Cdct (4361602) - Problem Solver

Antigravity AI

June 4, 2026

Cdct

First Edition: 2026

Author: Antigravity AI
Website: milav.in
YouTube: @milav.dabgar

To my students,
who constantly inspire me to find new analogies,
break down complex theories, and make learning
an engineered science.

Preface

About this Book

Welcome to **Computer Networks**! This textbook is meticulously designed to serve as a comprehensive problem solver and study companion. It provides a robust and intuitive foundation in the core principles of this subject, bridging the gap between theoretical models and practical applications. A unique feature of this book is its focus on decoding complex concepts using clear, step-by-step methodologies and real-world analogies.

Target Audience

This book is intended for students and professionals looking to master the concepts of Computer Networks. Whether you are revising for exams, seeking to solidify your fundamental understanding, or looking for practical examples to clarify abstract concepts, this book has been structured to support your learning journey. It serves as an accessible guide designed to remove the fear of complex topics, replacing it with an appreciation for the underlying mechanics.

Scope and Coverage

The coverage is divided logically to follow standard academic curriculums. Each chapter focuses on key topics, breaking them down into digestible sections:

- **Theoretical Insights** – Core concepts explained intuitively.
- **Method Blocks** – Standardized, step-by-step procedures for solving specific types of problems.
- **Worked Examples** – Fully solved problems that apply the methods in practice.

We hope this resource proves invaluable in your studies and helps you achieve excellence in **Computer Networks**. Happy learning!

Acknowledgments

Writing a comprehensive book that connects the fundamental forces of Chemistry with practical engineering applications requires more than just academic knowledge—it requires a community of support. I would like to express my sincere gratitude to everyone who contributed to the realization of this project.

Specially, I extend my heartfelt gratitude to the **Department of Technical Education (DTE), Government of Gujarat**, and **Government Polytechnic, Palanpur**, for providing an environment that fosters innovation, academic rigor, and continuous professional development.

I am deeply thankful to my family for their unwavering patience and support during the countless hours spent researching, formatting, and refining these chapters.

Finally, my deepest appreciation goes to my students. Your curiosity, challenging questions, and continuous feedback are the true driving force behind the unique analogies and streamlined structure of this book. This textbook was engineered for you.

Antigravity AI

About the Author

Milav Jayeshkumar Dabgar is an engineering educator and R&D professional with over 9 years of experience in electronics hardware development, embedded systems, AI/ML, and full-stack software engineering. He currently serves as a **Lecturer (Class - II) & Innovation Leader** at Government Polytechnic, Palanpur, under the Education Department of the Government of Gujarat.

Milav holds a Master of Engineering (ME) in Communication Systems from L.D. College of Engineering and a Bachelor of Engineering (BE) in Electronics & Communication. He is also currently pursuing a **BS in Data Science and Applications from IIT Madras**, where he has completed both the **Diploma in Programming** and the **Diploma in Data Science** with distinction.

Most recently, he completed the **AICTE QIP PG Certificate Program in Deep Learning from SVNIT Surat** with a **perfect 10/10 CGPA**, topping his batch.

A dedicated mentor, Milav has guided numerous student teams in securing over **INR 45 lakhs** in innovation funding, including INR 25 lakhs from Shark Tank India and INR 20 lakhs through iHub Gujarat, resulting in two student patents. He is recognized as an **NPTEL Evangelist** and **NPTEL Discipline Star** in Computer Science, reflecting his commitment to continuous learning and academic excellence.

His technical expertise spans from embedded systems (8051, STM32, Arduino) to modern web technologies (Next.js, Python, PostgreSQL) and Data Science (TensorFlow, PyTorch). Through this textbook, he aims to simplify complex Engineering Chemistry concepts by integrating relatable, real-world analogies drawn from modern tech and engineering landscapes.

Milav is also an active content creator, running a popular **YouTube channel** (@milav.dabgar) where he shares technical tutorials and academic resources. He maintains a personal blog and resource hub at milav.in and can be reached for professional collaborations on LinkedIn.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction to Cloud Computing

Cloud computing provides on-demand availability of computer system resources without direct active management by the user. While often considered a theoretical concept, evaluating and selecting the right cloud architecture is a systematic and computational decision-making process.

