CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2021 L24

Mass Storage

Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

FAQ

- LAN: local area network
- WAN: wide area network consisting of many LANs
- Page$_{\text{memory}}$ vs blocks/sectors$_{\text{disk}}$
- Difference among a file, its inode, and inode number?
  - inode number is the index of the inode in the inode table
- Hard links vs symbolic links:
  - Hard links refer to the same inode
  - Symbol link file is a pointer

CS370 Operating Systems

Colorado State University
Yashwant K Malaiya

Reliability & RAIDs

- Various sources

RAID Techniques

- **Striping** uses multiple disks in parallel by splitting data: higher performance (ex. RAID 0)
- **Mirroring** keeps duplicate of each disk: higher reliability (ex. RAID 1)
- **Block parity: One Disk hold** parity block for other disks. A failed disk can be rebuilt using parity. Wear leveling if interleaved (RAID 5, double parity RAID 6).
- Ideas that did not work: Bit or byte level level striping (RAID 2, 3) Bit level Coding (RAID 2), dedicated parity disk (RAID 4).
- Nested Combinations:
  - RAID 01: Mirror RAID 0
  - RAID 10: Multiple RAID 1, striping
  - RAID 50: Multiple RAID 5, striping
  - others

Ch 11 + external

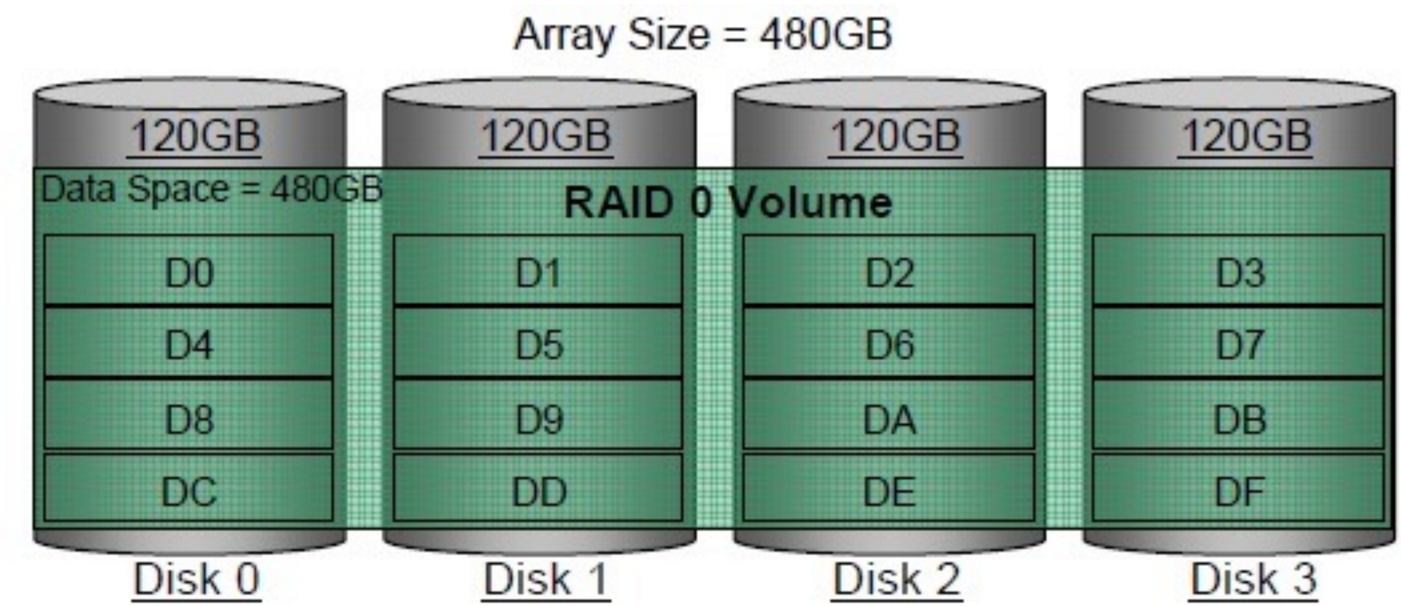
RAID

- Replicate data for availability
  - RAID 0: no replication, data split across disks
  - RAID 1: mirror data across two or more disks
    - Google File System replicated its data on three disks, spread across multiple racks
  - RAID 5: split data across disks, with redundancy to recover from a single disk failure
  - RAID 6: RAID 5, with extra redundancy to recover from two disk failures

Failures and repairs

- If a disk has *mean time to failure (MTTF) of* 100,000 hour.
  - Failure rate is 1/100,000 per hour.
- May be estimated using historical data
- If a disk has a bad data, it may be repaired
  - Copy data from a backup
  - Reconstruct data using available data and some invariant property.
- If data cannot be repaired, it is lost.

