Advanced Memory Management

Main Points

- Applications of memory management
  - What can we do with ability to trap on memory references to individual pages?
- File systems and persistent storage
  - Goals
  - Abstractions
  - Interfaces

Address Translation Uses

- *Process isolation*
  - Keep a process from touching anyone else's memory, or the kernel's
- *Efficient interprocess communication*
  - Shared regions of memory between processes
- *Shared code segments*
  - E.g., common libraries used by many different programs
- *Program initialization*
  - Start running a program before it is entirely in memory
- *Dynamic memory allocation*
  - Allocate and initialize stack/heap pages on demand

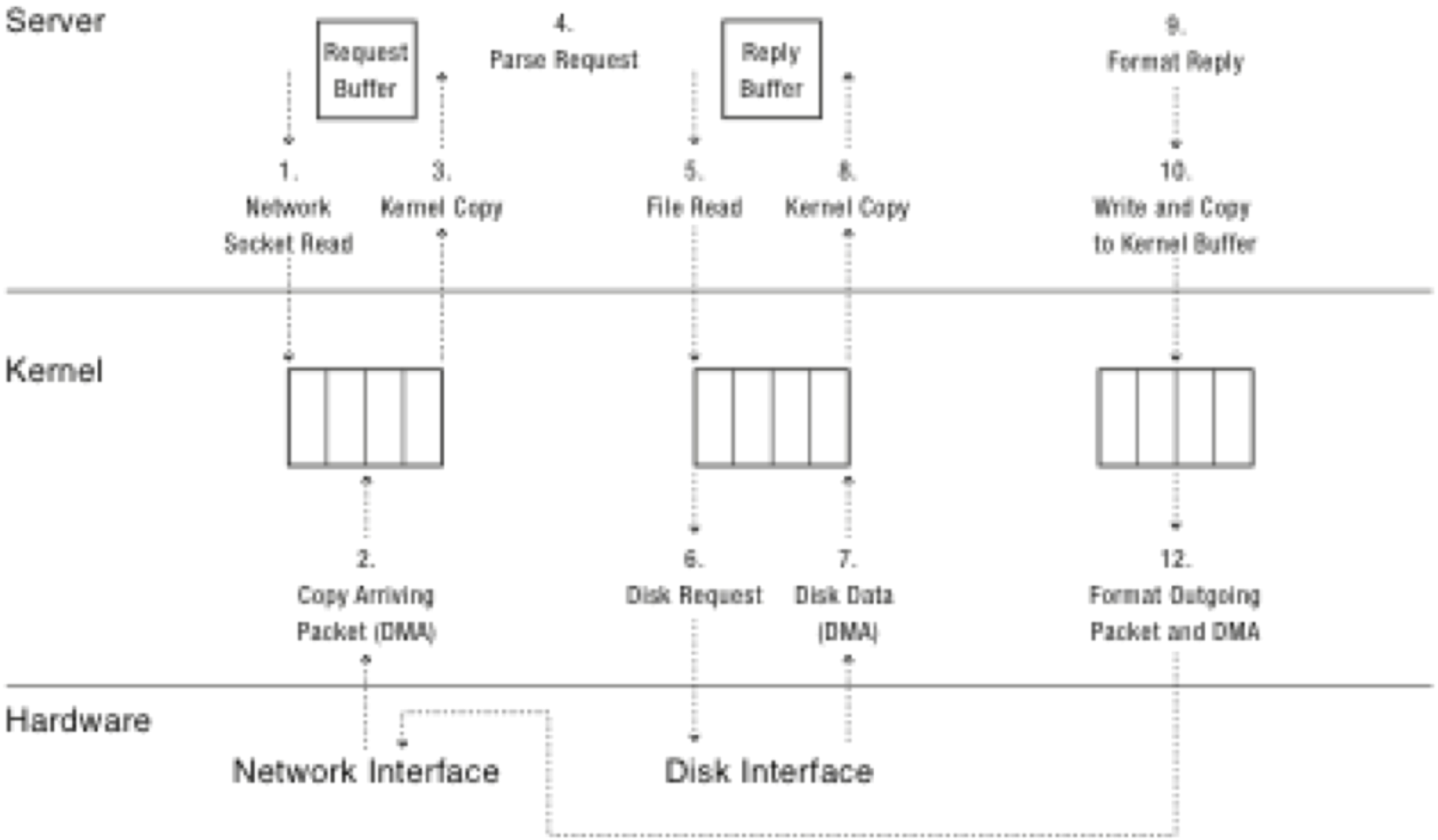
Address Translation (more)