1.1 Define Cloud Computing

To successfully implement cloud computing, one must logically evaluate the workload characteristics. We define cloud computing computationally by analyzing resource utilization metrics and cost efficiency.

Methodology:

Evaluating Cloud Suitability Algorithm To determine if a traditional IT workload should be migrated to the cloud, evaluate the system using the following mathematical and logical steps:

1. **Identify Workload Variability (V):** Calculate the ratio of peak resource demand to average resource demand. If $V > 1.5$, the workload is highly variable.
2. **Determine Resource Efficiency (E):** Calculate the cost difference:

$$E = C_{\text{local}} - C_{\text{cloud}}$$

   where C_{local} is the cost of upgrading local dedicated hardware, and C_{cloud} is the estimated cost of cloud resources over the same period.
3. **Evaluate Elasticity Requirement (R):** If the system requires automatic scaling to prevent failure, set $R = 1$ (True). Otherwise, $R = 0$ (False).
4. **Decision Rule:** If $V > 1.5$, $E > 0$, and $R = 1$, then migrating to the cloud is mathematically optimal.

Worked Example:

Assessing Workload Migration A retail company hosts an e-commerce website on local servers. The average CPU utilization is 200 GB RAM equivalent, but during holiday sales, the demand spikes to 800 GB RAM equivalent. The cost of upgrading local servers to handle the peak is \$50000. Alternatively, dynamic cloud hosting costs \$12000 annually. The system must not crash during peak sales. Evaluate if cloud computing is suitable.

Solution:

1. **Workload Variability (V):**

$$V = \frac{\text{Peak Demand}}{\text{Average Demand}} = \frac{800}{200} = 4.0$$

   Since $4.0 > 1.5$, the workload is highly variable.
2. **Resource Efficiency (E):** Given $C_{\text{local}} = \$50000$ and $C_{\text{cloud}} = \$12000$:

$$E = 50000 - 12000 = 38000$$

Since $E > 0$, cloud hosting provides positive cost savings.

3. **Elasticity Requirement (R):** The system must automatically handle traffic spikes to prevent crashes. Thus, $R = 1$.

4. **Conclusion:** Because $V = 4.0$, $E = 38000$, and $R = 1$, all conditions of the decision rule are met. The system should be migrated to cloud computing.

1.2 Cloud Service Models

Cloud computing offers three main service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Choosing the correct model is evaluated using a discrete decision algorithm based on administrative requirements.

Methodology:

Service Model Selection Algorithm Apply the following hierarchical logic to determine the correct service model:

1. **Compute Infrastructure Control (I_{ctrl}):** Does the user require administrative access to the Operating System and Network?
   - If I_{ctrl} = True, select **IaaS** and terminate.
   - If I_{ctrl} = False, proceed to Step 2.
2. **Compute Development Control (D_{ctrl}):** Is the user deploying custom code and databases without managing the OS?
   - If D_{ctrl} = True, select **PaaS** and terminate.
   - If D_{ctrl} = False, proceed to Step 3.
3. **Compute Software Usage (S_{use}):** Does the user only need to consume a pre-built application?
   - If S_{use} = True, select **SaaS**.

Worked Example:

Determining Service Models for IT Projects Determine the optimal cloud service model for a software development team that needs a runtime environment with pre-installed Node.js and MongoDB to deploy a new web application rapidly. They do not want to manage OS updates or firewalls.
Solution:

1. **Evaluate Infrastructure Control (I_{ctrl}):** The team explicitly states they do not want to manage OS updates or firewalls. Therefore, I_{ctrl} = False. We bypass IaaS and move to Step 2.
2. **Evaluate Development Control (D_{ctrl}):** The team is deploying custom code and needs a specific runtime environment (Node.js and MongoDB). Therefore, D_{ctrl} = True.
3. **Conclusion:** Based on the algorithm, since D_{ctrl} = True, the optimal model is **PaaS** (Platform as a Service).

1.3 Types of Clouds

Cloud deployments are categorized into Public, Private, and Hybrid clouds. Deployment selection is formulated as a constraint satisfaction problem balancing data sensitivity, capital cost, and workload predictability.

