Operating Systems (A)
(Honor Track)

Lecture 21: Distributed Systems

Yao Guo (郭耀)

Peking University

Fall 2021

Acknowledgements: Prof. Xiangqun Chen & Tao Wang at PKU and Prof. Yuanyuan Zhou at UCSD

Buzz Words

- Client/Server
- Naming
- Remote procedure call (RPC)
- Network File System (NFS)
- Marshal
- Virtualization
- Paravirtualization
- Xen

This Lecture

Distributed Systems

Overview

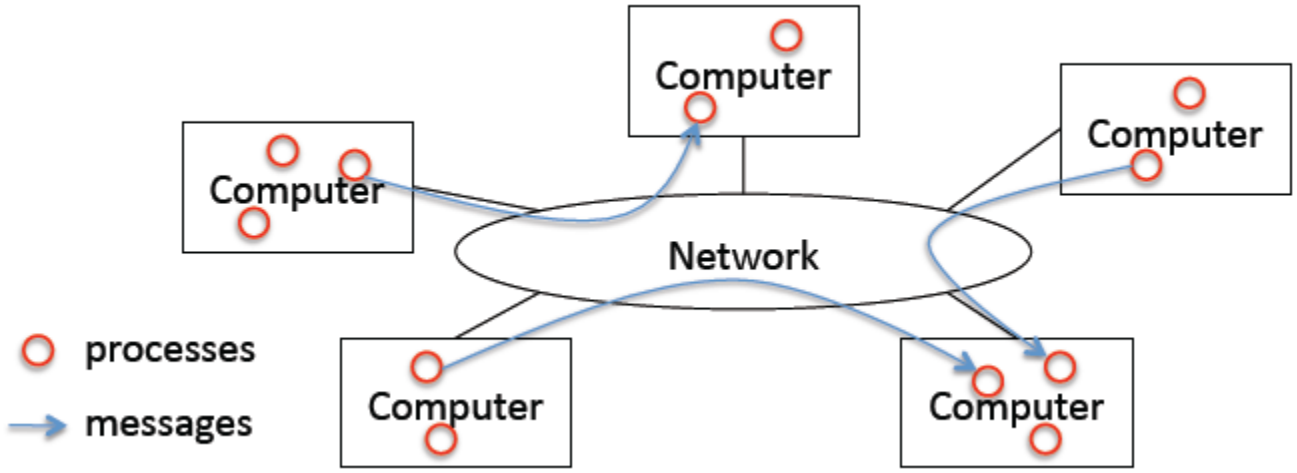
Remote procedure call (RPC)

Network file system (NFS)

Virtualization and Cloud

What is a Distributed System?

- Cooperating processes in a computer network
- Degree of integration
  - Loose: Internet applications, email, web browsing
  - Medium: remote execution, remote file systems
  - Tight: distributed file systems



Benefits

- Performance: parallelism across multiple nodes
  - Google file systems, BigTable, MapReduce, hadoop, etc
- Reliability and fault tolerance
  - Redundancy
  - E.g.: Google search engine
- Scalability by adding more nodes

Clients and Servers

- The prevalent model for structuring distributed computation is the client/server paradigm
- A server is a program (or collection of programs) that provide a service (file server, name service, etc.)
  - The server may exist on one or more nodes
  - Often the node is called the server, too, which is confusing
- A client is a program that uses the service
  - A client first binds to the server (locates it and establishes a connection to it)
  - A client then sends requests, with data, to perform actions, and the servers sends responses, also with data

Naming

- Name systems in network
  - often hierarchical
  - pku.edu.cn is a “domain”
- Network Address (Internet IP address)
  - 162.105.4.131 – 162.105.4.**
  - 128.174.240.**
- Physical Network Address
  - Ethernet address or Token Ring address
- Address processes/ports within system (host, id) pair
- Domain name service (DNS) specifies naming structure of hosts and provides resolution of names to network address

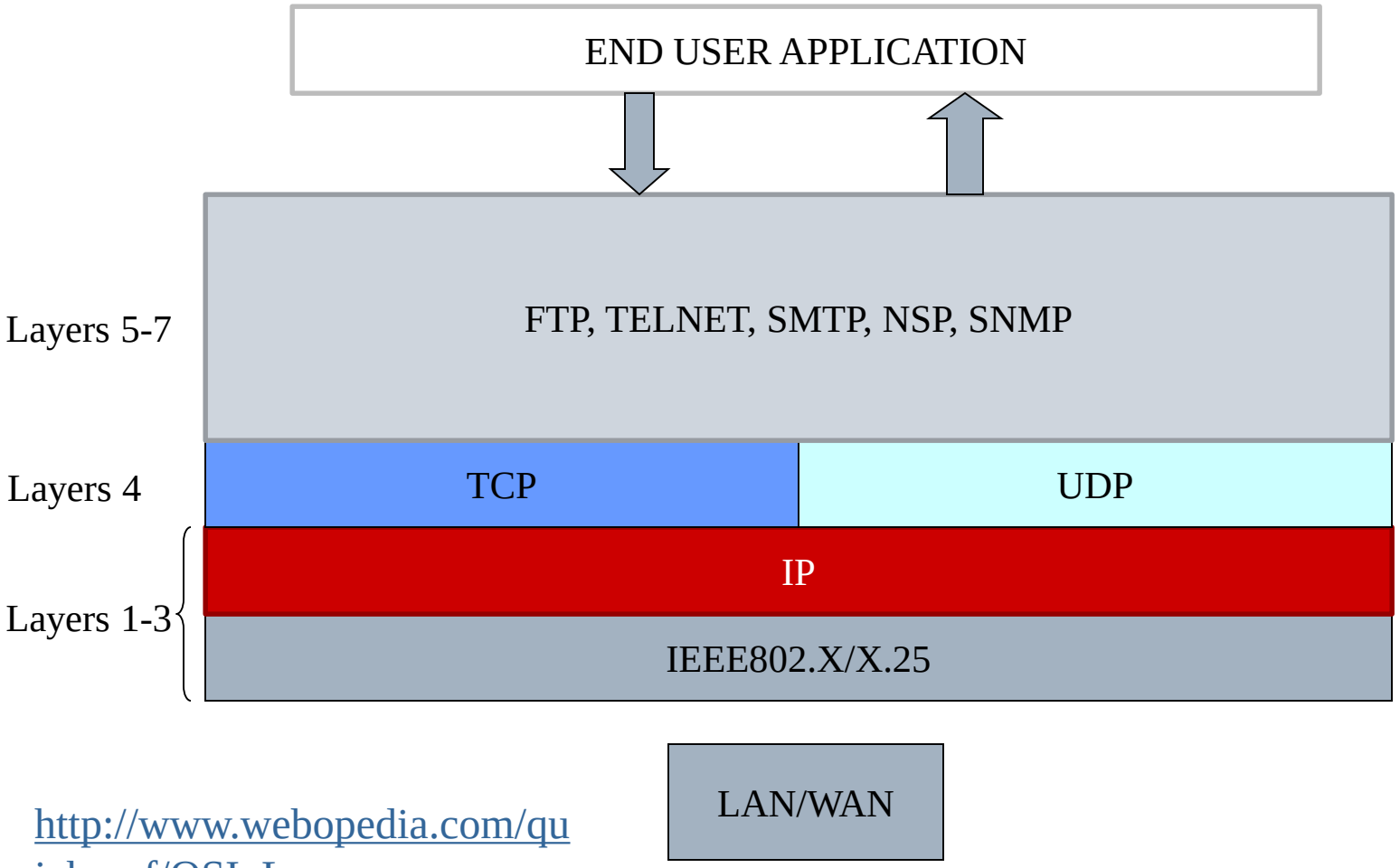
Communication

- Socket (TCP/IP)
- Remote Procedure Call (RPC)

