CS 491/591: Introduction to Computer Security

Confinement

James Hook

Plan

- Confinement Problem (Lampson)
- Isolation
  - Virtual Machines
  - Sandboxes
- Covert Channels

The Confinement Problem

- Lampson, "A Note on the Confinement Problem", CACM, 1973.

  This note explores the problem of confining a program during its execution so that it cannot transmit information to any other program except its caller. A set of examples attempts to stake out the boundaries of the problem. Necessary conditions for a solution are stated and informally justified.

Discussion

Problem and Threat

- "[The Customer] Create(s) a controlled environment within which another, possibly untrustworthy program [the service], can be run safely. ….
- "…two ways in which the customer may be injured by the service:
  1. it may not perform as advertised
  2. it may lead, i.e. transmit to its owner the input data which the customer gives it."

Possible Leaks

0. If a service has memory, it can collect data, wait for its owner to call it, then return the data
1. The service may write into a permanent file
2. The service may create a temporary file
3. The service may send a message to a process controlled by its owner [via ipc]
4. More subtly, the information may be encoded in the bill rendered for the service…

Possible Leaks (cont)

5. If the system has interlocks which prevent files from being open for writing and reading at the same time, the service can leak data if it is merely allowed to read files which can be written by the owner.

Leak 5 (cont)

The interlocks allow a file to simulate a shared Boolean variable which one program can set and the other can't

Given a procedure `open (file, error)` which does **`goto`** `error` if the file is already open, the following procedures will perform this simulation:

```
procedure settrue (file);
    begin loop1: open (file, loop1) end;
procedure setfalse (file);
    begin close (file) end;
Boolean procedure value (file);
    begin value : = true;
          open (file, loop2);
          value := false;
          close (file);
      loop2:
    end;
```

Leak 5 (cont)

Using these procedures and three files called data, sendclock, and receiveclock, a service can send a stream of bits to another concurrently running program. Referencing the files as though they were variables of this rather odd kind, then, we can describe the sequence of events for transmitting a single bit:

```
sender:      data : = bit being sent;
             sendclock : = true
receiver:    wait for sendclock = true;
             received bit : = data;
             receive clock : = true;
sender:      wait for receive clock = true;
             sendclock : = false;
receiver:    wait for sendclock = false;
             receiveclock : = false;
sender:      wait for receiveclock = false;
```

Leak 6

6. By varying its ratio of computing to input/output or its paging rate, the service can transmit information which a concurrently running process can receive by observing the performance of the system. …

One solution

- Just say no!
- Total isolation: A confined program shall make no calls on any other program
- Impractical

Confinement rule

- Transitivity: If a confined program calls another program which is not trusted, the called program must also be confined.

Classification of Channels:

- Storage
- Legitimate (such as the bill)
- Covert
  - I.e. those not intended for information transfer at all, such as the service program's effect on the system load

- In which category does Lampson place 5?

Mitigation

- Lampson proposes a mitigation strategy for 5
- Confined read makes a copy (this can be done lazily on a conflicting write)

Root Problem:

- Resource sharing enables covert channels
- The more our operating systems and hardware enable efficient resource sharing the greater the risk of covert channels

Lipner's Comments

- 1975 paper discusses how confidentiality models and access control address storage and legitimate channels
- Discussion?
- How does Lipner think BLP fits in?

Lipner's Contribution

- Identifies time as "A difficult problem"
  - "While the storage and legitimate channels of Lampson can be closed with a minimal impact on system efficiency, closing the covert channel seems to impose a direct and unreasonable performance penalty."

Resources

- Lampson, A note on the Confinement Problem, CACM Vol 16, no. 10, October 1973.
  - http://doi.acm.org/10.1145/362375.362389
- Lipner, A Comment on the Confinement Problem, Proceedings of the 5th Symposium on Operating Systems Principles, pp 192 -196 (Nov. 1975)
  - http://doi.acm.org/10.1145/800213.806537

Timing Channel: Kocher

- CRYPTO '96: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Kocher attack

- ```
  Let s[0] = 1
  For k = 0 upto w - 1
    If (bit k of x) is 1 then
      Let R[k] = (s[k] * y) mod n
    Else
      Let R[k] = s[k]
    Let s[k+1] = R[k] * R[k] mod n
  EndFor
  Return R[w-1]
  ```
- Computes R = y^x mod n (x is w bits long)
- Given multiple observations of y, n and time deduce x
  From bits 0..(b-1) find bit b
  ```

Timing channel