- Program debugging
  - Data breakpoints when address is accessed
- *Zero-copy I/O*
  - Directly from I/O device into/out of user memory
- *Memory mapped files*
  - Access file data using load/store instructions
- *Demand-paged virtual memory*
  - Illusion of near-infinite memory, backed by disk or memory on other machines

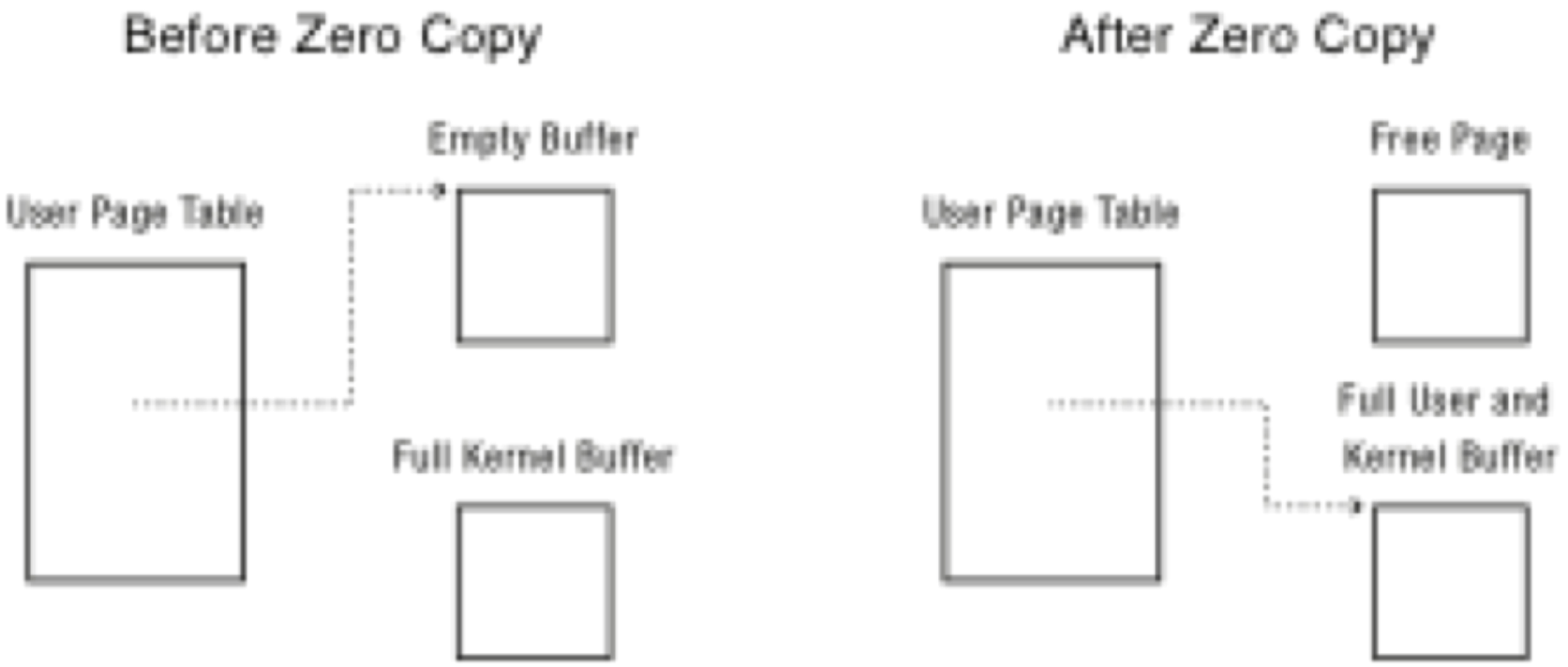
Address Translation (even more)