TCP/IP (Socket)

- Transport Protocols
  - User Datagram Protocol (UDP)
    - UDP/IP is an unreliable, connectionless transport protocol, which uses IP to transport IP datagrams but adds error correction and a protocol **port address** to specify the process on the remote system for which the packet is destined.
  - Transmission Control Protocol (TCP)
    - TCP/IP is a reliable stream protocol for communicating information between two processes

TCP/IP Protocol Layers



http://www.webopedia.com/quick_ref/OSI_Layers.asp

This Lecture

Distributed Systems

Overview

Remote procedure call (RPC)

Network file system (NFS)

Virtualization and Cloud

Messages

- Initially with network programming, programmers hand-coded messages to send requests and responses
- Hand-coding messages are time-consuming
  - Need to worry about message formats
  - Have to pack and unpack data from messages
  - Servers have to decode and dispatch messages to handlers
  - Messages are often asynchronous
- Messages are not a very natural programming model
  - Could encapsulate messaging into a library
  - Just invoke library routines to send a message
  - Which leads us to RPC…

Procedure Calls

- Procedure calls are a more natural way to communicate
  - Every language supports them
  - Semantics are well-defined and understood
  - Natural for programmers to use
- Idea: have servers export a set of procedures that can be called by client programs
  - Similar to module interfaces, class definitions, etc.
- Clients just do a procedure call as if they were directly linked with the server
  - Under the covers, the procedure call is converted into a message exchange with the server

Remote Procedure Calls

- We would like to use procedure call as a model for distributed (remote) communication
- Lots of issues
  - How do we make this invisible to the programmer?
  - What are the semantics of parameter passing?
  - How do we bind (locate, connect to) servers?
  - How do we support heterogeneity (OS, arch, language)?
  - How do we make it perform well?

RPC Model

- A server defines the server's interface using an interface definition language (IDL)
  - The IDL specifies the names, parameters, and types for all client-callable server procedures
- A stub compiler reads the IDL and produces two stub procedures for each server procedure (client and server)
  - The server programmer implements the server procedures and links them with the server-side stubs
  - The client programmer implements the client program and links it with the client-side stubs
  - The stubs are responsible for managing all details of the remote communication between client and server

RPC Stubs

- The stubs send messages to each other to make RPC work "transparently"
  - A client-side stub is a procedure that looks to the client as if it were a callable server procedure
  - A server-side stub looks to the server as if a client called it

- Remote Procedure Call (RPC) is used both by OS and applications
  - Network File System is implemented as a set of RPCs
  - DCOM, CORBA, Java RMI, etc., are all basically just RPC

RPC Example

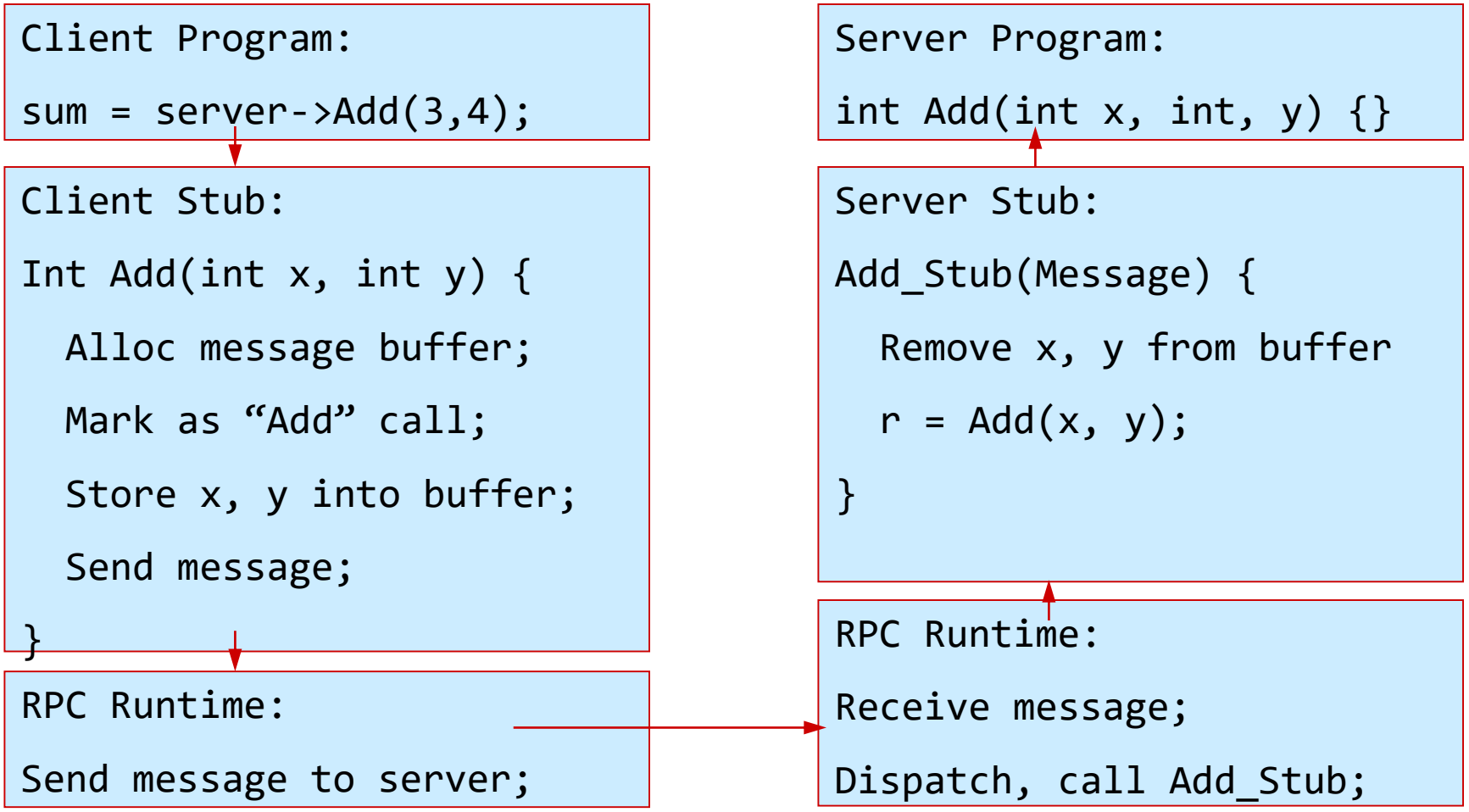
- If the server were just a library, then Add would just be a procedure call

```
Server Interface:
int Add(int x, int y);
```