- ```
  Let s[0] = 1
  For k = 0 upto w - 1
     If (bit k of x) is 1 then
       Let R[k] = (s[k] * y) mod n
     Else
       Let R[k] = s[k]
     Let s[k+1] = R[k] * R[k] mod n
  EndFor
  Return R[w-1]
  ```

Premise: multiplication mod n takes longer than the assignment
  ```

Basic attack:

- Prework:
  - Study the computation of
    - u * v mod k
  - measure timings for real values (they will probably not be uniform)
- Attack
  - Collect data on (y, n, run time)
  - Guess a bit of x (start with bit 0)
    - Use guess of x to calculate predicted runtimes for algorithm (simulating all intermediate values)
    - If prediction is no better than random guess again
    - If prediction is better than random guess the next bit

Isolation

- Virtual machines
  – Emulate computer
  – Process cannot access underlying computer system, anything not part of that computer system
- Sandboxing
  – Does not emulate computer
  – Alters interface between computer, process

Virtual Machines

- “Third Generation” of Computers
  - First introduced in mid-1960’s
  - Mainstream in early 1970’s
  - IBM 360/67, Honeywell 6000, etc.
- Sources:
  - Formal requirements for Virtualizable Third Generation Architectures, Popek and Goldberg, CACM, vol 17 number 7, July 1974

Virtual Machines

- Original Concept
  - VMM (sometimes called a Control Program or Hypervisor) provided virtualization
    - CP-67, VM/370
  - Family of simple operating systems ran as clients of the VM
    - Single process DOS/360
    - Multi-tasking OS/360
    - Time sharing TSS/360, TSO

Virtual Machine

- "A virtual machine is taken to be an efficient, isolated duplicate of the real machine. … Virtual Machine Monitor (VMM) … a VMM has three essential characteristics:
  1. Provides an environment for programs which is essentially identical with the original machine
  2. Programs run in this environment show at worst only a minor decrease in speed
  3. The VMM is in complete control of system resources"

Popek and Goldberg, 1974

Criteria

- Not all attempts at “3rd Generation” machines succeeded in supporting Virtualization
  - PDP-10 required more emulation
- Popek and Goldberg articulated virtualization criteria

Definitions

- Assumptions:
  - The machine has at least two modes: *user* and *supervisor*
  - The machine has some kind of *fault* (trap) mechanism
- An instruction is *privileged* if it faults (traps) when executed in user mode
- An instruction is *sensitive* if it reveals hidden state of the underlying machine (particularly state about state of privilege)
  - Popek and Goldberg give a more elaborate definition with two types of sensitivity, *control sensitivity* and *behavior sensitivity*
  - Example: an instruction that reveals the physical address of a page in virtual memory is behavior sensitive

P&G Main Theorem

- A virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the privileged instructions

Virtualization for Security

- Schaefer, Gold, Linde and Scheid, Program confinement in KVM/370, 1977, http://doi.acm.org/10.1145/800179.1124633
- Adapt the VMM to include reference monitor to protect security critical resources

Sandbox

- Environment in which actions of process are restricted according to security policy
  - Can add extra security-checking mechanisms to libraries, kernel
    - Program to be executed is not altered
  - Can modify program or process to be executed
    - Similar to debuggers, profilers that add breakpoints
    - Add code to do extra checks (memory access, etc.) as program runs (*software fault isolation*)

Example: Limiting Execution

- Sidewinder
  - Uses type enforcement to confine processes
  - Sandbox built into kernel; site cannot alter it
- Java VM
  - Restricts set of files that applet can access and hosts to which applet can connect

Additional Resources

- R. Wahbe, S. Lucco, T. Anderson, and S. Graham, Efficient Software-based Fault Isolation, http://www.cs.cornell.edu/home/jgm/cs711sp02/sfi.ps.gz
- Christopher Small, MiSFIT: A Tool for Constructing Safe Extensible C++ Systems, http://www.dogfish.org/chris/papers/misfit/misfit-ieee.ps