- Checkpoint/restart
  - Transparently save a copy of a process, without stopping the program while the save happens
- Persistent data structures
  - Implement data structures that can survive system reboots
- Process migration
  - Transparently move processes between machines
- Information flow control
  - Track what data is being shared externally
- Distributed shared memory
  - Illusion of memory that is shared between machines

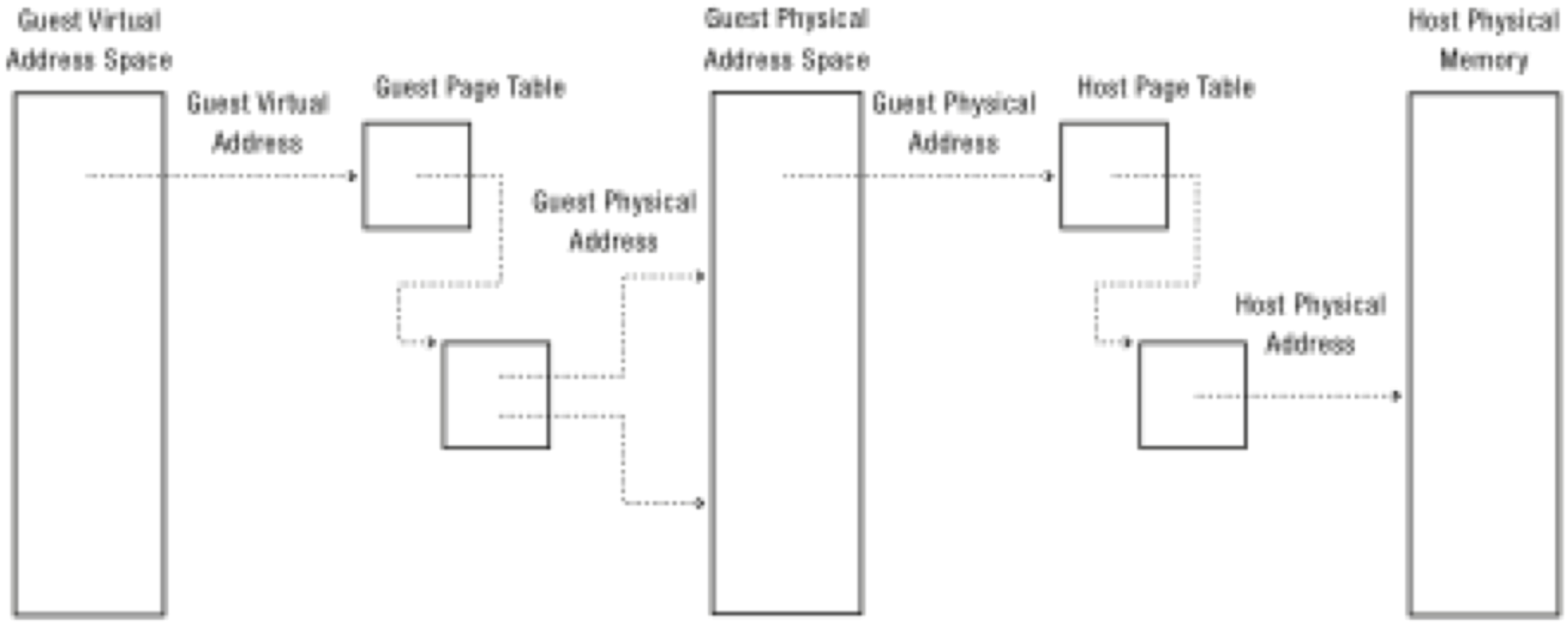
Web Server


Server
Request Buffer
4. Parse Request
Reply Buffer
9. Format Reply
1. Network Socket Read
3. Kernel Copy
5. File Read
8. Kernel Copy
10. Write and Copy to Kernel Buffer
Kernel
2. Copy Arriving Packet (DMA)
6. Disk Request
7. Disk Data (DMA)
12. Format Outgoing Packet and DMA
Hardware
Network Interface
Disk Interface

