Virtual machines

CS503: Operating systems, Spring 2019

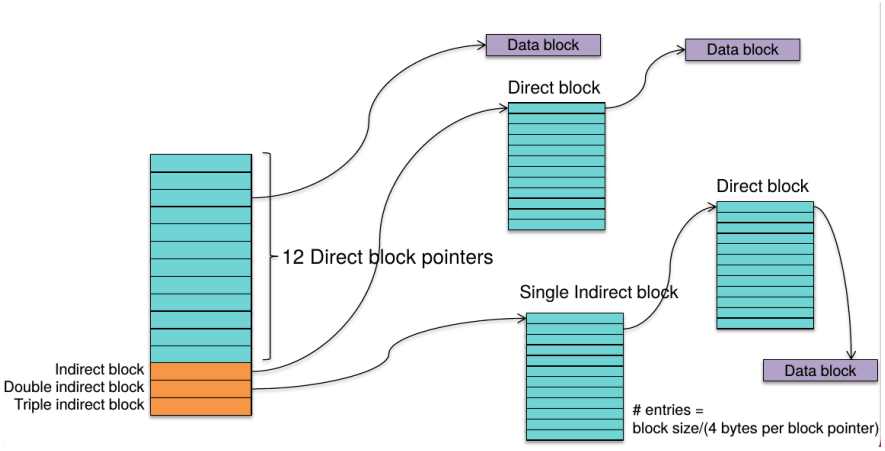
Pedro Fonseca
Department of Computer Science
Purdue University

Admin

- Minor update to the midterm solution sketch
- Regrade request period
  - Mistakes in grading can happen that's why this exists
  - Gradescope or in-person

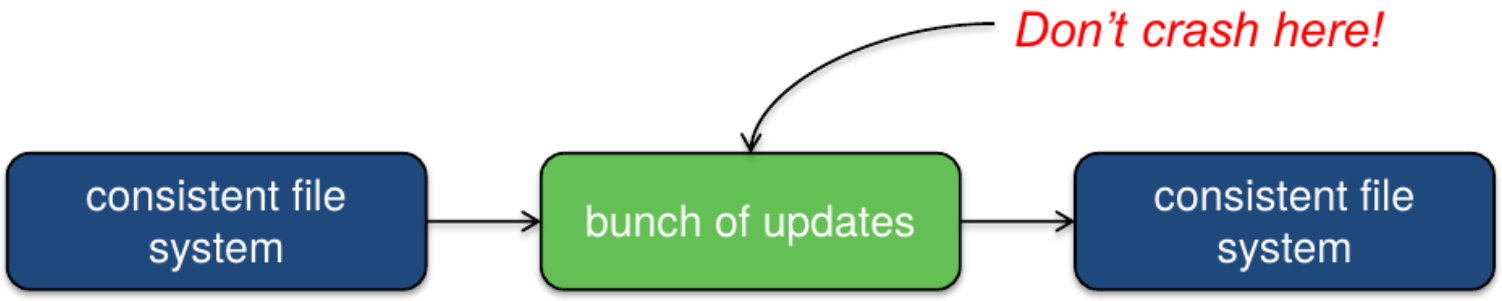
Recall: Linux ext2

- Inodes with direct, indirect, double-indirect, and triple-indirect blocks



Recall: Consistent update problem

- Example: writing a block to a file may require several operations:
  - Inode is:
    - Updated with a new block pointer
    - Updated with a new file size
  - Data free block bitmap is updated
  - Data block contents are written to disk
  - If any of these are not written to disk, we have a file system inconsistency