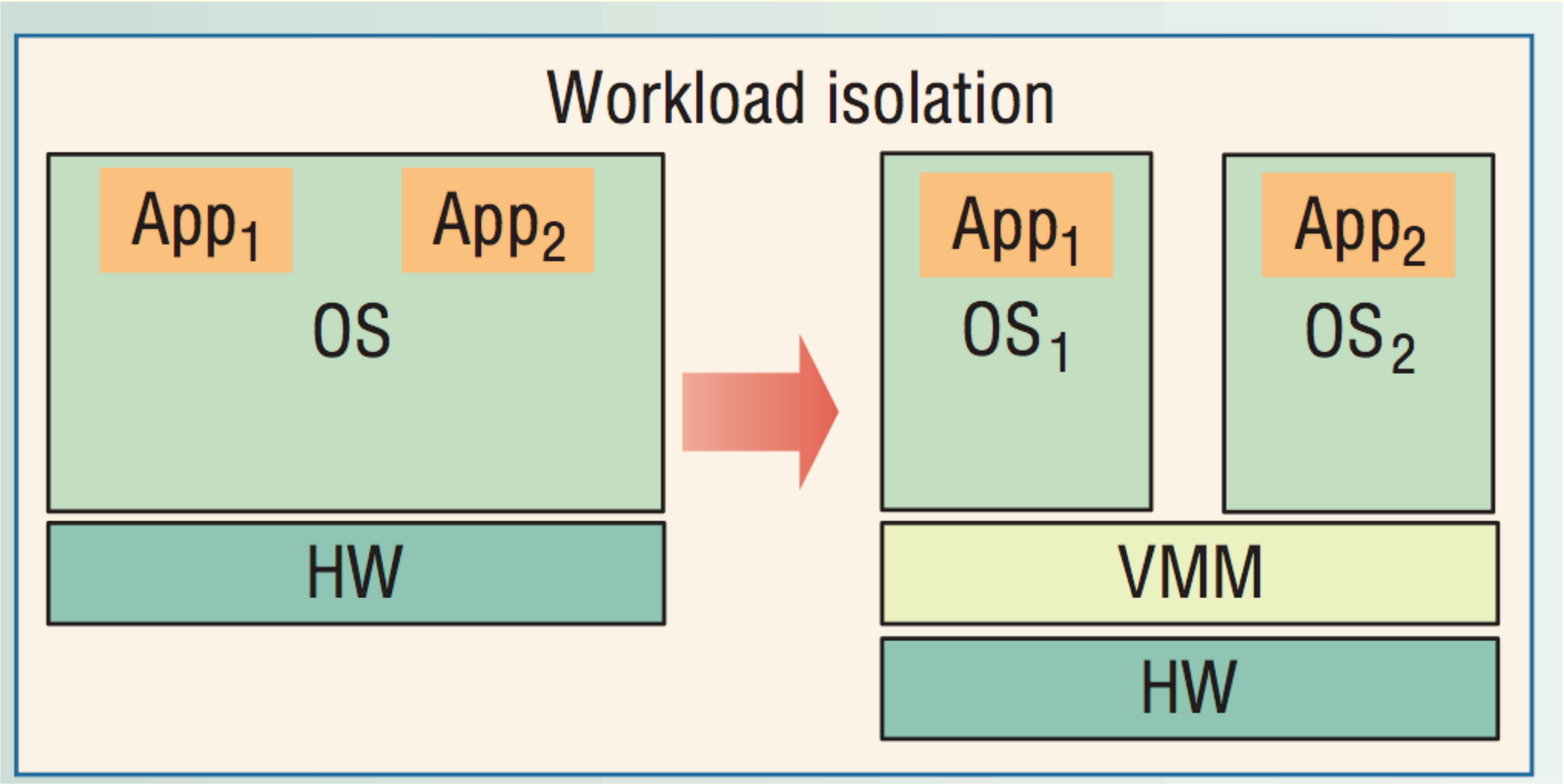
Virtualization Returns

- Intel’s Vanderpool architecture brings Virtual Machines back to the mainstream
- Intel Virtualization Paper
  - (Some figures that follow are taken from the paper)