Zero Copy I/O
Block Aligned Read/Write System Calls



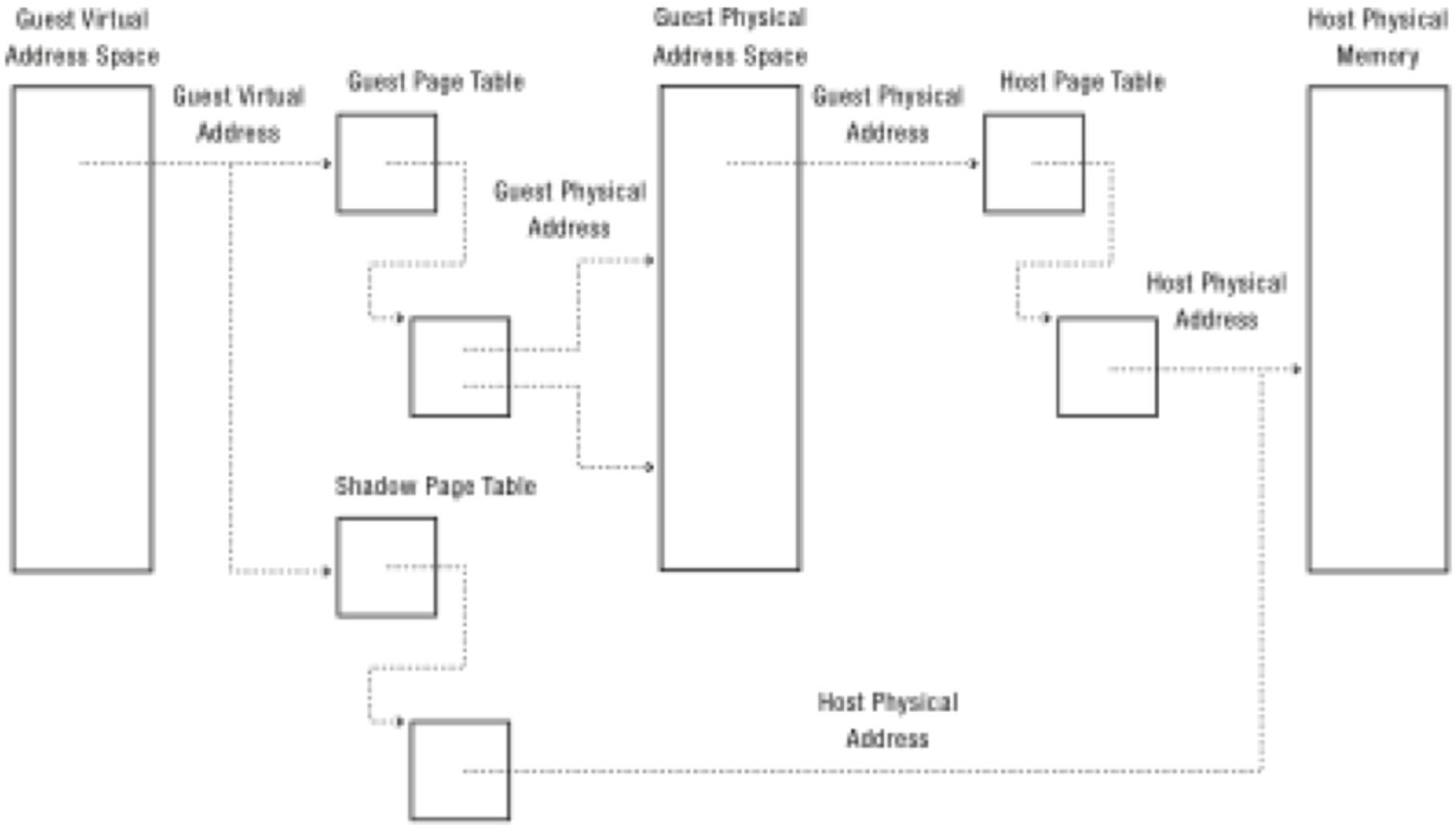
Virtual Machines and Virtual Memory



Segment Table		Page Table A		Page Table B	
0	Page Table A	0	0002	0	0001
1	Page Table B	1	0006	1	0004
x	(rest invalid)	2	0000	2	0003
		3	0005	x	(rest invalid)
		x	(rest invalid)		

Segment Table		Page Table K	
0	Page Table K	0	BEEF
x	(rest invalid)	1	F000
		2	CAFE
		3	3333
		4	(invalid)
		5	BA11
		6	DEAD
		7	5555
		x	(rest invalid)