RAID 0: Striping



- Additional disks provide additional storage
- No redundancy

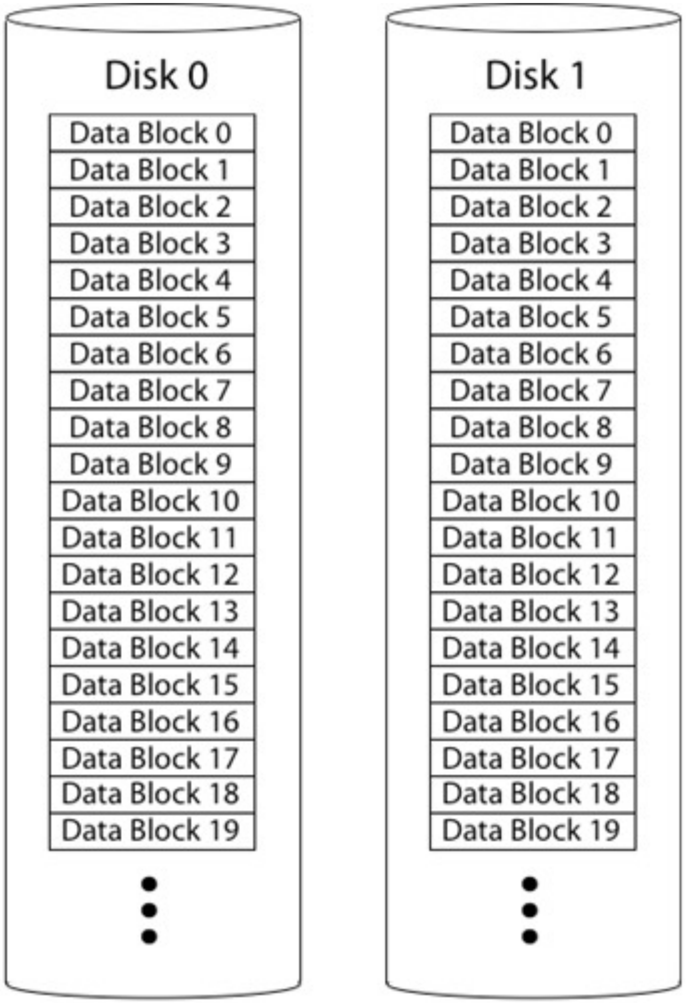
RAID 1: Mirroring

- Replicate writes to both disks
- Reads can go to either disk
- If they fail independently, consider disk with 100,000 hour *mean time to failure* and 10 hour *mean time to repair*
  - probability that two will fail within 10 hours =

    $(2x10) \quad /100{,}000^2$
  - *Mean time to data loss* is

    $100{,}000^2/(2x10) = 500x10^6$

    hours, or 57,000 years!



Colorado State University

Parity

- Data blocks: Block1, block2, block3, ...
- Parity block: Block1 xor block2 xor block3 …

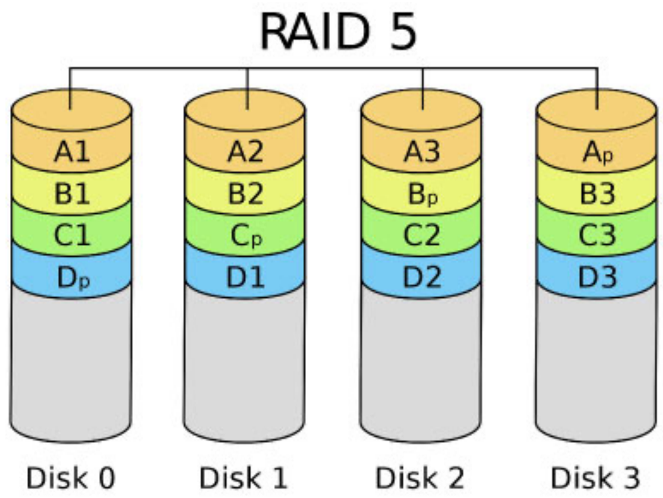
10001101	block1
01101100	block2
11000110	block3

00100111	parity block (*ensures even number of 1s*)

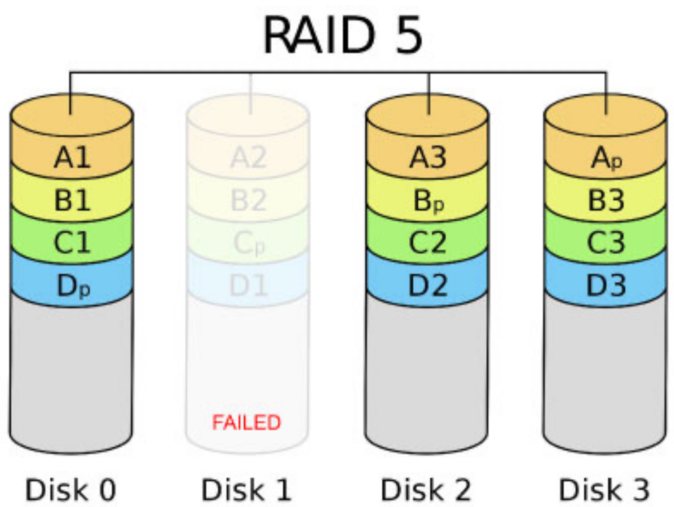
- Can reconstruct any missing block from the others Error-control coding identifies that a block is bad.

Colorado State University

RAID 5: Rotating Parity



Parity blocks Ap, Bp, Cp, Dp distributed across disks.



Time to rebuild depends on disk capacity and data transfer rate

Read Errors and RAID recovery

- Example: RAID 5
  - Each bit has 10^{-15} probability of being bad.
  - 10 one-TB disks, and 1 disk fails
  - Read remaining disks to reconstruct missing data
- Probability of an error in reading 9 TB disks =

  10^{-15}*total bits =10^{-15}* (9 disks * 8 bits * 10^{12} bytes/disk)

  = 7.2%   Thus recovery probability = 92.8%
- Even better:
  - RAID-6: two redundant disk blocks parity plus Reed-Solomon code
  - Can work even in presence of one bad disk, can recover from 2 disk failures
  - Scrubbing: read disk sectors in background to find and fix latent errors

CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Big Data: HDFS and map-reduce

- Various sources, mostly external

Hadoop: Distributed Framework for Big Data

Big Data attributes:

- Large volume: TB -> PB varies with Kryder's law: disk density doubles / 13 months
- Geographically Distributed: minimize data movement
- Needs: reliability, analytic approaches

History:

- Google file system 2003 and Map Reduce 2004 programming lang
- Hadoop to support distribution for the Yahoo search engine project '05, given to Apache Software Foundation '06
- Hadoop ecosystem evolves with Yarn '13 resource management, Pig '10 scripting, Spark '14 distributed computing engine. etc.