Recall: Journalling

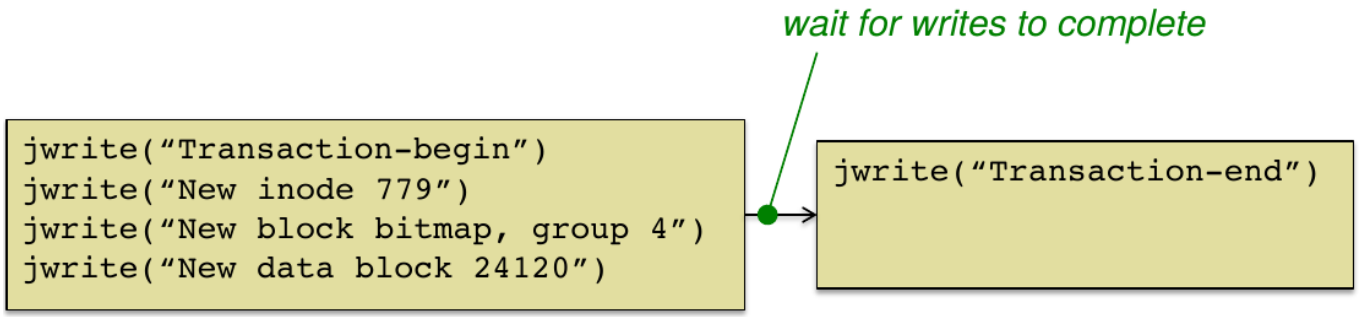
- Journaling == write-ahead logging
- Keep a transaction-oriented journal of changes:
  - Record what the FS is about to do (along with all the data)

```
Transaction-begin
  New inode 779
  New block bitmap, group 4
  New data block 24120
Transaction-end
```

- Once the log entry has been committed to disk then overwrite the real data in disk
  - If all goes well, no need for this entry
  - If there is a crash after the log has been committed:
    - Replay the log on reboot (*redo logging*)
- This scheme is called full data journaling

Recall: Writing the journal

- Writing the journal all at once would be risky
  - We don't know the order the disk will schedule the individual block writes
    - E.g., don't want to risk having a "transaction end" written before the transaction contents
- Steps:
  - Write all blocks, except "transaction end" tags
  - Wait for all the writes to complete
  - Then write "transaction end" tag
- If the log is replayed and a transaction-end is missing, ignore the log



Recall: Cost of journaling

- We're writing everything twice and constantly seeks to the journal area of the disk
- Optimization (metadata journalling, aka ordered journalling):
  - Do not write user-data to journal
  - Only metadata

```
Transaction-begin
  New inode 779
  New block bitmap, group 4
Transaction-end
```

More discussion

- What can go wrong with a file system?
  - Power loss
  - Disk failure?
  - What if disk failure is partial? How can a file system design help / mitigate such problems?
- How to implement file deletion?
  - What is the impact of such implementations?
- Concurrency
- When is data guaranteed to be written to disk?