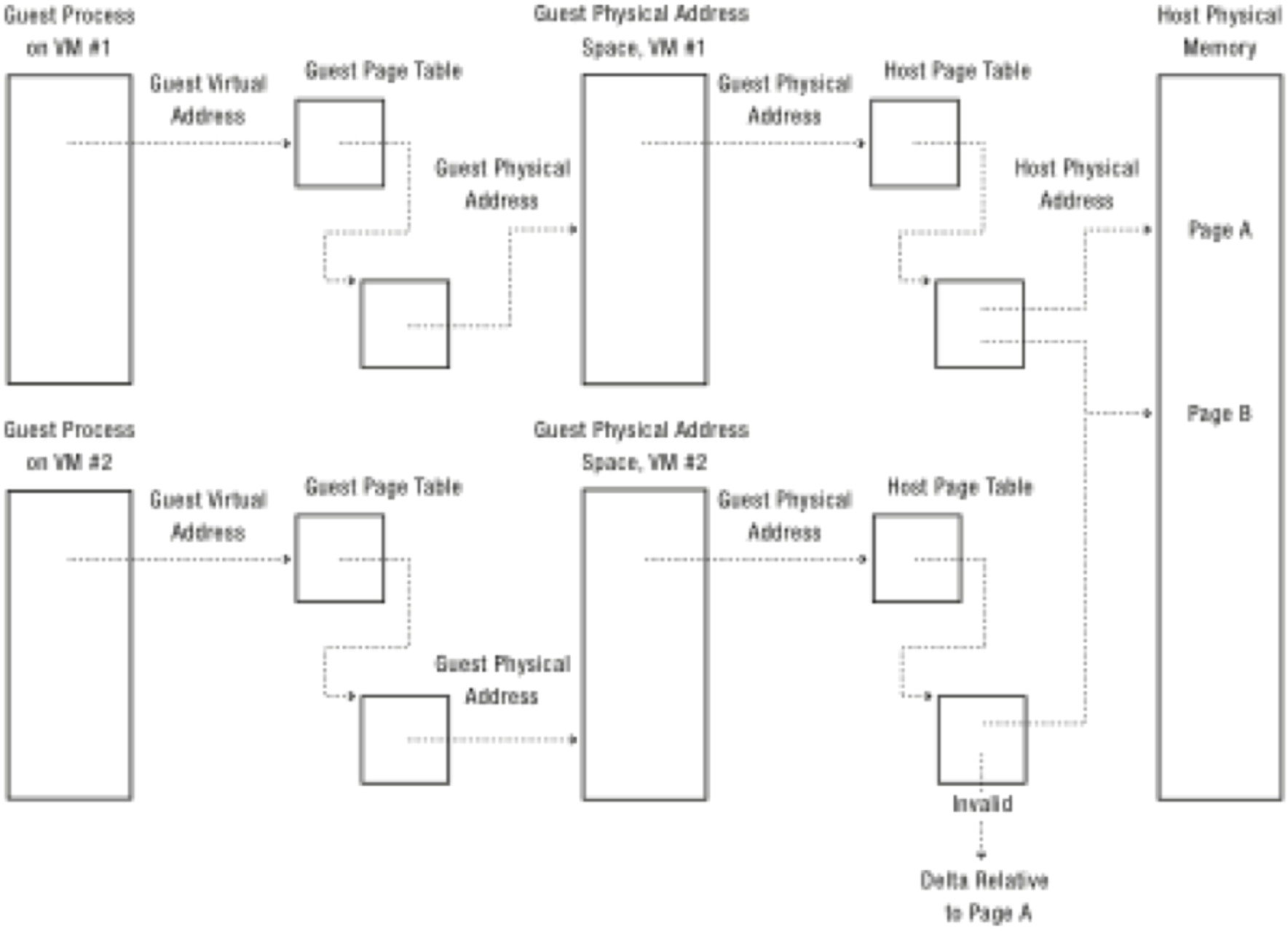
Shadow Page Tables



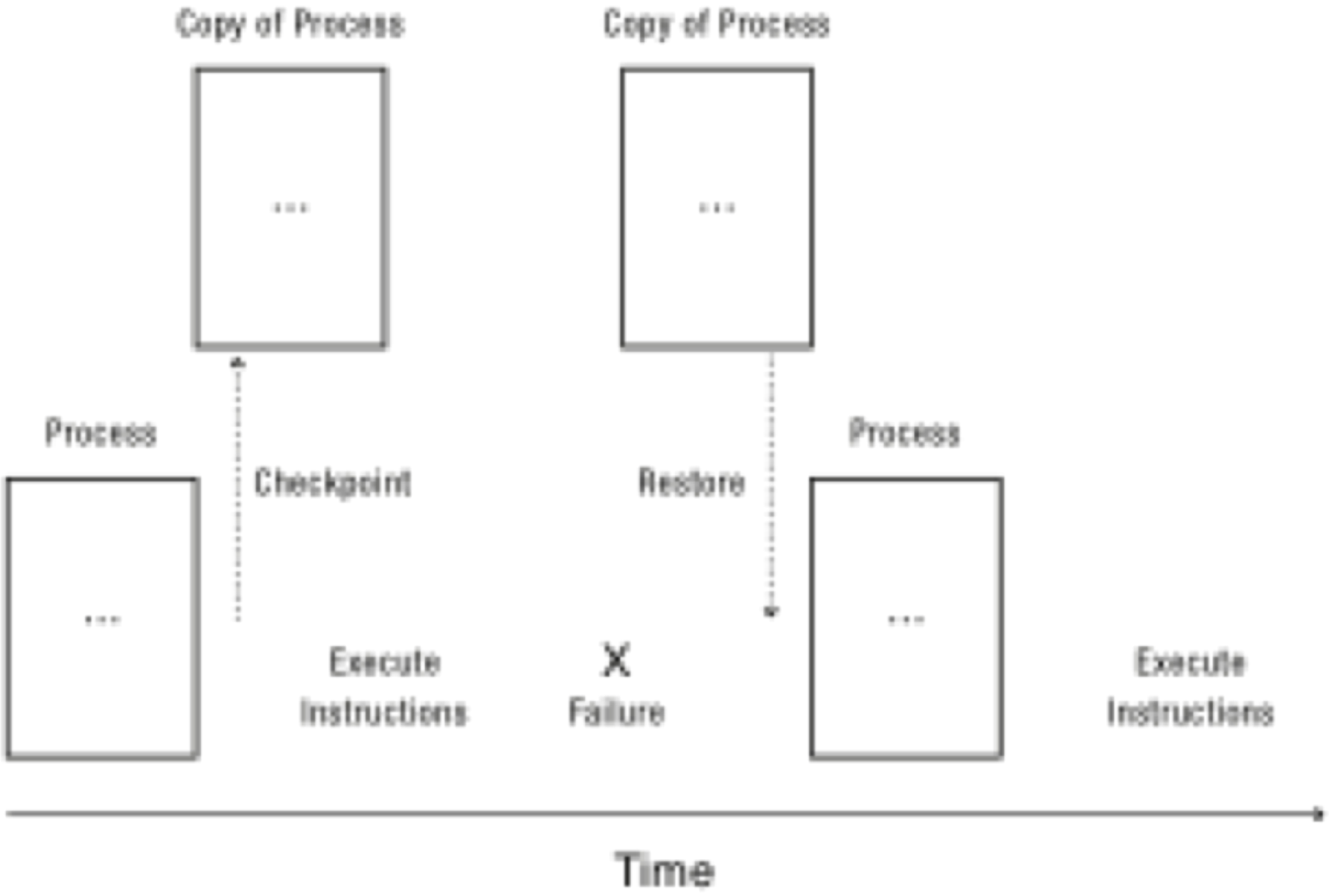
Hardware Support for Virtual Machines

- x86 recently added hardware support for running virtual machines at user level
- Operating system kernel initializes two sets of translation tables
  - One for the guest OS
  - One for the host OS
- Hardware translates address in two steps
  - First using guest OS tables, then host OS tables
  - TLB holds composition