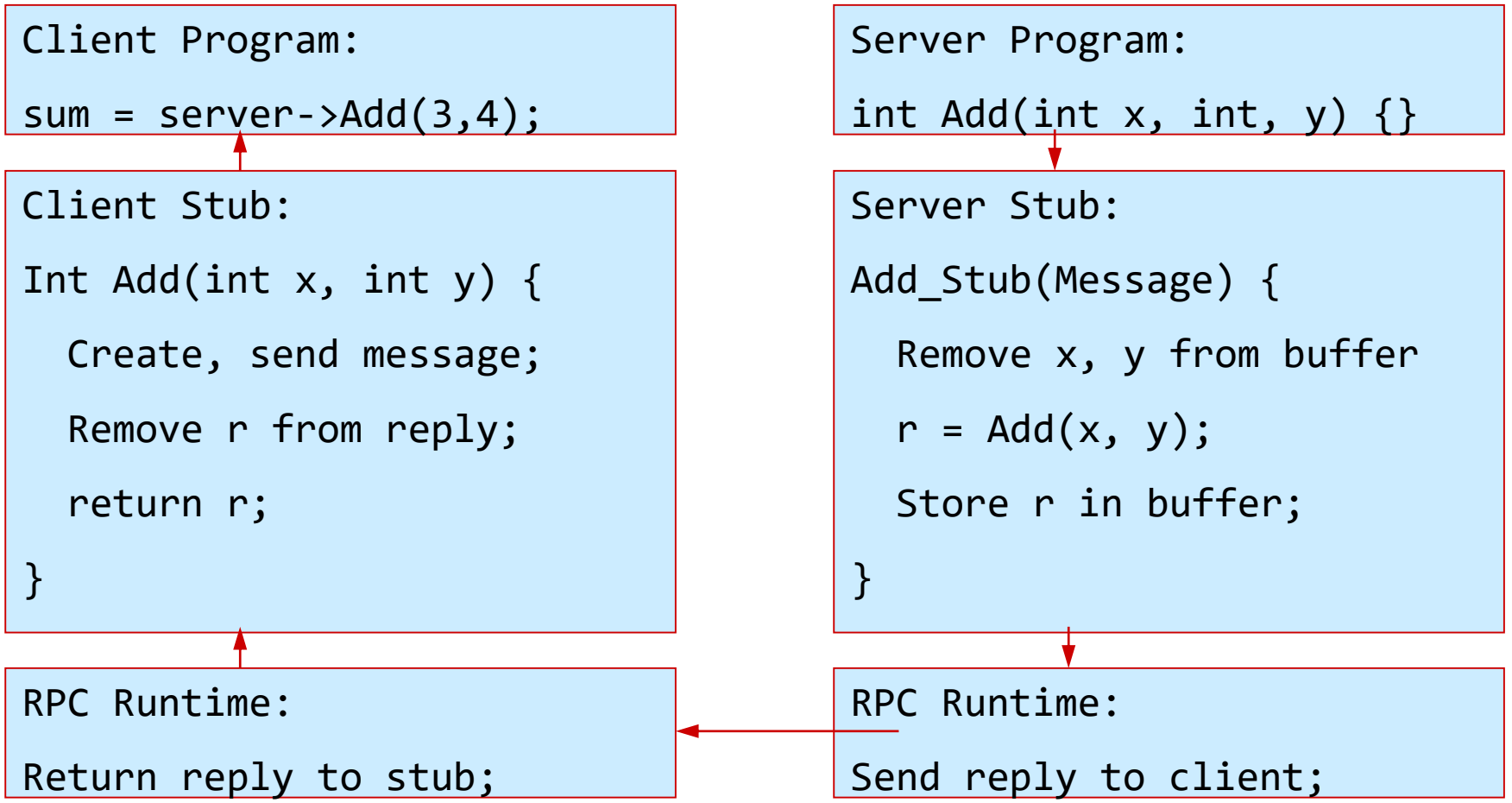
```
Client Program:
…
sum = server->Add(3,4);
…
```

```
Server Program:
int Add(int x, int, y) {
  return x + y;
}
```

RPC Example: Call



RPC Example: Return



RPC Marshalling

- Marshalling is the packing of procedure parameters into a message packet
- The RPC stubs call type-specific procedures to marshal (or unmarshal) the parameters to a call
  - The client stub marshals the parameters into a message
  - The server stub unmarshals parameters from the message and uses them to call the server procedure
- On return
  - The server stub marshals the return parameters
  - The client stub unmarshals return parameters and returns them to the client program
- Comparing to serialization?

RPC Binding

- Binding is the process of connecting the client to the server

- The server, when it starts, exports its interface
  - Identifies itself to a network name server
  - Tells RPC runtime it's alive and ready to accept calls
- The client, before issuing any calls, imports the server
  - RPC runtime uses the name of the server to find the location of a server and establish a connection

- The import and export operations are explicit in the server and client programs
  - Breaking transparency

RPC Transparency

- One goal of RPC is to be as transparent as possible
  - Make remote procedure calls look like local procedure calls
- We have seen that binding breaks transparency
- What else?
  - Failures – remote nodes/networks can fail in more ways than with local procedure calls
    - Need extra support to handle failures well
  - Performance – remote communication is inherently slower than local communication
    - If program is performance-sensitive, could be a problem

RPC Summary

- RPC is the most common model for communication in distributed applications
  - "Cloaked" as DCOM, CORBA, Java RMI, etc.
  - Also used on same node between applications
- RPC is essentially the language support for distributed programming
- RPC relies upon a stub compiler to automatically generate client/server stubs from the IDL server descriptions
  - These stubs do the marshalling/unmarshalling, message sending/receiving/replying

This Lecture

Distributed Systems

Overview

Remote procedure call (RPC)

Network file system (NFS)

Virtualization and Cloud

Network File System

We'll now look at a file system that uses RPC

- Network File System (NFS)
  - Protocol for remote access to a file system
    - Does not implement a file system per se
    - Remote access is transparent to applications
  - File system, OS, and architecture independent
    - Originally developed by Sun
    - Although Unix-y in flavor, explicit goal to work beyond Unix
  - Client/server architecture
    - Local file system requests are forwarded to a remote server
    - These requests are implemented as RPCs

Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network Filesystem . Proceedings of the Summer 1985 USENIX Conference, Portland OR, June 1985, pp. 119-130.