Methodology:

Cloud Deployment Optimization Method Use the following parameters to map a workload to its optimal deployment model:

1. **Data Security Level (S):** Set $S = 1$ if the data contains strictly confidential or legally regulated information. Set $S = 0$ for public or low-risk data.
2. **Workload Predictability (P):** Set $P = 1$ if resource utilization is constant. Set $P = 0$ if resource usage bursts unpredictably.
3. **Evaluate the Decision Matrix:**
   - If $S = 0$ and $P = 0$, compute deployment as **Public Cloud**.
   - If $S = 1$ and $P = 1$, compute deployment as **Private Cloud**.
   - If a system has mixed modules (e.g., Module A has $S = 1, P = 1$ and Module B has $S = 0, P = 0$), compute deployment as **Hybrid Cloud**.

Worked Example:

Deployment Optimization for an Enterprise A bank operates two main software systems. System A manages confidential core banking transactions with steady daily traffic. System B is a promotional website for new credit cards, which experiences sudden massive traffic spikes during marketing campaigns. Determine the optimal cloud deployment model for the bank.

Solution:

1. **Analyze System A (Core Banking):**
   - Security Level: Contains confidential financial data, so $S = 1$.
   - Predictability: Steady daily traffic, so $P = 1$.
   - Result: System A maps to a **Private Cloud**.
2. **Analyze System B (Promotional Website):**
   - Security Level: Publicly accessible marketing information, so $S = 0$.
   - Predictability: Sudden massive traffic spikes, so $P = 0$.
   - Result: System B maps to a **Public Cloud**.
3. **Conclusion:** Because the bank's total infrastructure requires both a Private Cloud for secure, steady workloads and a Public Cloud for public, bursty workloads, the overall optimal architecture is a **Hybrid Cloud**.

CHAPTER 2

Virtualization and Hypervisors

2.1 Introduction to Cloud Virtualization

Virtualization is the process of creating a software-based representation of resources rather than a physical one. In cloud computing, it allows the partitioning of a single physical server into multiple logical servers. While it is a structural concept, resource allocation requires precise computational planning to ensure efficiency without overloading hardware.

Methodology:

Virtual Machine Resource Allocation Algorithm To determine if a physical host can support a requested set of Virtual Machines (VMs), use the following steps:

1. Identify the total physical resources of the host (CPU cores, RAM, Storage).
2. Identify the maximum allowed overcommitment ratio for CPU and RAM (e.g. 2:1 for CPU means 2 virtual CPUs can map to 1 physical CPU core).
3. Calculate the effective available virtual resources:
   - Effective vCPU Capacity = Physical Cores $\times$ CPU Overcommitment Ratio
   - Effective RAM Capacity = Physical RAM $\times$ RAM Overcommitment Ratio
4. Sum the resource requirements for all proposed VMs.
5. Compare the total requested VM resources against the effective available virtual resources. The system is feasible if requested $\leq$ effective available.

Worked Example:

Calculating VM Hosting Feasibility A physical server has 16 CPU cores and 64 GB of RAM. The cloud administrator permits a CPU overcommitment ratio of 3:1 and a RAM overcommitment ratio of 1.2:1. You need to host as many identical VMs as possible where each VM requires 4 vCPUs and 8 GB of RAM. Calculate the maximum number of such VMs the server can host.

Step 1: Calculate Effective Virtual Resources

- Effective vCPUs = 16 cores $\times$ 3 = 48 vCPUs
- Effective RAM = 64 GB $\times$ 1.2 = 76.8 GB

Step 2: Determine constraints based on each resource

- Max VMs based on CPU = $\lfloor 48/4 \rfloor = 12$ VMs
- Max VMs based on RAM = $\lfloor 76.8/8 \rfloor = \lfloor 9.6 \rfloor = 9$ VMs

Step 3: Find the limiting factor The maximum number of VMs is the minimum of the calculated constraints.

$$\text{Maximum VMs} = \min(12, 9) = 9 \text{ VMs}$$

Conclusion: The server can host a maximum of 9 VMs. RAM is the limiting resource.

2.2 Types of Virtualization

Virtualization can be applied at different layers of a computer system including server virtualization, network virtualization, and storage virtualization. The primary goal across all types is resource optimization and server consolidation. By converting physical workloads into virtual ones, organizations maximize hardware usage.

Methodology:

Server Consolidation Analysis Server consolidation involves reducing the number of underutilized physical servers by migrating their workloads into VMs on fewer, more powerful servers.