Virtualization

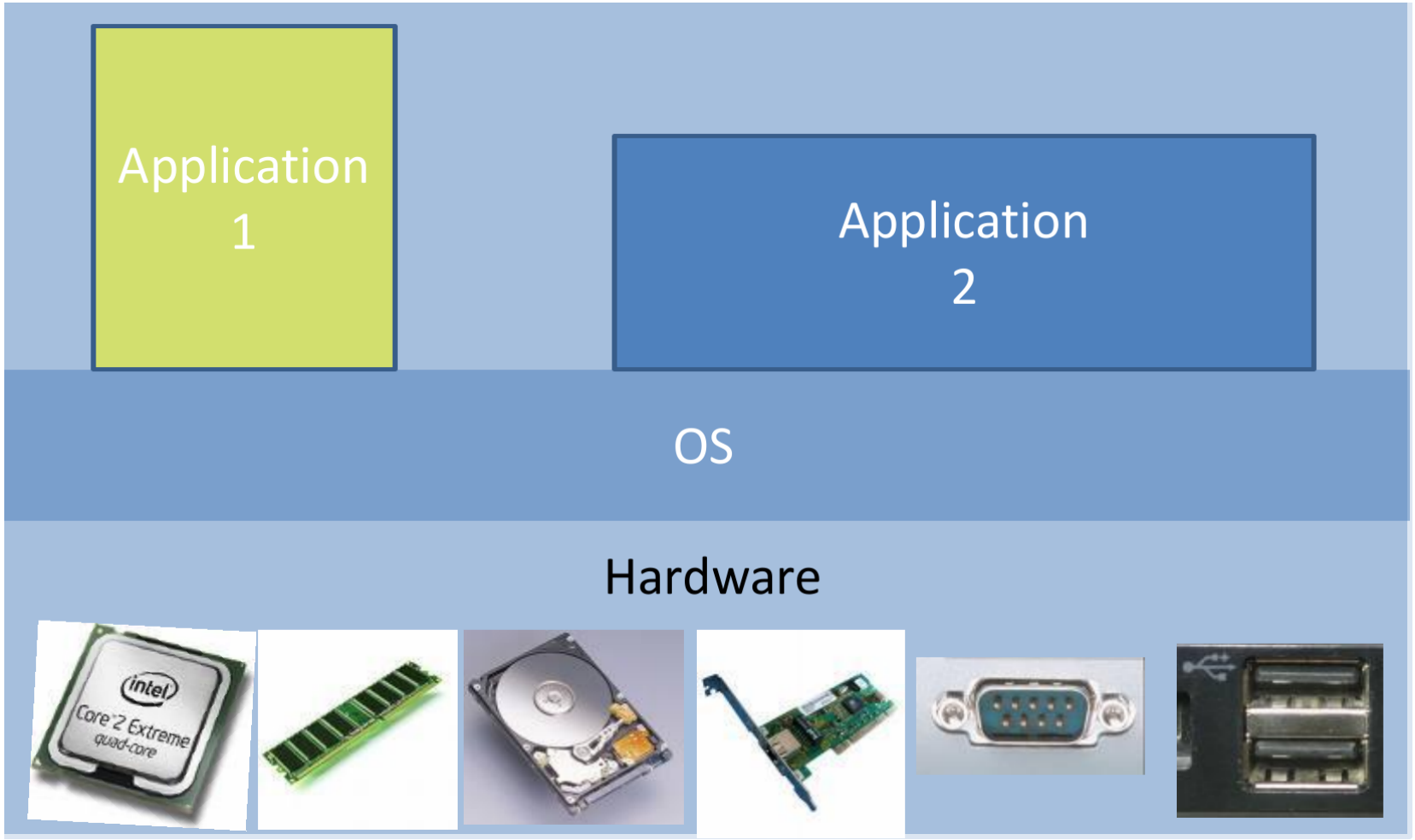
Virtualization

- Term used many times in the course
- One of the roles of the OS