- *The Google file system* by Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung (2003)
- *MapReduce: Simplified Data Processing on Large Clusters.* by Jeffrey Dean and Sanjay Ghemawat (2004)

Hadoop: Distributed Framework for Big Data

Recent development.

- Big data: multi-terabyte or more data for an app
- Distributed file system
  - Reliability through replication (Fault tolerance)
- Distributed execution
  - Parallel execution for higher performance

Hadoop: Core components

Hadoop (originally): HDFS + MapReduce

- HDFS: A **d**istributed **f**ile **s**ystem designed to efficiently allocate data across multiple commodity machines, and provide self-healing functions when some of them go down
- MapReduce: A programming framework for processing parallelizable problems across huge datasets using a large number of commodity machines.

- Commodity machines: lower performance per machine, lower cost, perhaps lower reliability compared with special high-performance machines.

Challenges in Distributed Big Data

Common Challenges in Distributed Systems

- **Node Failure**: Individual computer nodes may overheat, crash, have hard drive failures, or run out of memory or disk space.
- **Network issues**: Congestion/delays (large data volumes)**,** Communication Failures.
- **Bad data**: Data may be corrupted, or maliciously or improperly transmitted.
- **Other issues**: Multiple versions of client software may use slightly different protocols from one another.
- **Security**

HDFS Architecture

Hadoop Distributed File System (HDFS):

- HDFS Block size: 64-128 MB ext4: 4KB
- HDFS file size: "Big"
- Single HDFS FS cluster can span many nodes possibly geographically distributed. datacenters-racks-blades
- Node: system with CPU and memory

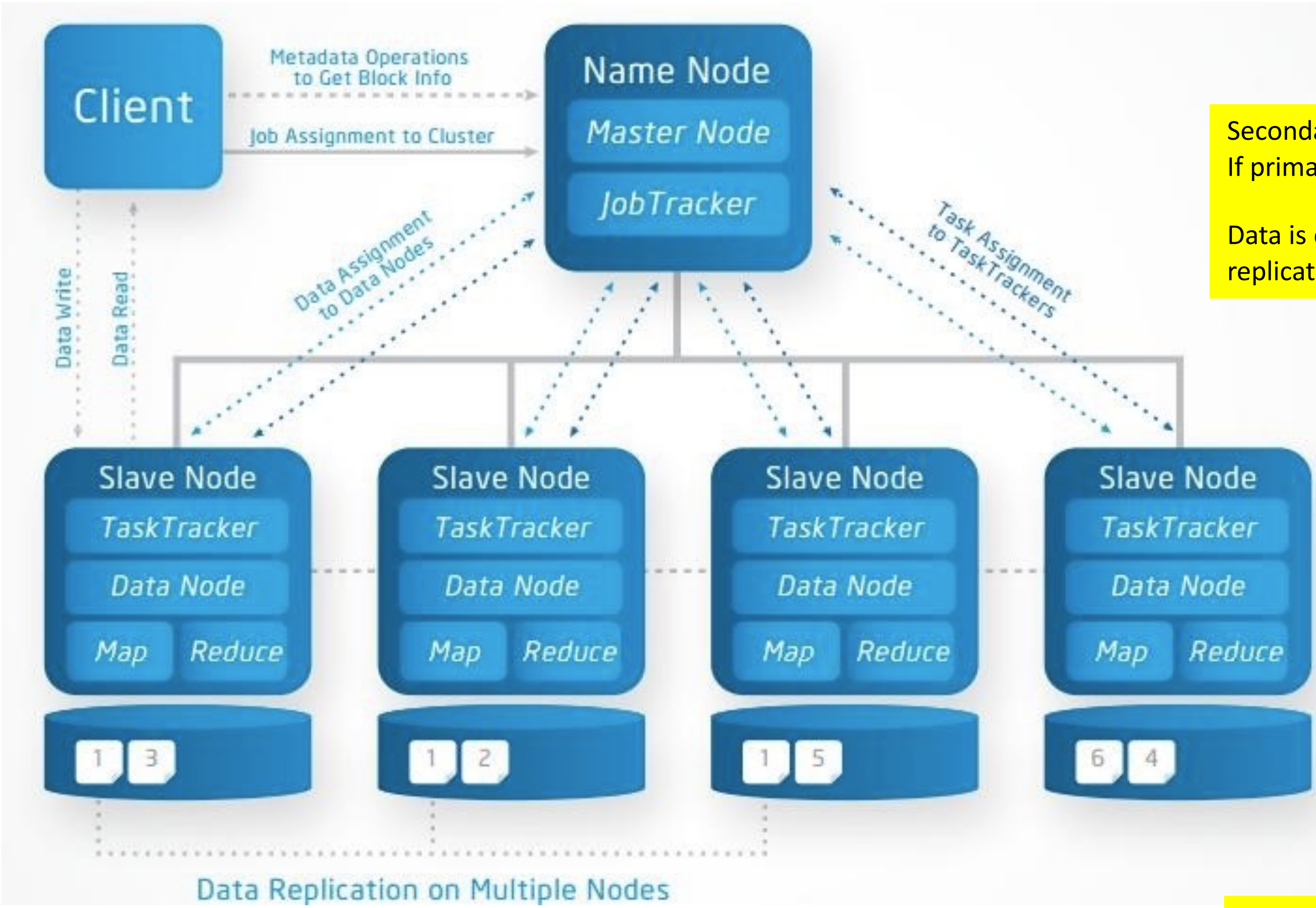
Metadata (corresponding to superblocks, Inodes)

- Name Node: metadata giving where blocks are physically located

Data (files blocks)

- Data Nodes: hold blocks of files (files are distributed)