VMM Memory Compression



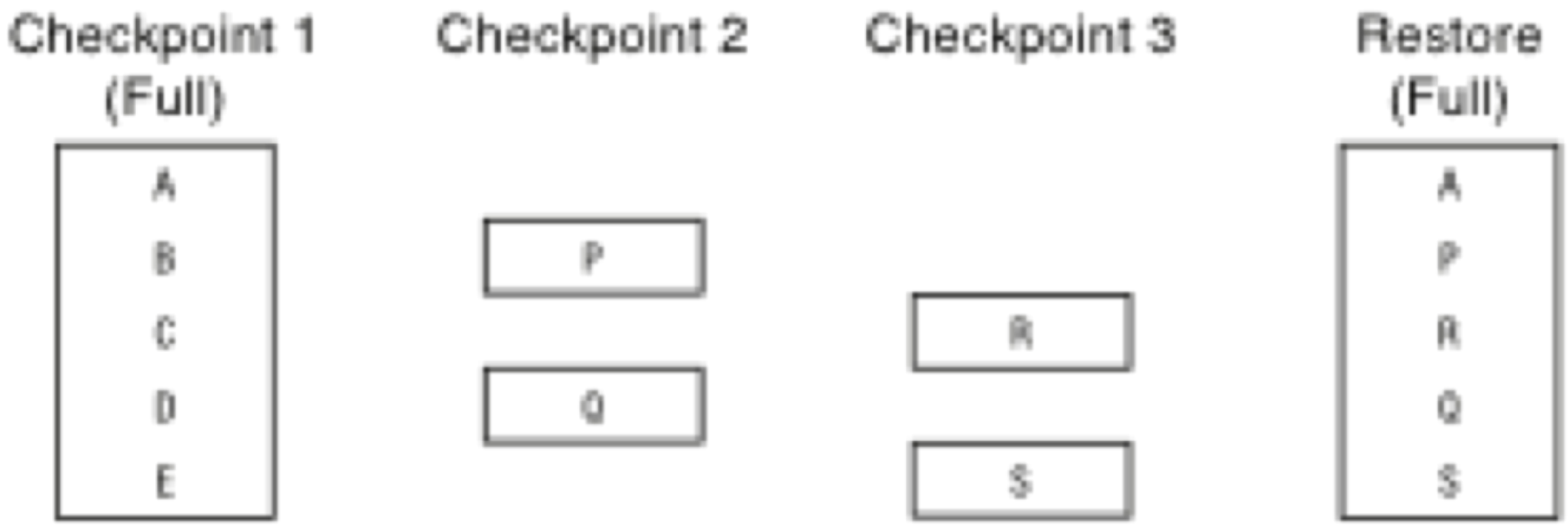
Transparent Checkpoint



Question

- At what point can we resume the execution of a checkpointed program?
  - When the checkpoint starts?
  - When the checkpoint is entirely on disk?

Incremental Checkpoint



Deterministic Debugging

- Can we precisely replay the execution of a multi-threaded process?
  - If process does not have a memory race
- From a checkpoint, record:
  - All inputs and return values from system calls
  - All scheduling decisions
  - All synchronization operations
    - Ex: which thread acquired lock in which order

Process Migration

- What if we checkpoint a process and then restart it on a different machine?
  - Process migration: move a process from one machine to another
  - Special handling needed if any system calls are in progress
    - Where does the system call return to?

Cooperative Caching

- Can we demand page to memory on a different machine?
  - Remote memory over LAN much faster than disk
  - On page fault, look in remote memory first before fetching from disk

Distributed Virtual Memory

- Can we make a network of computers appear to be a shared-memory multiprocessor?
  - Read-write: if page is cached only on one machine
  - Read-only: if page is cached on several machines
  - Invalid: if page is cached read-write on a different machine
- On read page fault:
  - Change remote copy to read-only
  - Copy remote version to local machine
- On write page fault (if cached):
  - Change remote copy to invalid
  - Change local copy to read-write

Recoverable Virtual Memory

- Data structures that survive failures
  - Want a consistent version of the data structure
  - User marks region of code as needing to be atomic
    - Begin transaction, end transaction
  - If crash, restore state before or after transaction

Recoverable Virtual Memory

- On begin transaction:
  - Snapshot data structure to disk
  - Change page table permission to read-only
- On page fault:
  - Mark page as modified by transaction
  - Change page table permission to read-write
- On end transaction:
  - Log changed pages to disk
  - Commit transaction when all mods are on disk
- Recovery:
  - Read last snapshot + logged changes, if committed