMware Workstation (V), and UML (U)**

- Benchmark suites
  - Spec INT200: compute intensive workload
  - Linux build time: extensive file I/O, scheduling, memory management
  - OSBD-OLTP: transaction processing workload, extensive synchronous disk I/O
  - Spec WEB99: web-like workload (file and network traffic)
- Fair comparison?

# I/O Performance




- Systems
  - L: Linux
  - IO-S: Xen using IO-Space access
  - IDD: Xen using isolated device driver
- Benchmarks
  - Linux build time: file I/O, scheduling, memory management
  - PM: file system benchmark
  - OSDB-OLTP: transaction processing workload, extensive synchronous disk I/O
  - httperf: static document retrieval
  - SpecWeb99: web-like workload (file and network traffic)