- Q: Examples already discussed?

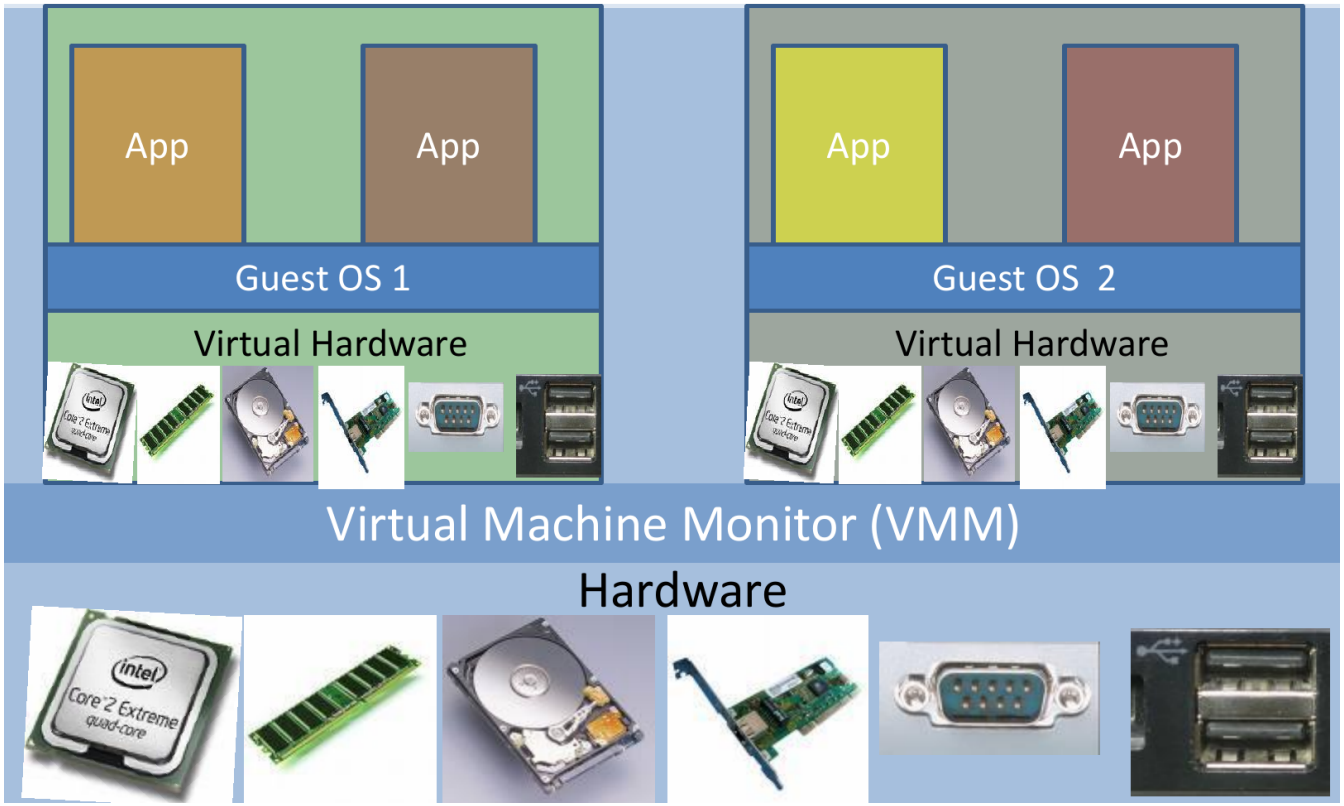
Machine-level virtualization (virtual machines)