Mounting

- Before a client can access files on a server, the client must mount the file system on the server
  - The file system is mounted on an empty local directory
  - Same way that local file systems are attached
  - Depending on OS (e.g., Unix dirs vs NT drive letters)
- Servers maintain ACLs of clients that can mount their directories
  - When mount succeeds, server returns a file handle
  - Clients use this file handle as a capability to do file operations
- Mounts can be cascaded
  - Can mount a remote file system on a remote file system

NFS Protocol

- The NFS protocol defines a set of operations that a server must support
  - Reading and writing files
  - Accessing file attributes
  - Searching for a file within a directory
  - Reading a set of directory links
  - Manipulating links and directories
- These operations are implemented as RPCs
  - Usually by daemon processes (e.g., nfsd)
  - A local operation is transformed into an RPC to a server
  - Server performs operation on its own file system and returns

Statelessness

- Note that NFS has no open or close operations
- NFS is stateless
  - An NFS server does not keep track of which clients have mounted its file systems or are accessing its files
  - Each RPC has to specify all information in a request
    - Operation, FS handle, file id, offset in file, sequence #
- Robust
  - No reconciliation needs to be done on a server crash/reboot
  - Clients detect server reboot, continue to issue requests
- Writes must be synchronous to disk, though
  - Clients assume that a write is persistent on return
  - Servers cannot cache writes

Consistency

- Client caching can lead to consistency problems
  - Caching a write on client A will not be seen by other clients
  - Cached writes by clients A and B are unordered at server
  - Since sharing is rare, though, NFS clients usually do cache
- NFS statelessness is both its key to success and its Achilles' heel
  - NFS is straightforward to implement and reason about
  - But limitations on caching can severely limit performance
    - Dozens of network file system designs and implementations that perform much better than NFS
  - But note that it is still the most widely used remote file system protocol and implementation

This Lecture

Distributed Systems

Overview

Remote procedure call (RPC)

Network file system (NFS)

Virtualization and Cloud

Virtualization: Definition

- Virtualization is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources*
  - It is the process by which one computer hosts the appearance of many computers.
- Virtualization is used to improve IT throughput and costs by using physical resources as a pool from which virtual resources can be allocated.

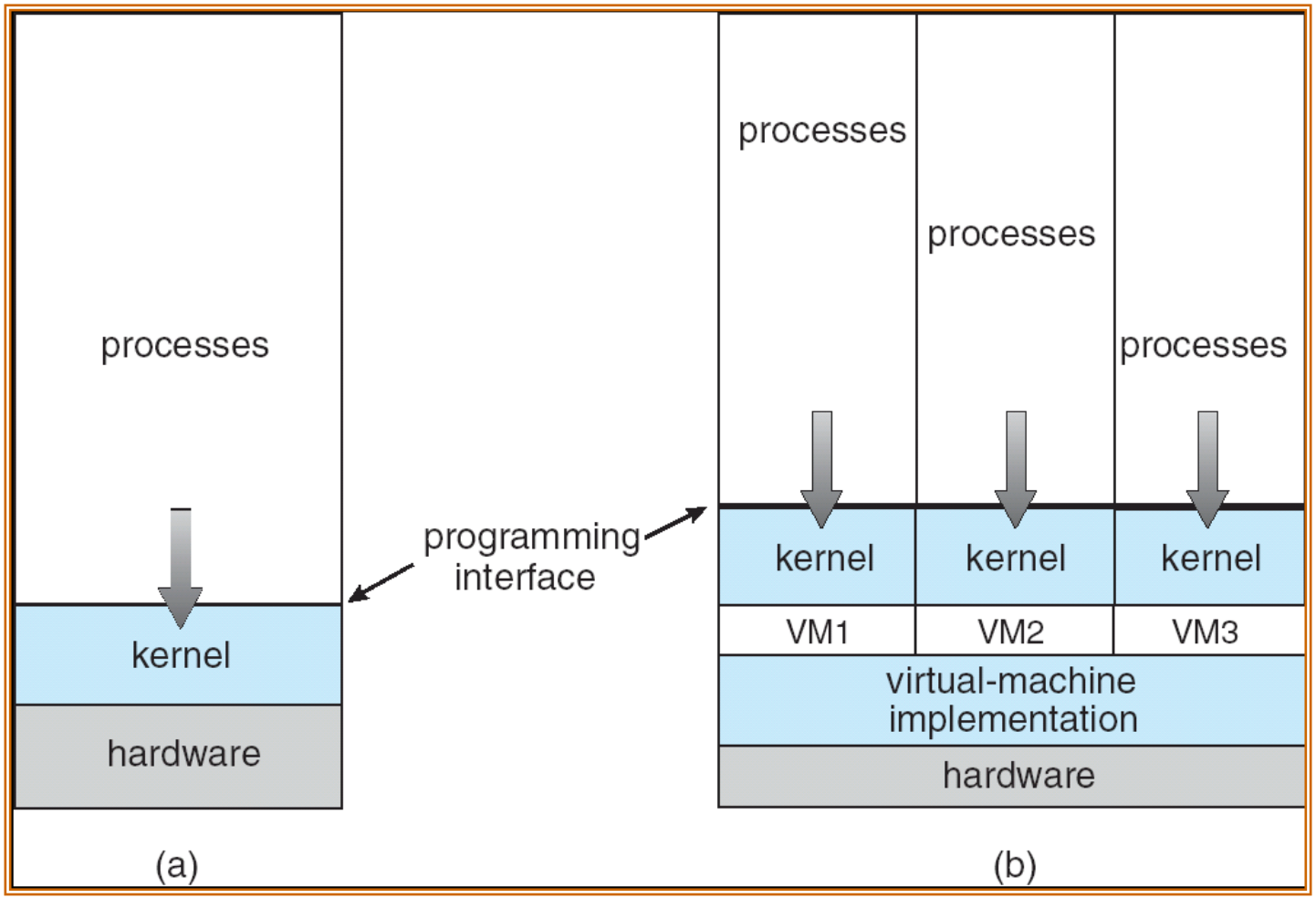
**VMWare white paper, Virtualization Overview*

Virtual Machine

- A virtual machine (VM) provides interface identical to underlying bare hardware
  - I.e., all devices, interrupts, memory, page tables, etc.

- A Virtual Machine Operating System creates illusion of multiple processors
  - Each capable of executing independently
  - No sharing, except via network protocols
  - Clusters and SMP can be simulated

Virtual Machines



Hypervisor

- A hypervisor is a program that allows multiple operating systems to share a single hardware host.
  - Also called a virtual machine manager/monitor (VMM), or virtualization manager
  - Each guest operating system appears to have the host's processor, memory, and other resources all to itself.
  - However, the hypervisor is actually controlling the host processor and resources, allocating what is needed to each operating system in turn and making sure that the guest operating systems (called virtual machines) cannot disrupt each other.

History – CP67 / CMS

- IBM Cambridge Scientific Center
- Ran on IBM 360/67
  - Alternative to TSS/360, which never sold very well
- Replicated hardware in each “process”
  - Virtual 360/67 processor
  - Virtual disk(s), virtual console, printer, card reader, etc.
- CMS: Cambridge Monitor System
  - A single user, interactive operating system
- Commercialized as VM370 in mid-1970s

History (continued)

- "Hypervisor" systems – mid 1970s⇒mid 1990s
  - Large mainframes (IBM, HP, etc.)
  - Internet hosting services
  - Virtual dedicated services
  - …

Modern Virtualization Systems

- VMware
  - Workstation and Player
  - Multiple versions of VMware Server
  - Virtual appliances
- Xen
  - Public domain hypervisor
  - Adaptive support in operating systems
  - Emerging support in processor chips
    - Intel, AMD

Virtual Machines (continued)

- Virtual-machine concept provides complete protection of system resources
  - Each virtual machine is isolated from all other virtual machines.
  - However, limited sharing of resources
- Virtual-machine system is a good vehicle for operating-systems research and development.
  - System development is done on the virtual machine does not disrupt normal operation
  - Multiple concurrent developers can work at same time
- The virtual machine concept is difficult to implement due to the effort required to provide an exact duplicate to the simulated machine

Example – Page tables

- Suppose *guest OS* has its own page tables, the *virtualization layer* must
  - Copy those tables to its own
  - Trap every reference or update to tables and simulate it
- During page fault
  - *Virtualization layer* must decide whether fault belongs to *guest OS* or self
  - If *guest OS*, must simulate a page fault
- Likewise, *virtualization layer* must trap and simulate *every* privileged instruction in machine!