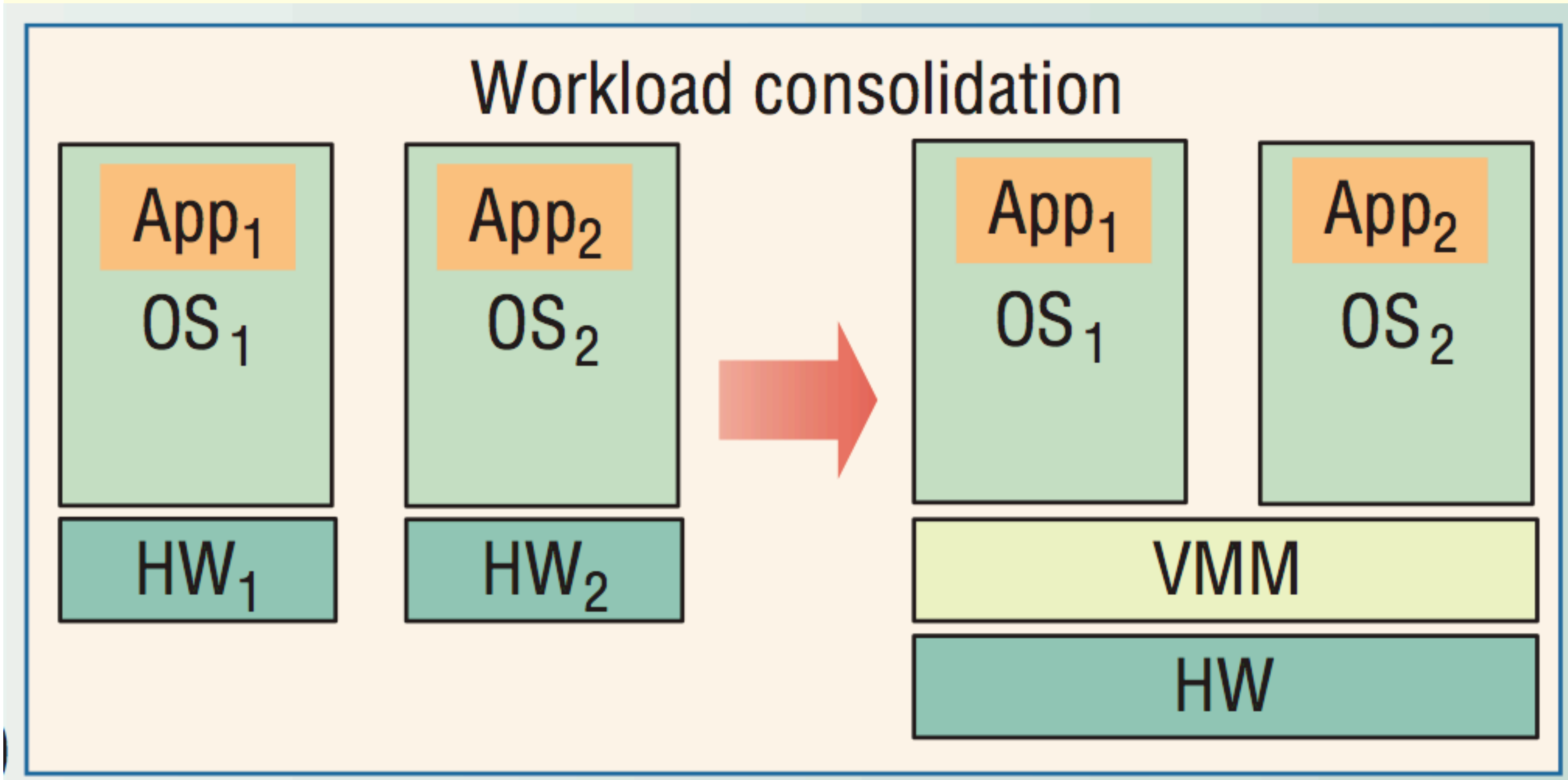
Applications of Virtualization

- Workload isolation
- Workload consolidation
- Workload migration

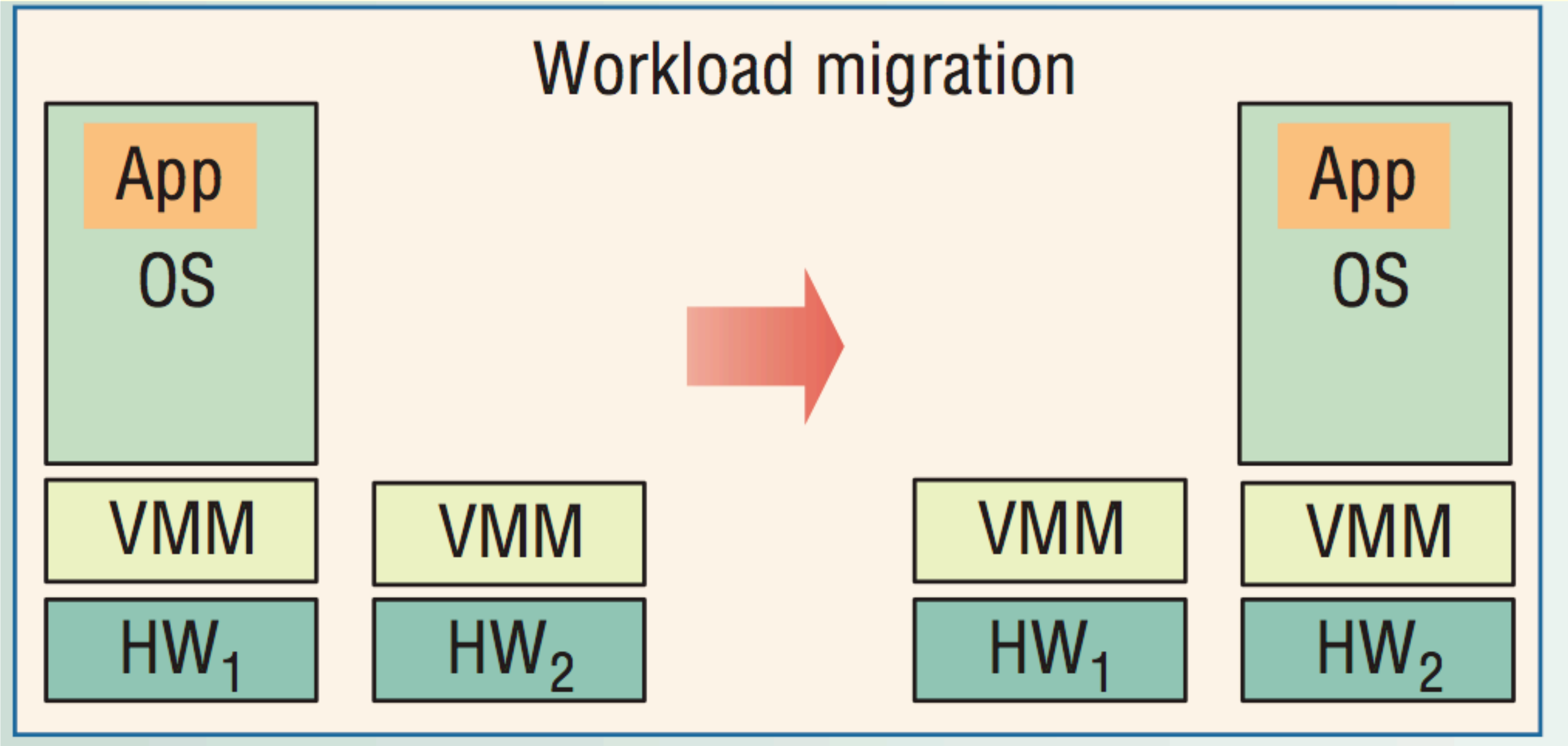
Isolation



Consolidation



Migration


Workload migration
App
OS
VMM
HW1
VMM
HW2
VMM
HW1
App
OS
VMM
HW2

Virtualizing Intel architectures

- As is, Intel architectures do not meet the two requirements:
  - Nonfaulting access to privileged state
    - IA-32 has registers that describe and manipulate the “global descriptor table”
    - These registers can only be set in ring 0
    - They can be queried in any ring without generating a fault
  - This violates rule 2 (all references to sensitive data traps)
- Software products to virtualize Intel hardware had to get around this.
  - Vmware and Virtual PC dynamically rewrite binary code!
  - Xen requires source changes (paravirtualization)

