PRINCIPLES OF COMPUTER SYSTEMS DESIGN

CSE130

Winter 2020

File Systems IV - RAID

Virtualisation

Baskin Engineering UC SANTA CRUZ

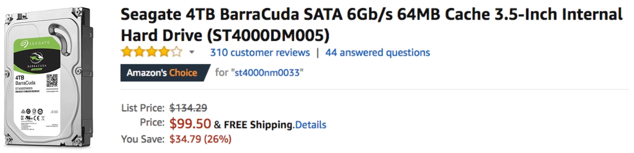
Notices

- **Assignment 5** due **Sunday March 8**
  - If you would like an extra 24 hours to finish it, just ask via e-mail
- 1% Class Wide "Guinea Pig" Credit ☺
  - Reward for having to deal with new Assignment glitches ☹
  - Will be posted to Canvas today

Today's Lecture

- Redundant Arrays of Inexpensive Disks
- Virtualisation

Redundant Arrays of Inexpensive Disks

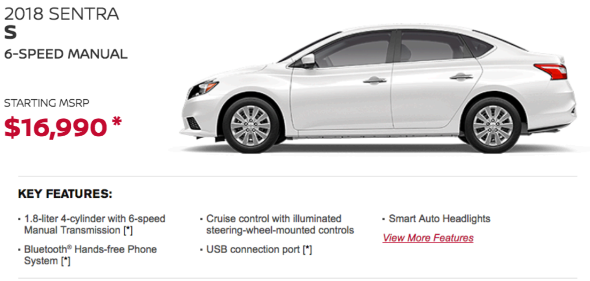


Seagate 4TB BarraCuda SATA 6Gb/s 64MB Cache 3.5-Inch Internal Hard Drive (ST4000DM005)

★★★★☆ 310 customer reviews | 44 answered questions

Amazon's Choice for "st4000nm0033"

List Price: ~~$134.29~~
Price: $99.50 & FREE Shipping.Details
You Save: $34.79 (26%)



2018 SENTRA
S
6-SPEED MANUAL

STARTING MSRP
$16,990 *

KEY FEATURES:

- 1.8-liter 4-cylinder with 6-speed Manual Transmission [*]
- Bluetooth® Hands-free Phone System [*]
- Cruise control with illuminated steering-wheel-mounted controls
- USB connection port [*]
- Smart Auto Headlights

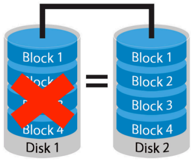
View More Features

Redundant Arrays of Inexpensive Disks

- Three key aspects:
  - How RAID provides greater **reliability** via **redundancy**
  - How RAID provides greater **performance** via **parallelism**
  - Classification of RAID schemes

RAID Reliability

- Multiple disk drives provide **reliability** via **redundancy**
- Increases the **mean time to failure**



- If the mean time to failure of each disk is 100,000 hours
- And the mean time to replace a disk is 10 hours
- The mean time to failure of a mirrored disk system is? **57,000 years!**

RAID Performance

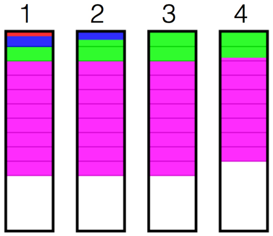
- Several improvements in disk-use techniques involve the use of **multiple disks working cooperatively**
- Disk **striping** uses a group of disks as one storage unit
- Data **mirroring** stores multiple copies of data on separate disks
- **Parallel reads** possible as blocks from one file will be on many disks
- **Increase the throughput** of multiple small file accesses
- **Reduce the response time** of large accesses

RAID Classification of Schemes

- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
- Different **combinations** of **mirroring** and **striping** across disks
- Tradeoff of reliability and performance versus cost

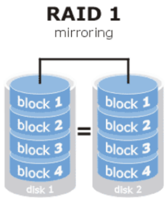
RAID 0 - Performance

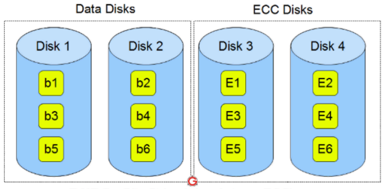
- Simple **striping** of data across multiple disks
- No **mirroring** so **no redundancy**
- Files are broken into stripes and sent to each disk
- Best overall performance characteristics of the single RAID levels
- Inexpensive ☺



RAID 1 & 2 - Reliability

- RAID 1: Simple **mirroring** of disks
- RAID 2: Bit-level **striping** with Hamming Error Correcting Code
  - Not used today as it is too computationally expensive resulting in too high a performance hit - plus it needs twice as many disks as other RAID levels ☹





RAID 2 – Bits Striped. (and stores ECC)