Virtual Machines (continued)

- Some hardware architectures or features are impossible to *virtualize*
  - Certain registers or state not exposed
  - Unusual devices and device control
  - Clocks, time, and real-time behavior

- Solution – drivers or tools in guest OS
  - *VMware Tools*
  - *Xen* configuration options in Linux build

Snapshots & Migration

- *Snapshot:* freeze a copy of virtual machine
  - Identify all pages in disk files, VM memory
  - Use copy-on-write for any subsequent modifications
  - To revert, throw away the copy-on-write pages

- *Migration:* move a VM to another host
  - Take snapshot (fast)
  - Copy all pages of snapshot (not so fast)
  - Copy modified pages (fast)
  - Freeze virtual machine and copy VM memory
    - Very fast, fractions of a second

Cloning

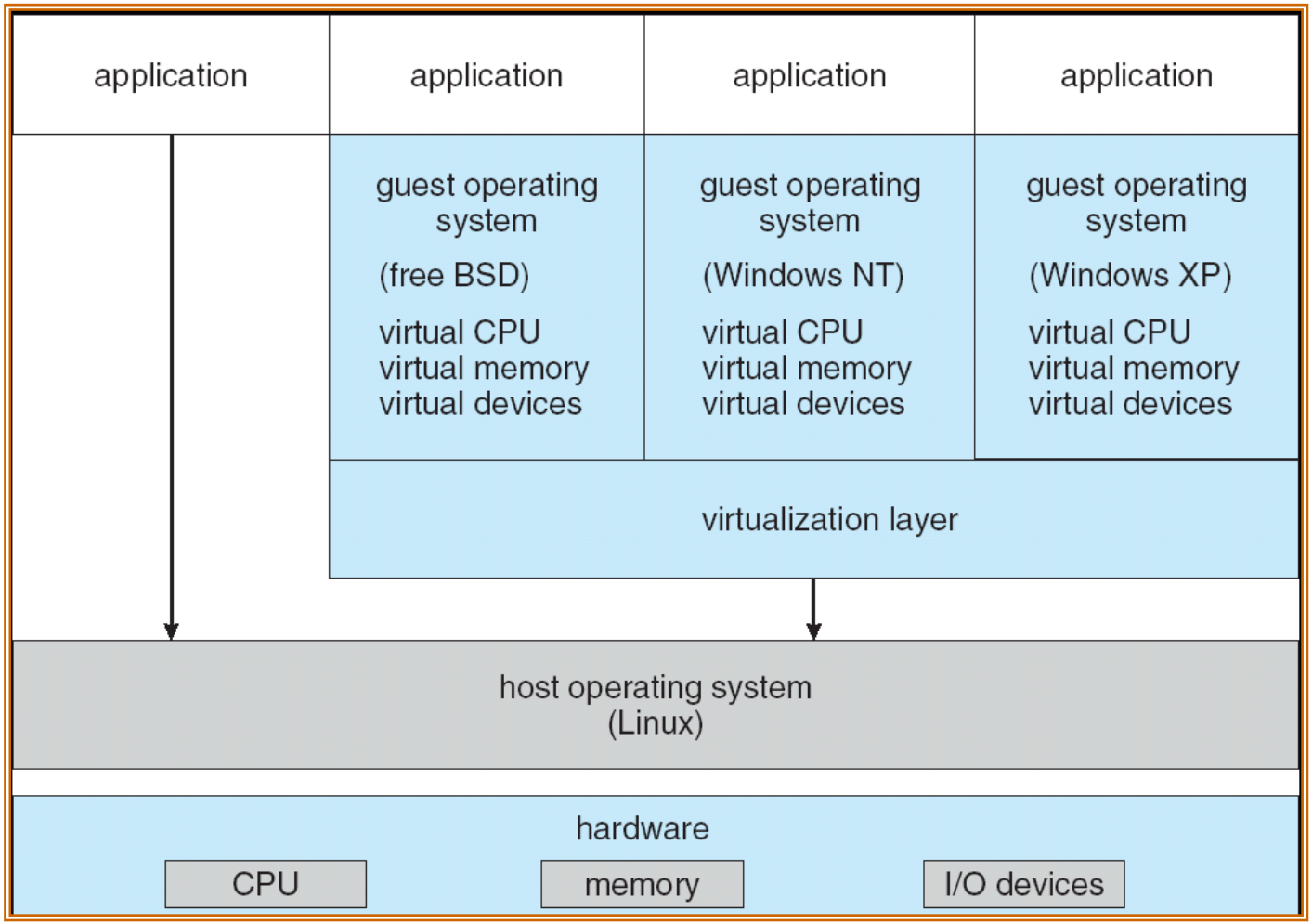
- *Simple clone:*
  - Freeze virtual machine
  - Copy all files implementing it
  - Use copy-on-write to speed up

- *Linked clone:*
  - Take snapshot
  - Original and each clone is a copy-on-write version of snapshot

VMware – Modern Virtual Machine System

- Founded 1998, Mendel Rosenblum *et al.*
  - Research at Stanford University
- *E.g., VMware Workstation*
  - Separates *Host OS* from *virtualization layer*
  - Host OS may be Windows, Linux, etc.
  - Wide variety of Guest operating systems
  - < $200
  - *VMware Player* is a free, stripped-down version of *VMware Workstation*

VMware Architecture



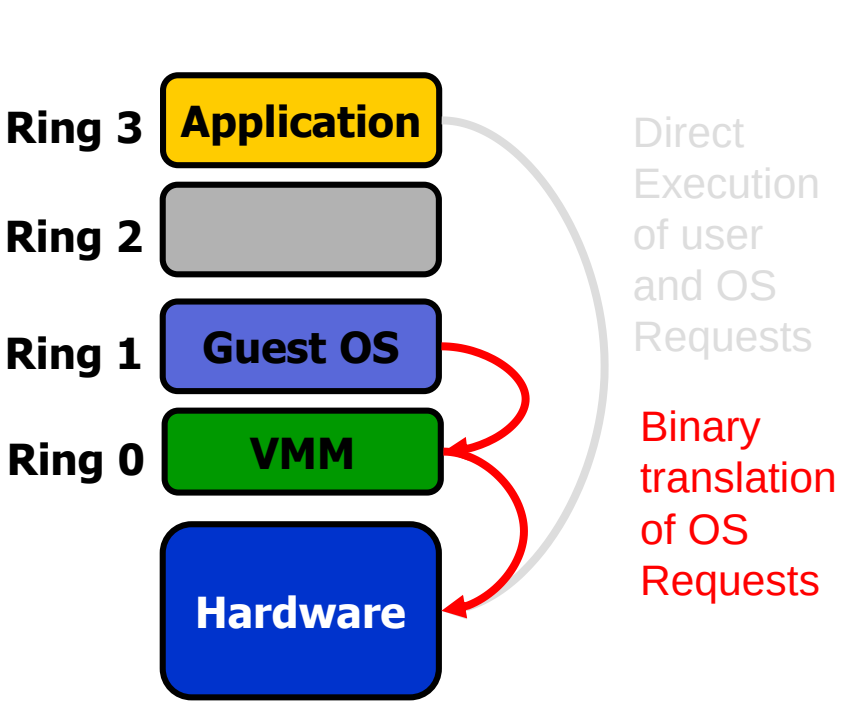
Xen and the Art of Virtualization

- Open source virtualization technology
  - From University of Cambridge
  - http://www.cl.cam.ac.uk/research/srg/netos/xen/