HDFS Architecture



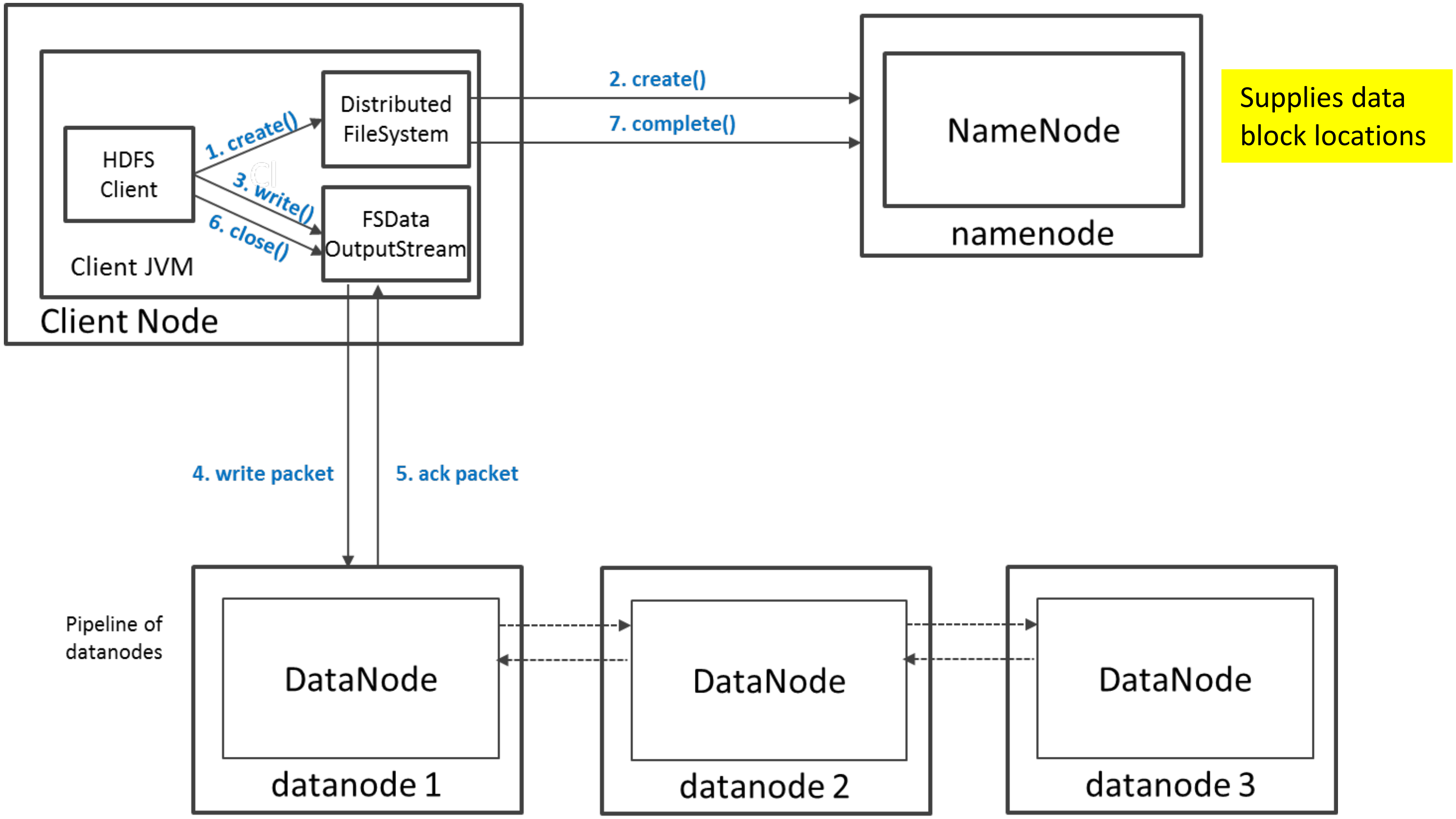
Secondary Name node
If primary fails.

Data is distributed and replicated.

http://a4academics.com/images/hadoop/Hadoop-Architecture-Read-Write.jpg

Slave nodes have been renamed worker nodes.

HDFS Write operation



https://indico.cern.ch/event/404527/contributions/968835/attachments/1123385/1603232/Introduction_to_HDFS.pdf

CERN

HDFS Fault-tolerance

- Disks use error detecting codes to detect corruption.
- Individual node/rack may fail.
- Data Nodes (on slave nodes):
  - data is replicated. Default is 3 times. Keep a copy far away.
  - Send periodic heartbeat (I’m OK) to Name Nodes. Perhaps once every 10 minutes.
  - Name node creates another copy if no heartbeat.

HDFS Fault-tolerance

Name Node (on master node) Protection:

- Transaction log for file deletes/adds, etc. Creation of more replica blocks, when necessary, after a Data Node failure
- Standby name node: namespace backup
  - In the event of a failover, the Standby will ensure that it has read all of the edits from the Journal Nodes and then promotes itself to the Active state
  - Implementation/delay version dependent

Name Node metadata is in RAM as well as checkpointed on disk.
On disk the state is stored in two files:

- fsimage: Snapshot of file system metadata
- editlog: Changes since last snapshot

HDFS Command line interface

- hadoop fs –help
- hadoop fs –ls : List a directory
- hadoop fs mkdir : makes a directory in HDFS
- hadoop fs –rm : Deletes a file in HDFS
- copyFromLocal : Copies data to HDFS from local filesystem
- copyToLocal : Copies data to local filesystem
- Java code can read or write HDFS files (URI) directly

HDFS is on top of a local file system

https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Colorado State University

Distributing Tasks

MapReduce Engine:

- JobTracker splits up the job into smaller tasks(“Map”) and sends it to the TaskTracker process in each node.
- TaskTracker reports back to the JobTracker node and reports on job progress, sends partial results (“Reduce”) or requests new jobs.
- Tasks are run on local data, thus avoiding movement of bulk data.
- Originally developed for search engine implementation.