Intel solutions

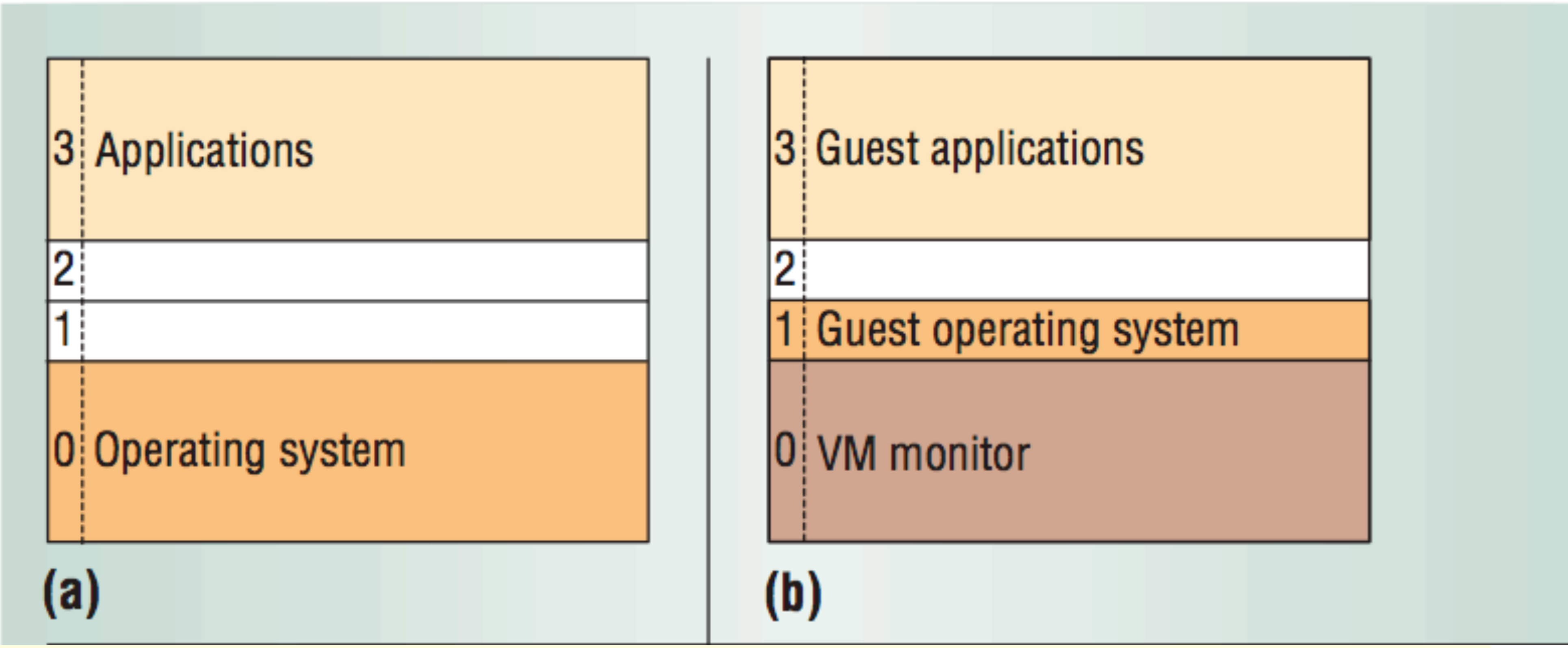
- VT-x, virtualization for IA-32
- VT-i, virtualization for Itanium