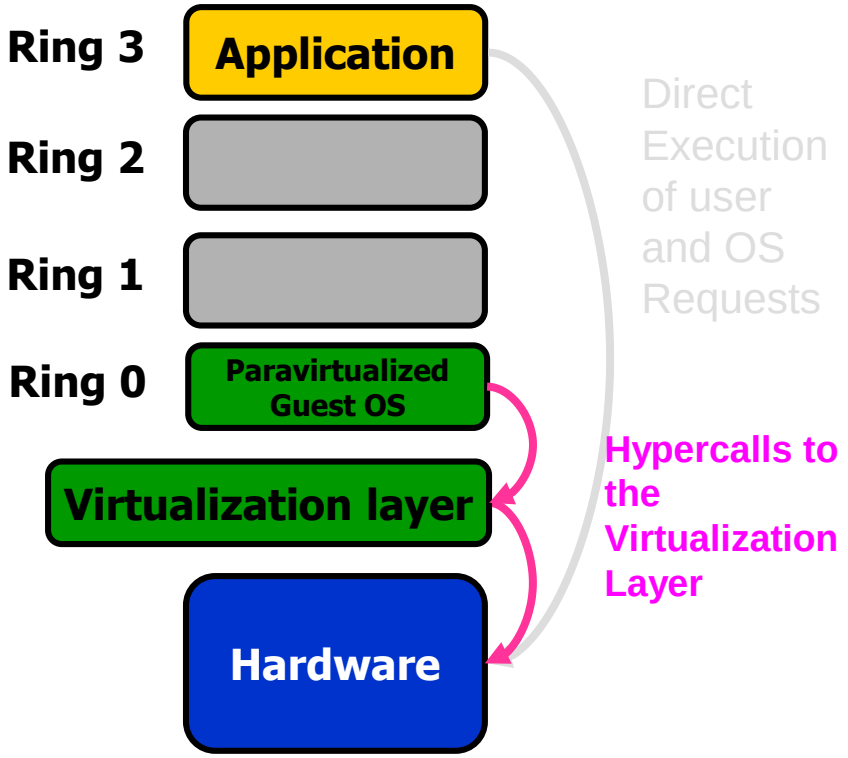
- Philosophy – Adapt Guest OS to virtualization layer

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03).

Full Virtualization vs. Paravirtualization



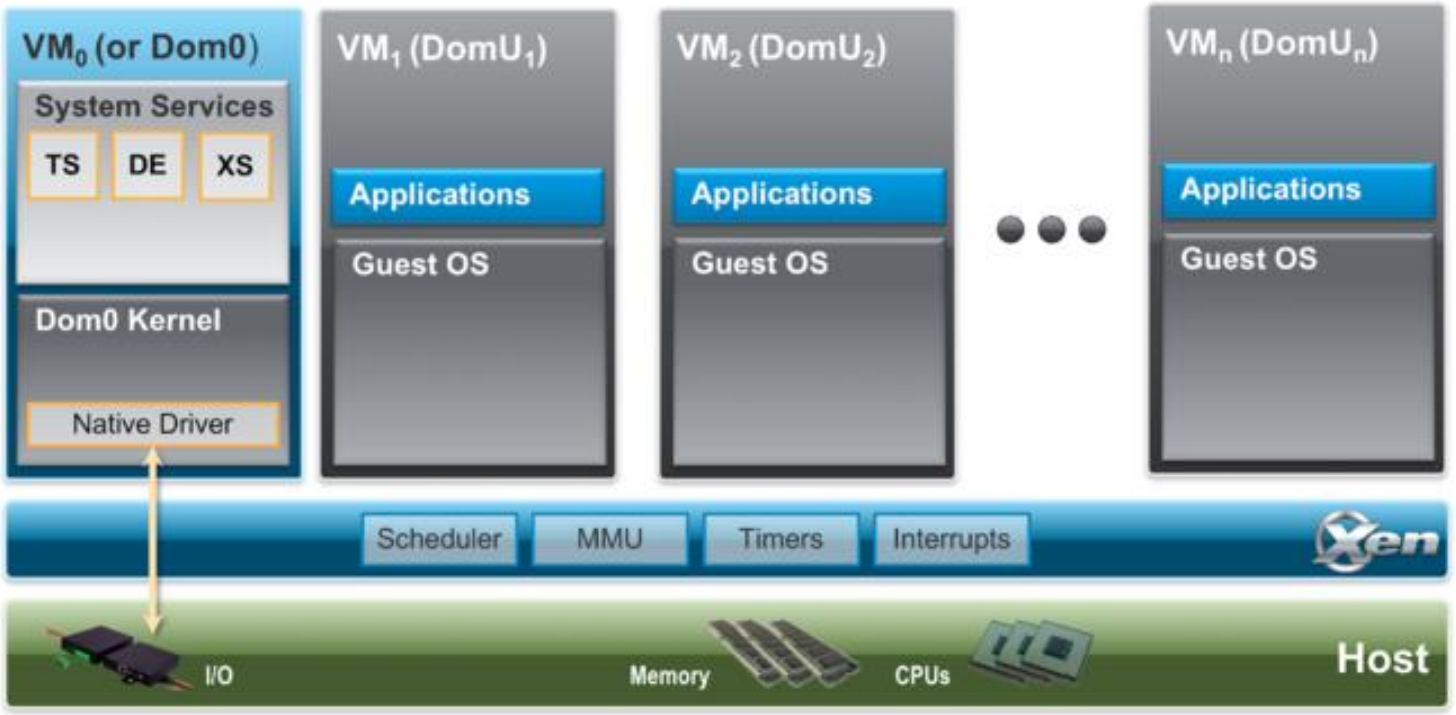
Full virtualization



Paravirtualization

The Xen Architecture

- The Control Domain (or Domain 0) is a specialized VM
  - Special privileges like the capability to access the hardware directly
  - Handles all access to the system's I/O functions and interacts with the other VMs.