Hadoop Ecosystem Evolution

- Hadoop YARN: A framework for job scheduling and cluster resource management, can run on top of Windows Azure or Amazon S3.
- Apache spark is more general, faster and easier to program than MapReduce.
    - Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, Berkeley, 2012

tate University

CS370 Operating Systems

Colorado State University
Yashwant K Malaiya
Fall 2021

Virtualization & Containerization

Slides based on

- Various sources

Virtualization

- Why we need virtualization?
- The concepts and terms
- Brief history of virtualization
- Types of virtualization
- Implementation Issues
- Containers

Ch 18 + external

Isolation and resource allocation

Isolation:

- Process: Isolated address space
- Container: Isolated set of processes, files and network
- Virtual Machines (VM): Isolated OSs
- Physically isolated machines

Resource allocation:

- Resources need to be allocated and managed appropriately.

Virtualization

- A Virtual scheme provides a simpler perspective of a Physical scheme. Needs mapping.
  – Example: each process a separate virtual address space.
  – OS allocates physical memory and disk space and handles mapping.
- System ("machine") virtualization
  – A machine needs its own CPU, memory, storage, I/O to run its OS and apps. "Machine" = {CPU, memory, storage, I/O, OS, apps}
  – Needs to be isolated from other machines.
  – "Virtual machines" allocated resources from physical hardware, with allocation done by a Virtual Machine Monitor (VMM or hypervisor.
  – A virtual machine can be "migrated" from one physical system to another.

Virtualization

"Tell that intern that you can't migrate physical machines."

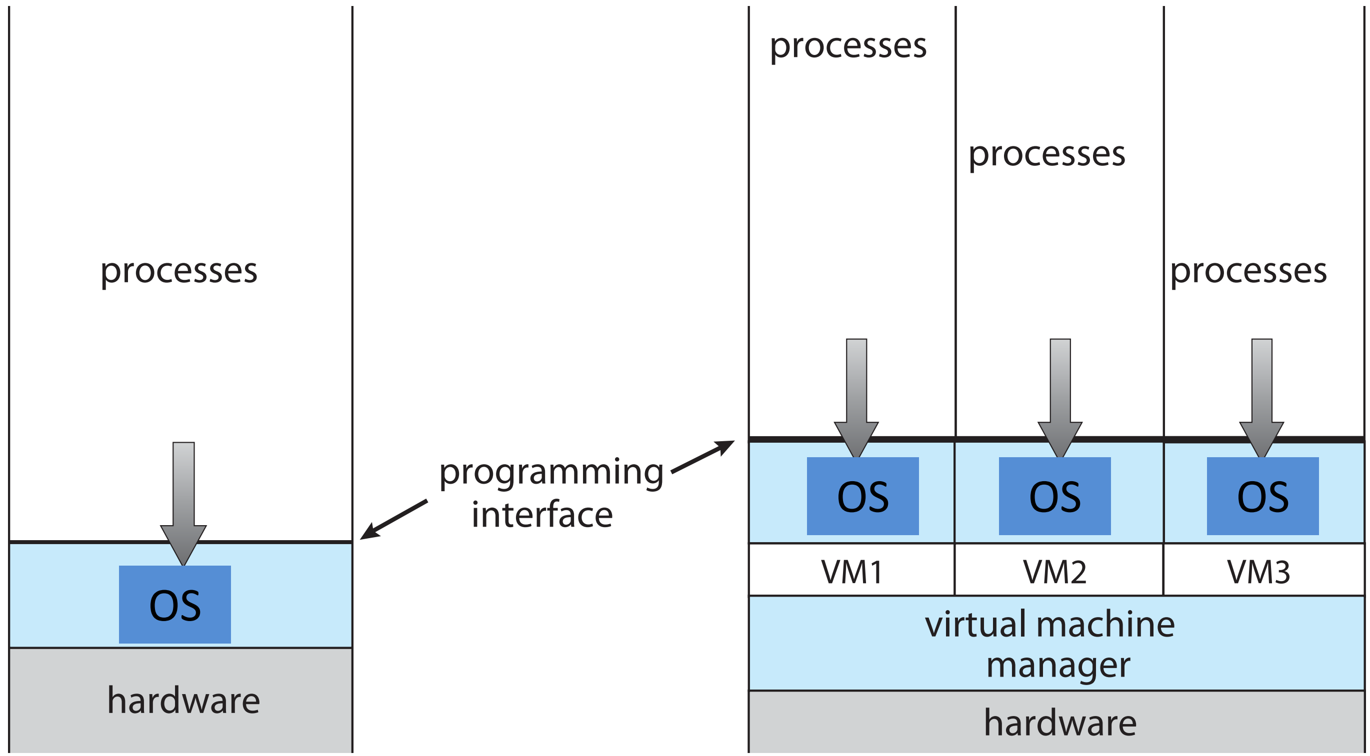
Virtualization

- Processors have gradually become very powerful
- Dedicated servers can be very underutilized (5-15%)
- Virtualization allow a single server to support several virtualized servers: typical consolidation ratio 6:1
- Power cost a major expense for data centers
  - Companies frequently locate their data centers in the middle of nowhere where power cost is low
- If a hardware server crashes, would be nice to migrate the load to another one.
- A key component of cloud computing

Virtual Machines (VM)

- **Virtualization** technology enables a single PC/server to simultaneously run multiple Virtual Machines, with different operating systems or multiple sessions of a single OS.
- A machine with virtualization can host many applications, including those that run on different operating systems, on a single platform.
- The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS.
- The software that enables virtualization is a **virtual machine monitor (VMM),** or **hypervisor.**

Virtual Machines (VM)



Traditional
physical machine

Hypervisor with
virtual machines

Colorado State University

Kinds of Virtual Systems

Virtualization

- Hypervisor based
  - Full virtualization: bare metal hypervisor
  - Para virtualization: modified guest OS
  - Host OS virtualization
- Container system: multiple user space instances
- Environment virtualization
  - Java virtual machine, Dalvic virtual machine
- Software simulation of hardware/ISA
  - Android JDK
  - SoftPC etc.
- Emulation using microcode