- Changed architecture to meet the criteria

Ring aliasing and ring compression

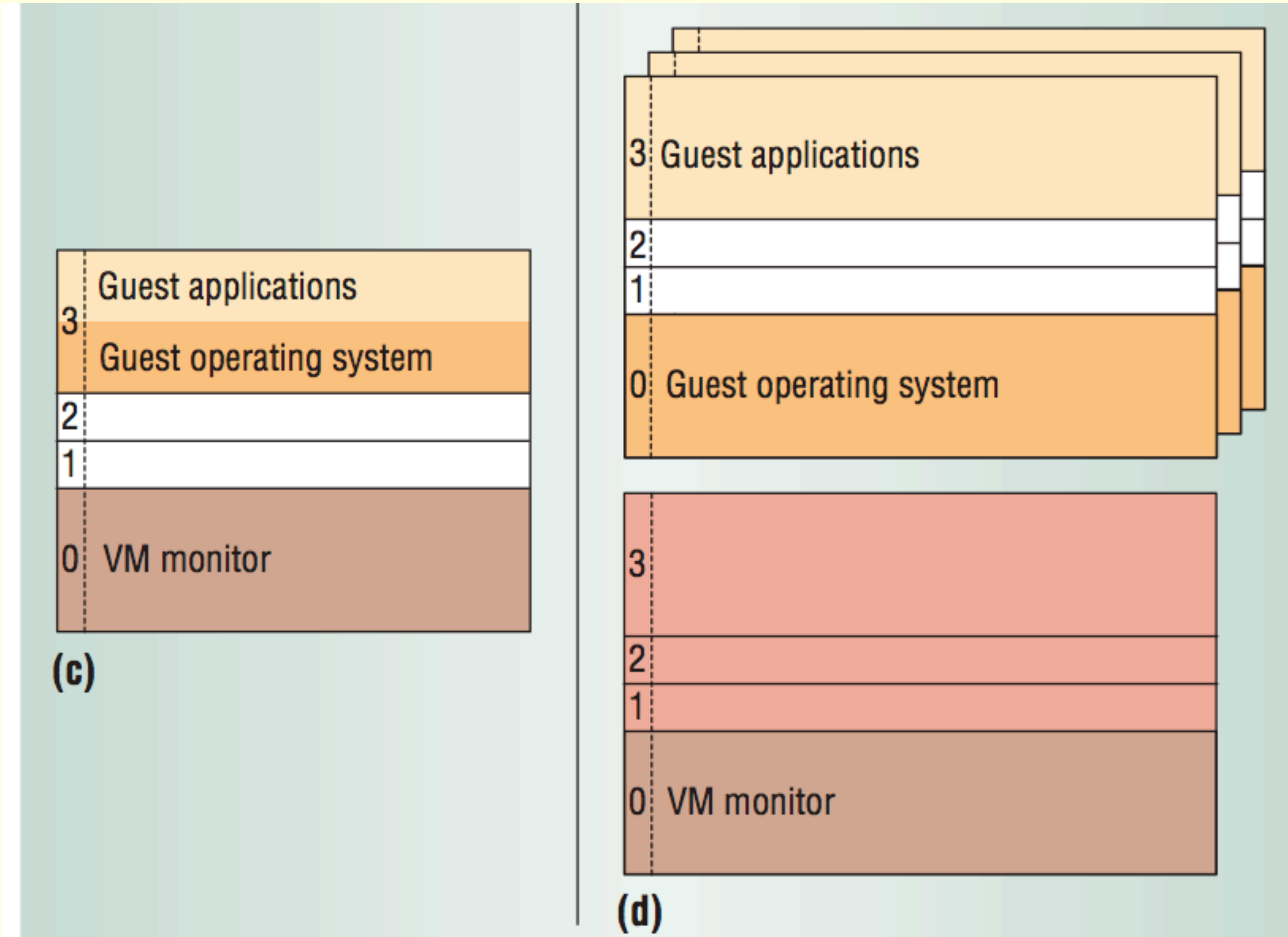
- Solution is to allow guest to run at intended privilege level by augmenting privilege levels.
- See Figure 2(d).

Nonvirtuallized and 0/1/3



- (a) is typical of x86 operating systems
- (b) and (c) give two strategies for virtualization in software

0/3/3 and VT-x



3
Guest applications
Guest operating system
2
1
0
VM monitor
(c)
3
Guest applications
2
1
0
Guest operating system
3
2
1
0
VM monitor
(d)

Nonfaulting access to privileged state

- Two kinds of changes
  - Make access fault to the VM
  - Allow nonfaulting access, but to state under the control of the VMM

Dark Side

- Malware and Virtual Machines
  - SubVirt: Implementing malware with virtual machines,
  - King, Chen, Wang, Verbowski, Wang, Lorch
  - Describes the construction of a "virtual-machine based rootkit" and potential defenses.
  - These appear to be detectable
    - **Compatibility is not transparency: VMM detection myths and realities**
    - T Garfinkel, K Adams, A Warfield, J Franklin - usenix.org