Different Virtualization Technology

Virtualizaton method	Type 1 hypervisor	Type 2 hypervisor
Virtualization without HW support	ESX Server 1.0	VMware Workstation 1
Paravirtualization	Xen 1.0	
Virtualization with HW support	vSphere, Xen, Hyper-V	VMware Fusion, KVM, Parallels
Process virtualization		Wine

Type 1 hypervisors: run on the bare metal
Type 2 hypervisors: run on an existing host operating system

Benefits of Virtualization

- Sharing of resources helps cost reduction
- Isolation: Virtual machines are isolated from each other as if they are physically separated
- Encapsulation: Virtual machines encapsulate a complete computing environment
- Hardware Independence: Virtual machines run independently of underlying hardware
- Portability: Virtual machines can be migrated between different hosts.

Virtualization in Cloud Computing

Cloud computing takes virtualization one step further:

- You don't need to own the hardware
- Resources are rented as needed from a cloud
- Various providers allow creating virtual servers:
  - Choose the OS and software each instance will have
  - The chosen OS will run on a large server farm
  - Can instantiate more virtual servers or shut down existing ones within minutes
- You get billed only for what you used

What is Cloud Computing?

- Cloud Computing is a general term used to describe a new class of network based computing that takes place over the Internet
  - basically a step on from Utility Computing
  - a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
  - Using the Internet for communication and transport provides hardware, software and networking services to clients
- These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).

Cloud Service Models

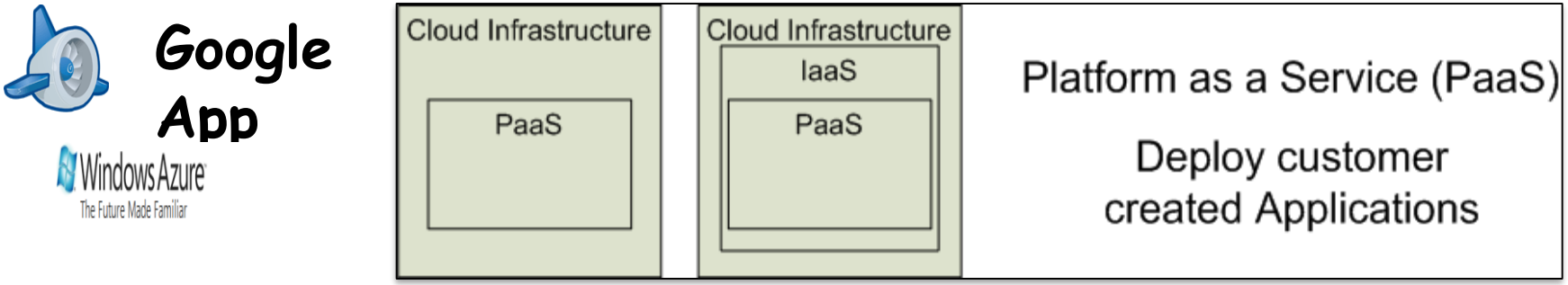
Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