Brief history

- Early 1960s IBM experimented with two independently developed hypervisors - SIMMON and CP-40
- Common CPU **modes**: **user** and **supervisor** (*Privileged)*
- In 1974, Popek and Goldberg published a paper which listed what conditions a computer architecture should satisfy to support virtualization efficiently
  - *Privileged instructions: Those that trap if the processor is in user mode and do not trap if it is in system mode (supervisor mode).*
  - *Sensitive instructions: that attempt to change the configuration of resources in the system or whose behavior or result depends on the configuration of resources*
  - Theorem. For any conventional third-generation computer, an effective VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.
  - The x86 architecture that originated in the 1970s did not meet these for requirements for decades.
-

“Strictly Virtualizable”

A processor or mode of a processor is *strictly virtualizable* if, when executed in a lesser privileged mode:

- all instructions that access privileged state trap
- all instructions either trap or execute identically

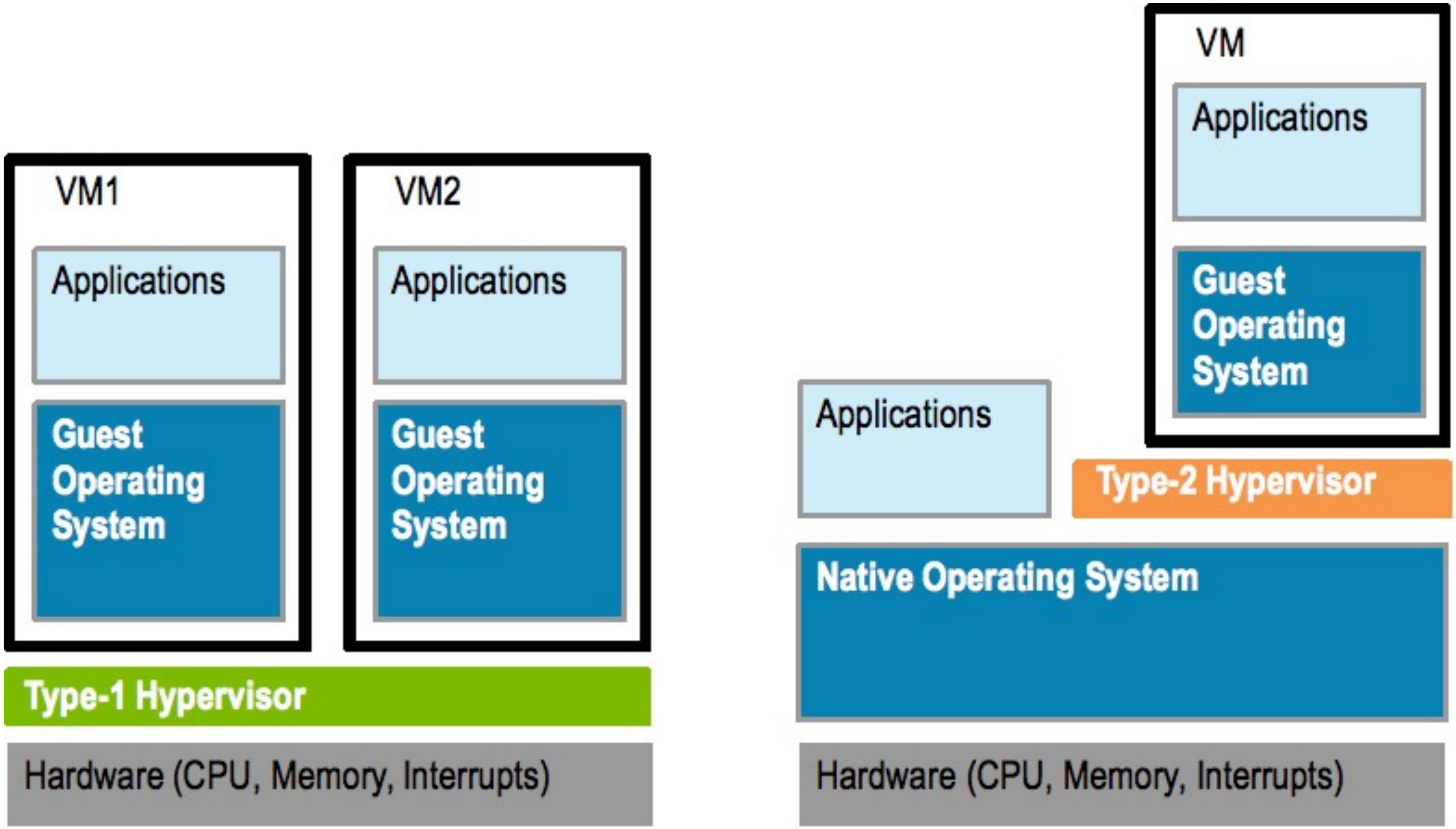
Brief history (recent)

- Stanford researchers developed a new hypervisor and then founded VMware
  - first virtualization solution for x86 in 1999
  - Linux, windows
- Others followed
  - Xen, 2003 University of Cambridge, Xen Project community
  - KVM, 2007 startup/Red Hat
  - VirtualBox (Innotek GmbH/Sun/Oracle) , 2007
  - Hyper-V (Microsoft), 2008

Implementation of VMMs

- **Type 1 hypervisors** - Operating-system-like software built to provide virtualization. Runs on ‘bare metal”.
  - Including VMware ESX, Joyent SmartOS, and Citrix XenServer
- Also includes general-purpose operating systems that provide standard functions as well as VMM functions
  - Including Microsoft Windows Server with HyperV and RedHat Linux with KVM
- **Type 2 hypervisors** - Applications that run on standard operating systems but provide VMM features to guest operating systems
  - Including VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