RAID 3 - Reliability

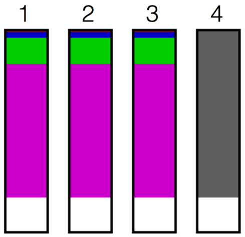
- Data is **striped** across **multiple disks** at **byte level**
- Parity information is sent to a **single dedicated parity disk**
- Failure of **any ONE disk** in the array can be tolerated
- The dedicated parity disk impacts performance because it must be accessed any time anything is written to the array

Parity calculations take the data from two drives and store the result on a third

Parity is computed by XOR'ing a bit from drive 1 with a bit from drive 2 and storing the result on drive 3

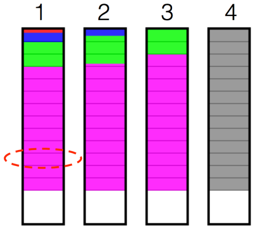
After a failed drive is replaced, the RAID controller rebuilds the lost data from the other two drives

E.g. Drive 1 = **0**, Drive 2 = **1**, Parity = **1** ⊕ **0** = **1**
Loose Drive 1, restored data = **1** ⊕ **1** = **0**



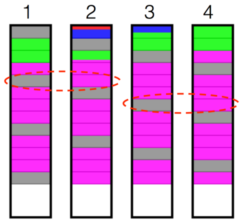
RAID 4

- Data **striped** across **many disks** but in **blocks not bytes**
- Allows new disks to be added easily
- Fault tolerance through a **single dedicated parity disk**
- Like RAID 3 except that it uses blocks instead of bytes for striping
- Failure of **any ONE disk** in the array can be tolerated



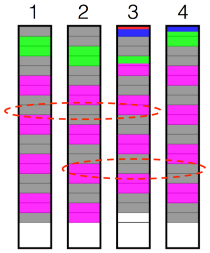
RAID 5

- RAID 4 with **distributed parity** spread across all disk in the array
- Removes the single parity drive "bottleneck"
- Failure of **any ONE disk** in the array can be tolerated
- Improves performance slightly and allows better parallelism
- Parity information for any given block of data is placed on a different drive – ensures fault tolerance



RAID 6

- Essentially RAID 5 +
- Calculates **two sets of parity** information
- Can handle the simultaneous failure of **any TWO disks** in the array
- Performance of RAID 6 is generally slightly worse than RAID 5 as it has additional writes and additional calculations to accommodate the extra parity information



Multiple Nested RAID

- RAID types can be combined to enhance performance or reliability
- Most common is to combine RAID 0 and RAID 1
- Cost is doubling the number of disks required
  - Disks are inexpensive, but power & cooling are not
- **RAID 0+1** = apply striping first, mirroring second
- **RAID 1+0** = apply mirroring first, striping second

Virtualisation

Virtualisation

- **Memory Virtualisation**
  - Each process feels like it has its own address space
  - Created by MMU, configured by OS
- **CPU Virtualisation**
  - Each process feels like it has its own CPU
  - Created by OS preemption and scheduler
- **Storage Virtualisation**
  - Logical view of disks “connected” to a machine
  - External pool of storage
- **Machine Virtualisation**
  - Each OS feels like it has its own machine
  - Created by specialist hardware and software

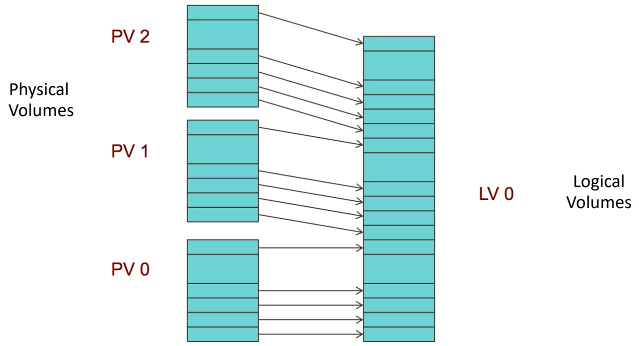
Storage Virtualisation

Logical Volume Manager

- Physical disk
  - Divided into one or more Physical Volumes
- Logical partitions - Volume Groups
  - Created by combining Physical Volumes
    - May span multiple physical disks
  - Can be resized
  - Each can hold a file system
- Storage on physical volumes is divided into “extents”
  - Fixed-size chunks
- Logical volume defined and managed by mapping of logical extents to physical extents
- **Logical Volume Manager** (LVM) takes care of this mapping

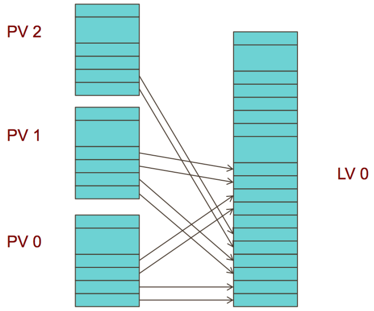
LVM Linear Mapping

- Concatenate multiple physical disks to create a larger disk



LVM Striped Mapping

- Groups from alternate physical volumes mapped to a logical volume
- Multiple physical extents / stripe => improved file transfer bandwidth