- Providing a hardware-like view to each process
- Running an OS inside an OS
- Running multiple OSes on a single physical machine

Traditional



VM virtualization



Topics

- Why VM-based virtualize?
  - Application of VM-based virtualization
- How to virtualize a machine:
  - Interpretation
  - Binary translation
  - Memory virtualization
  - Device emulation (disk, NIC, etc.)

Advantages of virtualization

- Take best of all worlds:
  - Run windows/mac/linux simultaneously
- Increased isolation between applications:
  - Internet VM and development VM
  - Mail server VM and database server VM
  - Support multi-tenant in clouds
- Encapsulation:
  - Suspend and resume
- Emerging applications:
  - Security, reproducibility, monitoring, migration, and legacy systems

How it works: interpretation

- Interpretation (e.g., bochs)
  - Interpret each instruction and emulate it
    - Each instruction is implemented using a c function
  - Slowdown: can be as bad as 50x (or worse)
  - Example: emulation of one asm instruction “inc (%eax)”

```
// incl (%eax):
r = regs[EAX];
tmp = read_mem(r);
tmp++;
write_mem(r, tmp);
```

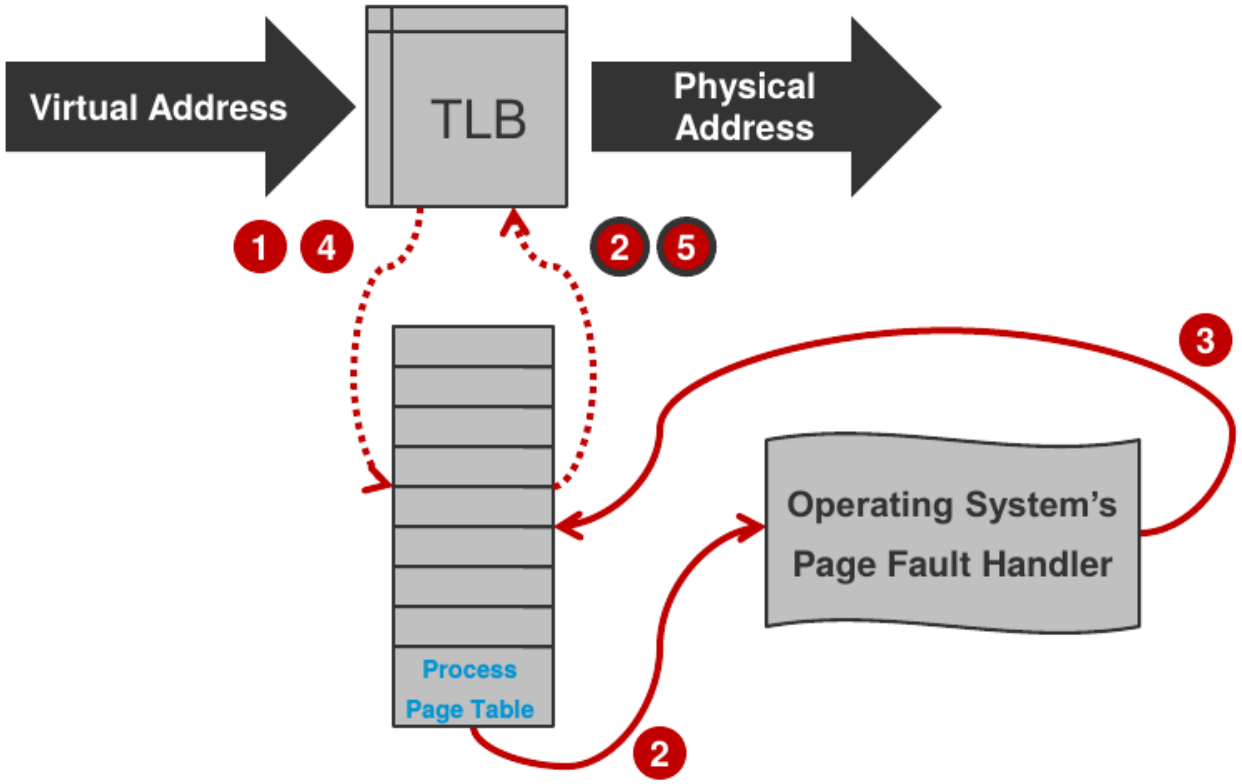
How it works: binary translation

- Binary translation:
  - Translate each guest instruction to the minimal set of host instructions required to emulate it
  - Advantages:
    - Avoid function-call overhead of interpreter-based approach
    - Can re-use translation by maintaining a translation cache
  - Slowdown: ~5-10x

```
// incl (%eax)
leal mem0(%eax), %esi
incl (%esi)
```

Virtual machine monitor

- VMM is placed under OS



VMM

- VMM: Direct execution if possible, binary translation otherwise
  - reg-reg instructions: e.g., movl %eax, %ecx
    - always possible
  - reg-mem instruction: e.g., movl (%eax), %ecx
    - needs memory virtualization
  - IO instructions
    - Need to **trap** and **emulate** + **binary translation**
    - Need to emulate the device in software

VMM trap

- Trap certain operations done by the virtual machine
  - Trap into the VMM
- Interrupts: deliver interrupts to guest OS
- Slowdown: 0-50%

VMM: history

- Popek, Goldberg 1974:
  - https://cs.nyu.edu/courses/fall14/CSCI-GA.3033-010/popek-goldberg.pdf
- An architecture is virtualizable if the set of instructions that could affect the correct functioning of the VMM are a subset of the privileged instructions
  - i.e., all sensitive instructions must always pass control to the VMM
  - x86 was not designed to be virtualizable

VMM: virtualization type

- Full-virtualization using binary-translation (VMware, 1999)
  - Running **unmodified** OS
  - Binary translate sensitive instructions to force them to trap into VMM
  - Most instructions are executed natively
- Para-virtualization (Xen, 2003)
  - Running **modified** OS
  - Fast!
- Hardware support: Intel VT-x and AMD-V (2008)
  - Support for virtualization in hardware for x86
  - Obey the principles required to make hardware virtualizable
  - On modern x86 we no longer require binary translation

CPU organization

- Instruction Set Architecture (ISA) defines the state and operations visible to programmers (registers, mem, machine instructions, etc.)
- ISA is typically divided into two parts:
  - User ISA
  - System ISA

Virtualizing the system ISA

- Typically much more challenging than virtualizing the user ISA
- Q: Why?

Advanced topics and discussion

- Scheduling multiple VMs
  - How does this affect the VMM?
  - Impact of caches
- Suspend and resume
- Migration of VMs
  - What are the uses?
  - What are the challenges?

Summary

- VM-level virtualization:
  - Virtualization paper
- Challenge in multiplexing
- Para-virtualization
- Advanced OS features
  - Checkpoint and restart
  - Migration