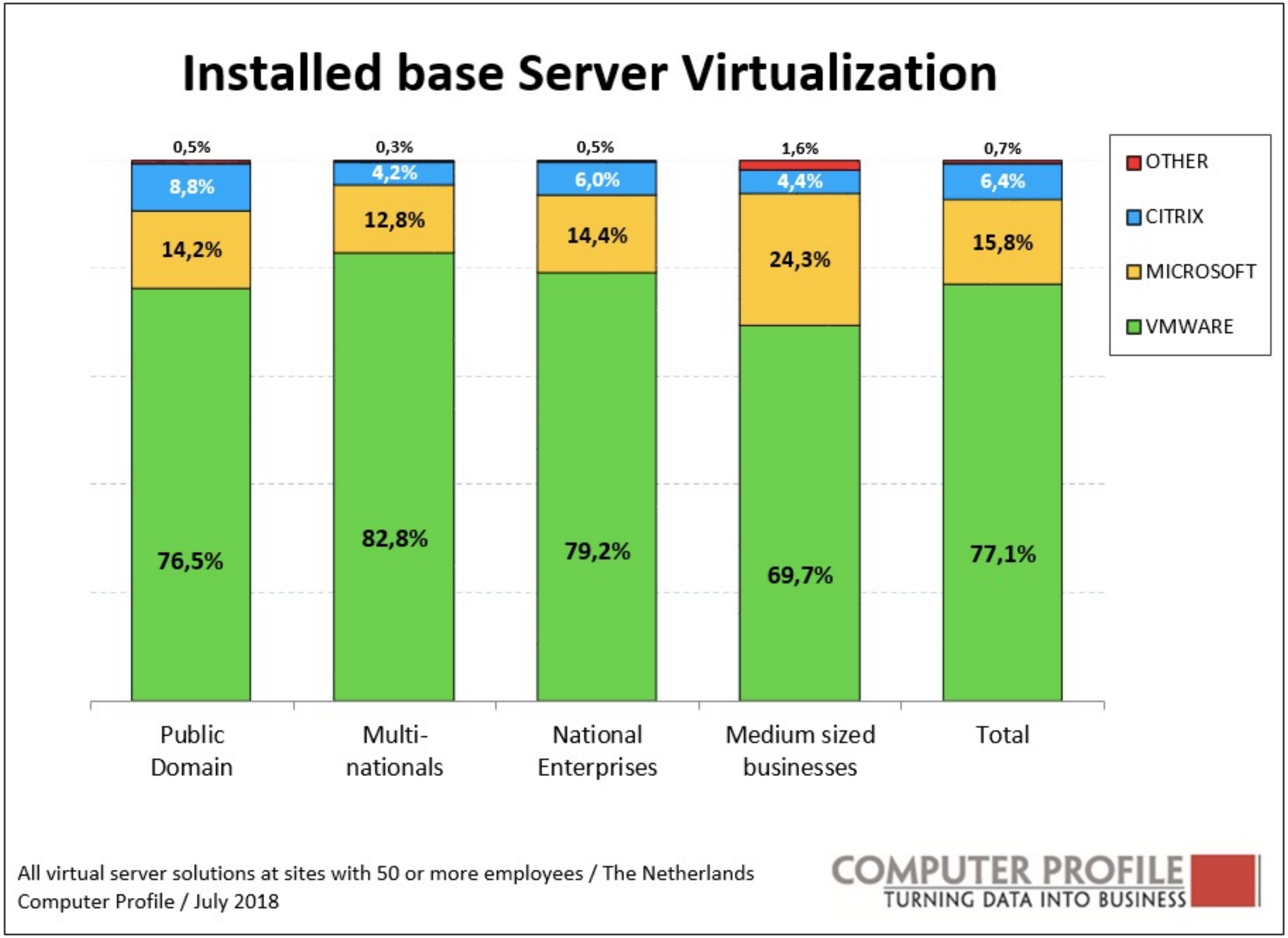
Implementation of VMMs



A higher layer uses services of the lower layers.