LVM Advantages

- Logical disks can be resized while mounted
  - Some file systems (e.g. ext3 on Linux or NTFS) support **dynamic resizing**
- Data can be relocated from one disk to another
- Improved performance (through disk striping)
- Improved redundancy (disk mirroring)
- Snapshots
  - Save the state of the volume at some point in time
  - Allow backups to proceed while the file system is being modified

Machine Virtualisation

Virtual CPUs

- Each process feels like it has its own CPU & memory
  - But cannot execute privileged instructions
  - e.g. modify the MMU or the interval timer, halt the processor, access I/O
- Illusion created by OS preemption, scheduler, and MMU
- User software must “ask the OS” to do critical functions
  - Via **system calls** (Lab 3 & Assignment 5)

Process Virtual Machines

- CPU interpreter / emulator running as a software process
- Pseudo-machine with interpreted instructions
  - 1966: O-code for BCPL
  - 1973: P-code for Pascal
  - 1995: Java Virtual Machine (JIT compilation added)
  - 2002: Microsoft .NET CLR (pre-compilation)
  - 2003: QEMU (dynamic binary translation - used in our Pintos labs)
  - 2008: Dalvik VM for Android
  - 2014: Android Runtime (ART) – ahead of time compilation
- **Advantage**: Run anywhere, sandboxing capability
- **Downside**: Inability to even pretend to access the system hardware
  - Just function calls to access system functions
  - Or “generic” hardware

Machine Virtualisation

- Normally all hardware and I/O managed by one operating system
- Machine virtualization
  - Abstract (virtualize) control of hardware and I/O from the OS
  - **Partition a physical computer to act like several**
- Manipulate memory mappings
- Set system timers
- Access devices
  - Migrate an entire OS & its applications from one machine to another
- 1972: IBM System 370
- 1985: ICL Series 39 / VME
  - ICL Series 39 SX Launched in 1991
  - World’s most powerful mainframe at the time
  - Support for 1,000 VMs on a single machine

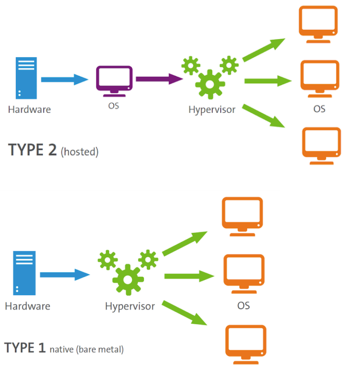
https://www.youtube.com/watch?v=1VU1oDExZWQ

Machine Virtualisation

- An OS is just a collection of code
- We must consider privileged vs. unprivileged instructions
- Regular applications use unprivileged instructions
  - Easy to virtualize
- If regular applications try to execute privileged instructions, an **instruction fault** is raised (a “trap”)
  - Like a page fault in memory management
- VM catches the trap and emulates the instruction
  - **“Trap & Emulate”**

Machine Virtualisation

- Virtualisation
  - OS is unaware it is not running on real hardware
- **Paravirtualisation**
  - OS is aware it that it is a guest OS and has drivers that allow it to access some hardware directly
- Hypervisor
  - **Virtual Machine Monitor** (VMM)
  - Software, hardware, and firmware that supports virtual machines
  - **Xen VMM** arguably most famous
    2003, University of Cambridge Computer Laboratory
    https://en.wikipedia.org/wiki/Xen

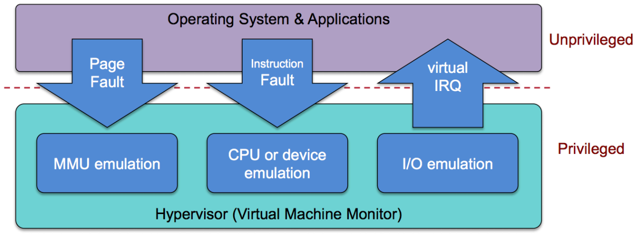


Hypervisor

- **Hypervisor**: Program in charge of virtualization
  - Also known as the **Virtual Machine Monitor** (VMM)
  - Provides the **illusion** that the OS has full access to the hardware
  - Arbitrates access to physical resources
  - Presents a set of virtual device interfaces to each host

Hypervisor

- Application or Guest OS runs until:
  - Privileged instruction trap
  - System interrupt
  - Exception (page fault)
  - Explicit call: VMCALL (Intel) or VMMCALL (AMD)