SalesForce CRM

LotusLive

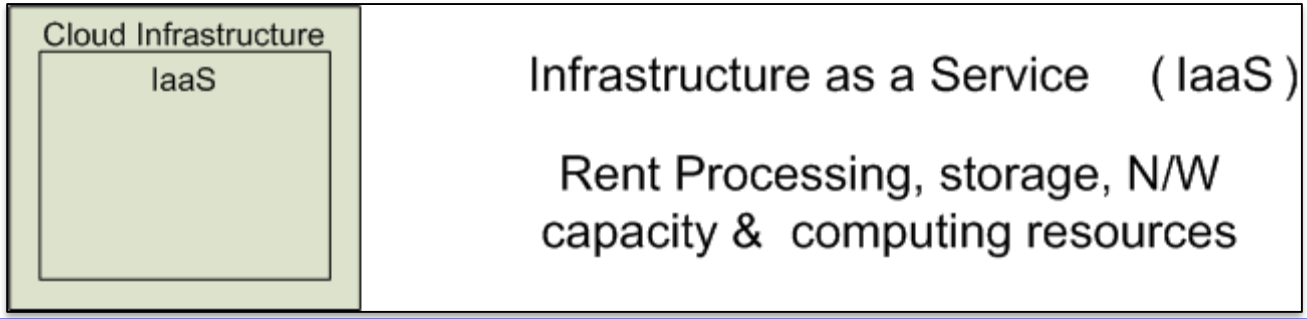


Google App

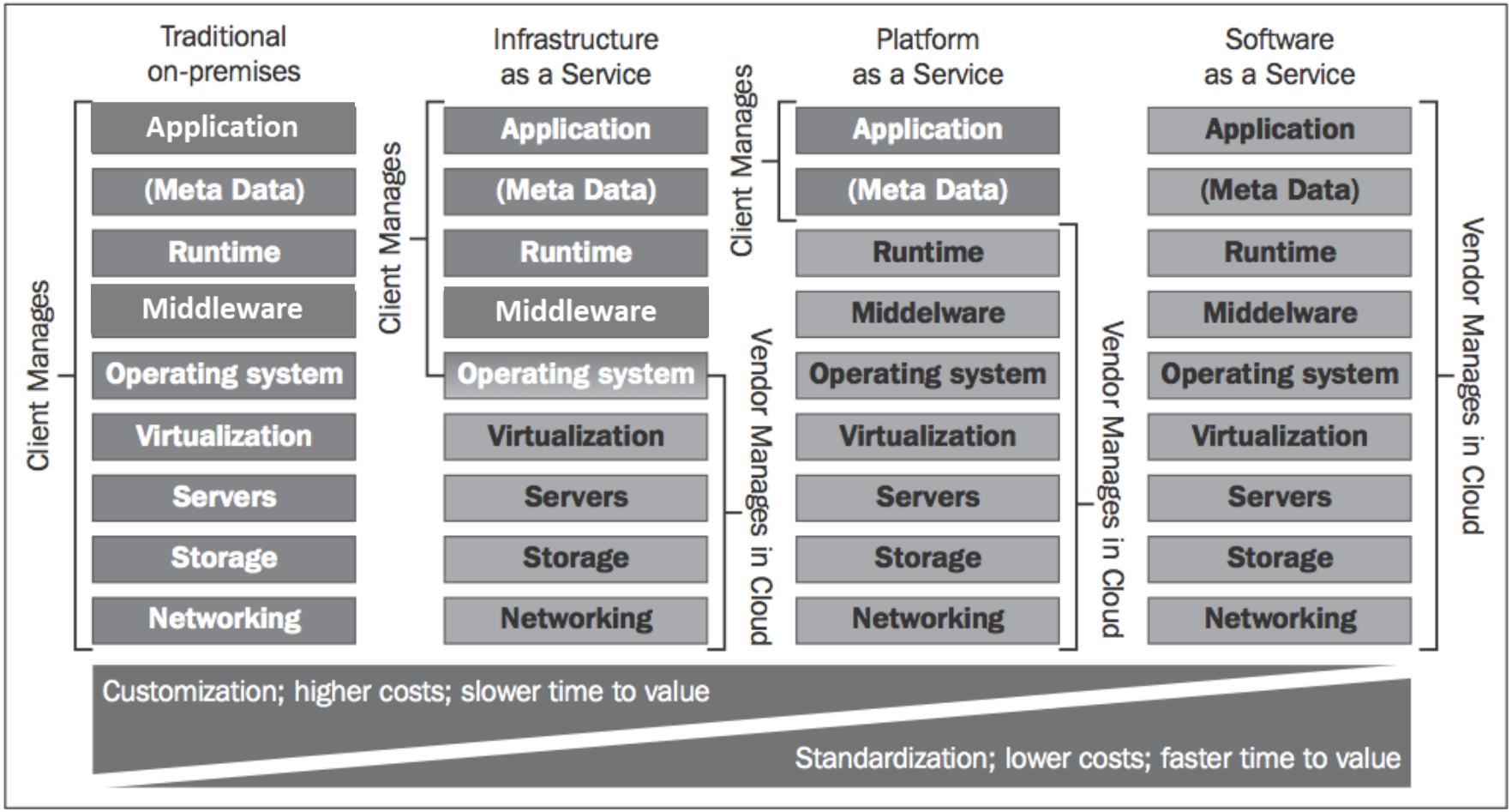
Windows Azure
The Future Made Familiar

amazon web services

rackspace HOSTING



Comparison: Different Cloud Services



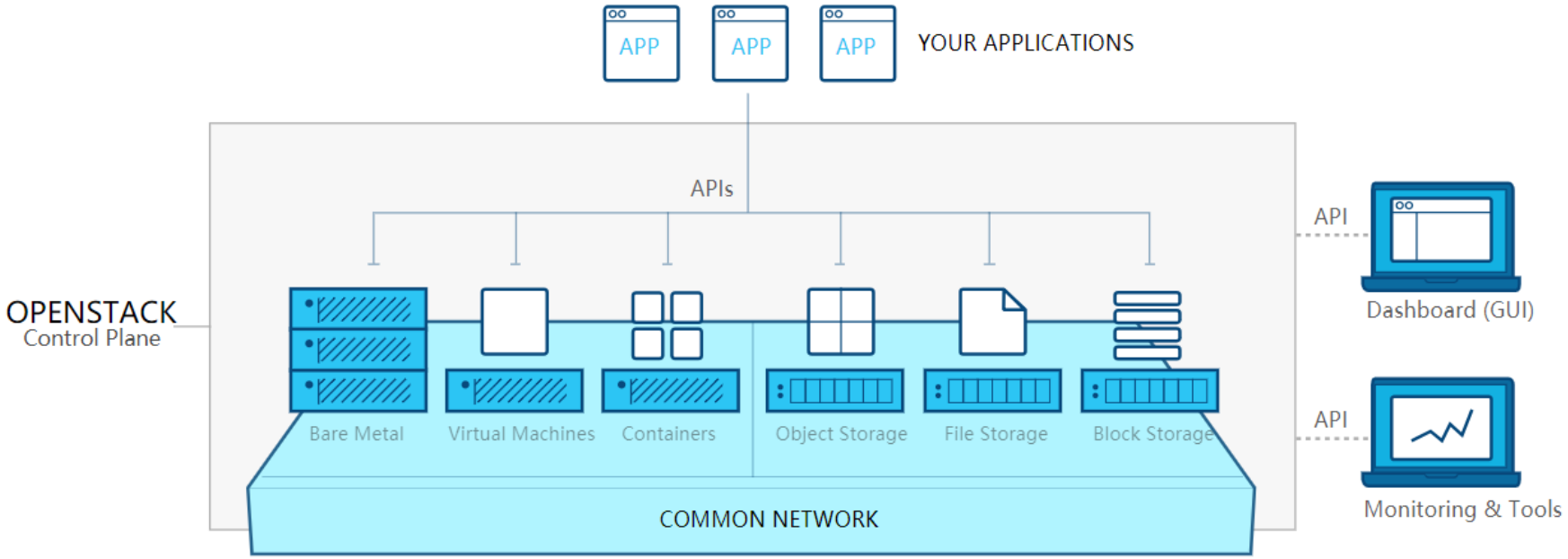
Cloud Operating Systems

- A cloud operating system is a type of operating system designed to operate within cloud computing and virtualization environments.
  - manages the operation, execution and processes of virtual machines, virtual servers and virtual infrastructure, as well as the back-end hardware and software resources.

- Examples:
  - Amazon AWS, Google Cloud, Microsoft Azure
  - OpenStack

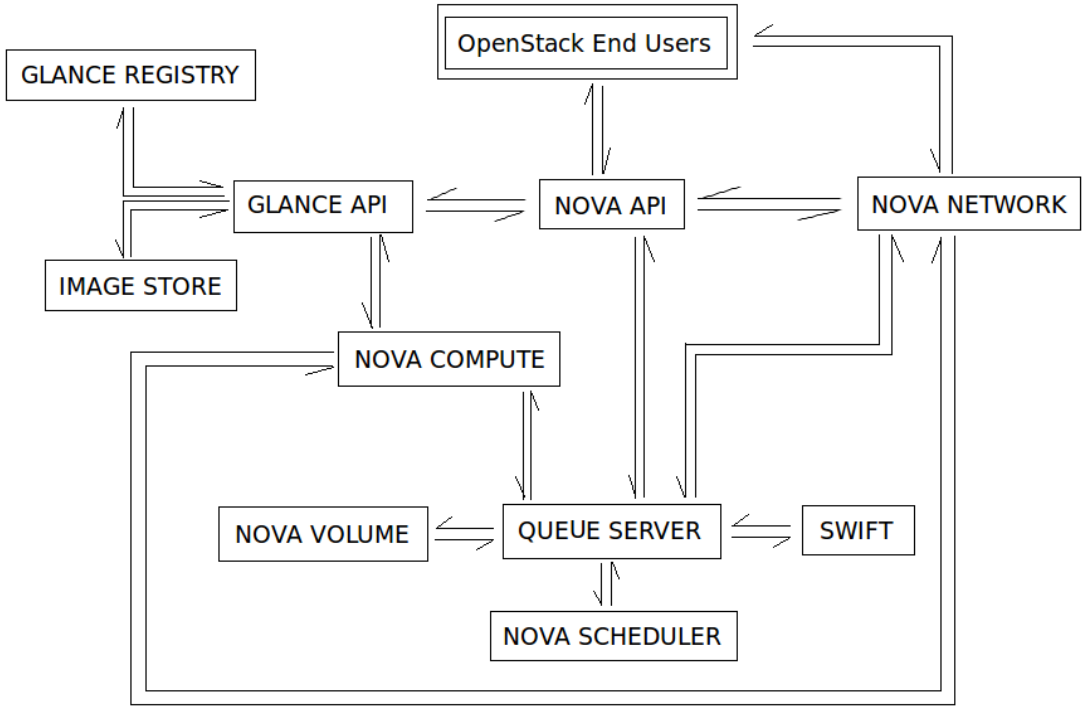
Example: OpenStack

- OpenStack is a cloud operating system
  - Controls large pools of compute, storage, and networking resources throughout a datacenter
  - All managed through a dashboard that gives administrators control
  - Empowering their users to provision resources through a web interface



The OpenStack Architecture

- 3 main modules
  - Nova
    - Compute Infrastructure
  - Swift
    - Storage Infrastructure
  - Glance
    - Imaging Service



Summary

- Overview of distributed systems
- Remote procedure call (RPC)
- Network file system (NFS)
- Virtualization and cloud

- Next lectures
  - Lab quiz
  - OS Design Issues