https://microkerneldude.files.wordpress.com/2012/01/type1-vs-2.png

Market share



Colorado State University

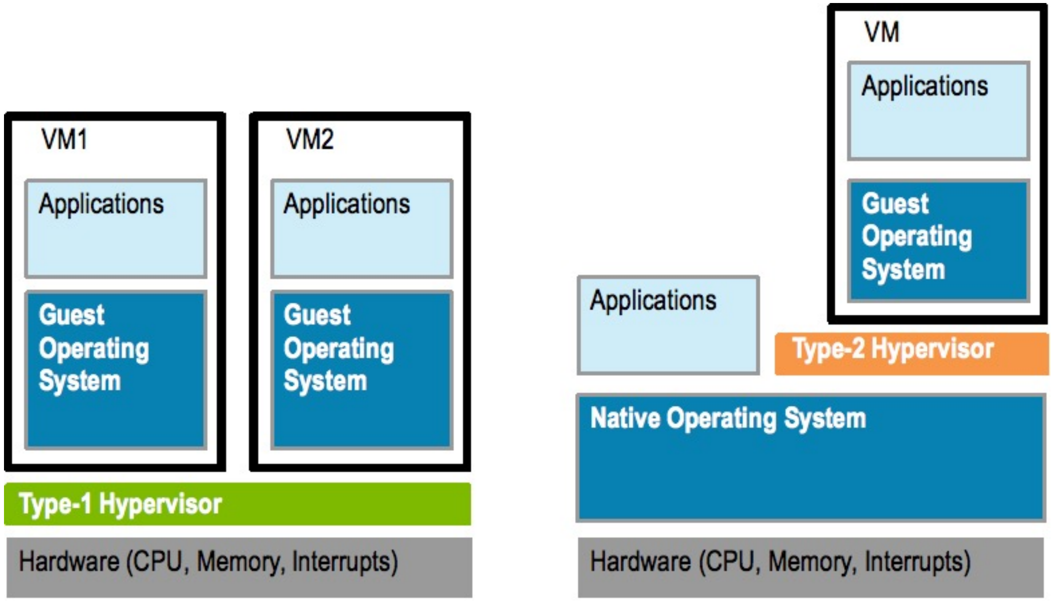
User mode and Kernel (supervisor) mode

- Special instructions:
- Depending on whether it is executed in kernel/user mode
  - "Sensitive instructions"
- Some instructions cause a trap when executed in user-mode
  - "Privileged instructions"
- A machine is virtualizable only if sensitive instructions are a subset of privileged instructions
  - Intel's 386 did not always do that. Several sensitive 386 instructions were ignored if executed in user mode.
- Fixed in 2005 virtualization may need to be enabled using BIOS
  - Intel CPUs: VT (Virtualization Technology)
  - AMD CPUs: SVM (Secure Virtual Machine)

Virtualization support

- Terminology:
  - Guest Operating System
    - The OS running on top of the hypervisor
  - Host Operating System
    - For a type 2 hypervisor: the OS that runs on the hardware ¨executions
- Create environments in which VMs can be run
- When a guest OS is started in an environment, continues to run until it causes an exception and traps to the hypervisor
  - For e.g., by executing an I/O instruction
- Set of operations that trap is controlled by a hardware bit map set by hypervisor
  - trap-and-emulate approach becomes possible

Implementation of VMMs



What problems do you see?

- What mode does hypervisor run in? Guest OSs?
- Are Guest OSs aware of hypervisor?
- How is memory managed?
- How do we know what is the best choice?

Virtual Machine (VM) as a software construct

- Each VM is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports.
- Once the VM is created it can be activated on like a physical server, loaded with an operating system and software solutions, and used just like a physical server.
- Unlike a physical server, VM only sees the resources it has been configured with, not all of the resources of the physical host itself.
- The hypervisor facilitates the translation and I/O between the virtual machine and the physical server.

Virtual Machine (VM) as a set of files

- Configuration file describes the attributes of the virtual machine containing
  - server definition,
  - how many virtual processors (vCPUs)
  - how much RAM is allocated,
  - which I/O devices the VM has access to,
  - how many network interface cards (NICs) are in the virtual server
  - the storage that the VM can access
- When a virtual machine is instantiated, additional files are created for logging, for memory paging etc.
- Copying a VM produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself

Virtualization benefits

- Run multiple, OSes on a single machine
  - **Consolidation**, app dev, …
- Security: Host system protected from VMs, VMs protected from each other
  - Sharing though shared file system volume, network communication
- Freeze, suspend, running VM
  - Then can move or copy somewhere else and **resume**
    - **Live migration**
  - Snapshot of a given state, able to restore back to that state
  - **Clone** by creating copy and running both original and copy
- Hence – cloud computing

Building Block – Trap and Emulate

- VM needs two modes: both in real user mode
  - virtual user mode and virtual kernel mode
- When Guest OS attempts to execute a privileged instruction, what happens?
  - Causes a trap
  - VMM gains control, analyzes error, executes operation as attempted by guest
  - Returns control to guest in user mode
  - Known as **trap-and-emulate**
- Trap-and-emulate was the technique used for implementing floating point instructions in CPUs without floating point coprocessor

Handling sensitive instructions

- Some CPUs didn't have clean separation between privileged and non-privileged instructions
  - Sensitive instructions
    - Consider Intel x86 `popf` instruction
    - If CPU in privileged mode -> all flags replaced
    - If CPU in user mode -> on some flags replaced
      - No trap is generated
- Binary translation (complex) solves the problem
  1. If guest VCPU is in user mode, guest can run instructions natively
  2. If guest VCPU in kernel mode (guest believes it is in kernel mode)
     1. VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
     2. Special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)
  3. Cached translations can reduce overhead
- Not needed in newer processors with virtualization support.

Type 1 Hypervisors

- Run on top of *bare metal*
- Guest OSs believe they are running on bare metal, are unaware of hypervisor
  - are not modified
  - Better performance
- Choice for data centers
  - Consolidation of multiple OSes and apps onto less HW
  - Move guests between systems to balance performance
  - Snapshots and cloning
- Hypervisor creates runs and manages guest OSes
  - Run in kernel mode
  - Implement device drivers
  - provide traditional OS services like CPU and memory management
- Examples: VMWare esx (dedicated) , Windows with Hyper-V (includes OS)

Type 2 Hypervisors

- Run on top of host OS
- VMM is simply a process, managed by host OS
  - host doesn’t know they are a VMM running guests
- poorer overall performance because can’t take advantage of some HW features
- Host OS is just a regular one
  - Individuals could have Type 2 hypervisor (e.g. Virtualbox) on native host (perhaps windows), run one or more guests (perhaps Linux, MacOS)

Full vs Para-virtualization

- Full virtualization: Guest OS is unaware of the hypervisor. It thinks it is running on bare metal.
- Para-virtualization: Guest OS is modified and optimized. It sees underlying hypervisor.
  - Introduced and developed by Xen
    - Modifications needed: Linux 1.36%, XP: 0.04% of code base
  - Does not need as much hardware support
  - allowed virtualization of older x86 CPUs without binary translation
  - Not used by Xen on newer processors

CPU Scheduling

- One or more virtual CPUs (vCPUs) per guest
  - Can be adjusted throughout life of VM
- When enough CPUs for all guests
  - VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
- Usually not enough CPUs (CPU overcommitment)
  - VMM can use scheduling algorithms to allocate vCPUs
  - Some add fairness aspect

CPU Scheduling (cont)

- Oversubscription of CPUs means guests may get CPU cycles they expect
  - Time-of-day clocks may be incorrect
  - Some VMMs provide application to run in each guest to fix time-of-day