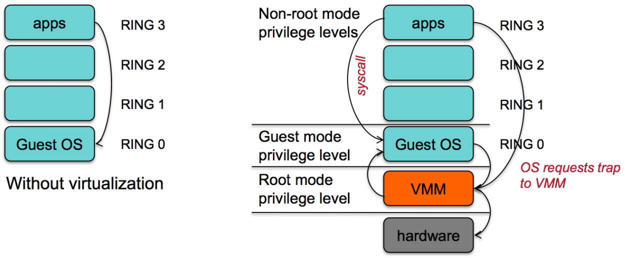


Intel & ARM Didn’t Make VM Easy

- Intel systems prior to Core 2 Duo (2006) did not support trapping privileged instructions
- Most ARM architectures did not trap on certain privileged instructions either
  - Hardware support added in Cortex-A15 (ARMv7 Virtualization Extension): 2011
- Two approaches
  - **Binary translation** (BT)
    - Scan instruction stream on the fly (when page is loaded) and replace privileged instructions with instructions that work with the virtual hardware (VMWare approach)
  - **Paravirtualization**
    - Don’t use non-virtualisable instructions (Xen approach)
    - Invoke hypervisor calls explicitly

Hardware Support for Virtualisation

- Root mode (Intel example)
  - Layer of execution more privileged than the kernel



Architectural Support

- Intel Virtual Technology
- **Guest mode execution**: can run privileged instructions directly
  - System calls do not need to go to the VM
  - Certain privileged instructions are intercepted as VM exits to the VMM
  - Exceptions, faults, and external interrupts are intercepted as VM exits
  - Virtualized exceptions/faults are injected as VM entries

CPU Architectural Support

- **Setup**
  - Turn VM support on/off
  - Configure what controls VM exits
  - Processor state
  - Saved & restored in guest & host areas
- **VM Entry**
  - Go from hypervisor to VM
  - Load state from guest area
- **VM Exit**
  - VM-exit information contains cause of exit
  - Processor state saved in guest area
  - Processor state loaded from host area

Virtualising Memory

- Similar to OS-based virtual memory
  - An OS sees a contiguous address space
  - But it is not necessarily directly tied to physical memory
- Need to virtualise MMU
  - Two levels of translation: **Shadow page tables**
- Host allocates virtual memory for guest
  - Guest treats that as physical memory
- Guest OS cannot access real page tables
  - Access attempts are trapped and emulated
- VMM maps guest “physical memory” settings to actual memory
  - **Second-level address translation** (SLAT) = Nested page tables
- Hardware support in MMU – similar to multilevel page tables
  - Performance enhancement over shadow page tables
- A guest’s physical address is treated as a virtual address

Scheduling VMs

- Each VM competes for a physical CPU
  - Typically # VMs > # CPUs
- VMs need to get scheduled
  - Each VM gets a time slice
- Often **round robin** scheduler – or minor variations
  - Allocate CPU to a single-CPU VM
  - Allocate multiple CPUs to multi-CPU VMs: co-scheduling
- **Strict co-scheduler**:
  - VM with two virtual CPUs gets two real CPUs
- **Relaxed co-scheduler**:
  - If two CPUs are not available, use one
  - CPU affinity: try to run the VM on the same CPU
- VM scheduler controls the level of multiprogramming of VMs

Virtualising Drivers and Events

- Operating systems cannot interact directly with I/O devices
- Device drivers
  - VMM has to multiplex physical devices & create network bridges
  - Virtualize network interfaces (e.g., MAC addresses)
  - Guest OS gets device drivers that interface to an abstract device implementation provided by the VMM
- VMM gets all system interrupts and exceptions
  - Needs to figure out which OS gets a simulated interrupt
  - Simulate those events on the guest OS

Live Migration

- Move VM from one host to another **whilst the VM is running** ☺
- **Select** new host
  - Mirror block devices (for file systems)
  - Initialize VM on new host
- **Initialize**
  - Copy dirty pages to new host iteratively
- **Migrate**
  - Suspend VM on old host
  - Send ARP message to redirect traffic to new host
    - ARP = Address Resolution Protocol (how machines find each other on a network)
  - Synchronize remaining VM state to new host
  - Release state on old host

OS Level Virtualisation

- Not full machine virtualisation
- Multiple instances of the same operating system
  - Each has its own environment
    - Process list, mount table, file descriptors, open file tables, virtual network interfaces, etc.
- Advantage: low overhead (no extra overhead to system calls)
- Example: ICL VME

Virtualisation Recap

- **Memory virtualisation**
  - Each process feels like it has its own address space
  - Created by MMU, configured by OS
- **CPU virtualisation**
  - Each process feels like it has its own CPU
  - Created by OS preemption and scheduler
- **Storage virtualisation**
  - Logical view of disks “connected” to a machine
  - External pool of storage

Handled By
Operating System

- **Machine virtualisation**
  - Each OS feels like it has it’s own machine
  - Created by specialist hardware and software

Handled By
Hypervisor & Hardware

Next Lectures

- Next week
  - Final Examination Prep