1. Calculate the **Consolidation Ratio**:

$$\text{Consolidation Ratio} = \frac{\text{Number of Original Physical Servers}}{\text{Number of New Virtualized Hosts}}$$

2. Compute the **Cost Savings**:

$$\text{Old Cost} = \text{Original Physical Servers} \times \text{Cost per Server}$$

$$\text{New Cost} = (\text{New Hosts} \times \text{Cost per Host}) + \text{Hypervisor License Costs}$$

$$\text{Total Savings} = \text{Old Cost} - \text{New Cost}$$

3. Compute the **Power Consumption Reduction**:

$$\text{Power Savings} = (\text{Original Servers} \times \text{Watts/Server}) - (\text{New Hosts} \times \text{Watts/Host})$$

Worked Example:

Server Consolidation and Cost Analysis An IT department currently runs 40 legacy physical servers. Each consumes 400 Watts and costs \$2000 per year to maintain. They plan to consolidate these onto 4 high-end host servers using hardware virtualization. Each new host server consumes 900 Watts and costs \$6000 per year to maintain. The hypervisor license costs \$1500 per host per year. Calculate the consolidation ratio and the annual maintenance and power savings.

Step 1: Calculate the Consolidation Ratio

$$\text{Consolidation Ratio} = \frac{40 \text{ Original Servers}}{4 \text{ Hosts}} = 10 \text{ (or 10:1)}$$

Step 2: Calculate Financial Savings

- Old Maintenance Cost $= 40 \times \$2000 = \$80000$
- New Hardware Cost $= 4 \times \$6000 = \$24000$
- New Software Cost $= 4 \times \$1500 = \$6000$
- Total New Cost $= \$24000 + \$6000 = \$30000$
- Annual Financial Savings $= \$80000 - \$30000 = \$50000$

Step 3: Calculate Power Savings

- Old Power Consumption $= 40 \times 400$ W $= 16000$ W $= 16$ kW
- New Power Consumption $= 4 \times 900$ W $= 3600$ W $= 3.6$ kW
- Power Reduction $= 16$ kW $- 3.6$ kW $= 12.4$ kW

Conclusion: The organization achieves a 10:1 consolidation ratio, saving $50000 annually and reducing power consumption by 12.4 kW.

2.3 Hypervisors and Virtual Machines

A hypervisor (or Virtual Machine Monitor) is the software layer that enables virtualization. It manages the physical resources and allocates them to VMs.

- **Type 1 (Bare-Metal):** Runs directly on the host hardware, offering higher performance.
- **Type 2 (Hosted):** Runs on top of a conventional operating system, introducing extra overhead.

Because the hypervisor sits between the VM and the hardware, it introduces computational and memory overheads that must be factored into deployment planning.

Methodology:

Evaluating Hypervisor Overhead and VM Performance To compute the true performance delivered to a VM, you must subtract the hypervisor overhead.

1. Identify the base performance capacity allocated to the VM (e.g., CPU instructions per second or memory bandwidth).
2. Identify the overhead percentage introduced by the hypervisor O_h.
3. Calculate the **Effective Capacity** for a Type 1 hypervisor:

$$\text{Effective Capacity} = \text{Allocated Capacity} \times \left(1 - \frac{O_h}{100}\right)$$

4. If running a Type 2 hypervisor, you must also account for the host OS overhead O_{os}:

$$\text{Effective Capacity (Type 2)} = \text{Allocated Capacity} \times \left(1 - \frac{O_h + O_{os}}{100}\right)$$

Worked Example:

Computing Effective Processing Power A cloud provider provisions a VM with 3.0 GHz of total processing power. Case A uses a Type 1 hypervisor with an overhead of 4%. Case B uses a Type 2 hypervisor with a hypervisor overhead of 5% and an underlying host OS overhead of 8%. Calculate the effective processing power delivered to the VM in both cases in MHz.

Step 1: Convert base capacity to standard units

$$\text{Allocated Capacity} = 3.0 \text{ GHz} = 3000 \text{ MHz}$$

Step 2: Calculate Effective Power for Case A (Type 1)

- $O_h = 4\%$
- Effective Capacity $= 3000 \times \left(1 - \frac{4}{100}\right) = 3000 \times 0.96 = 2880$ MHz

Step 3: Calculate Effective Power for Case B (Type 2)

- Total Overhead $= O_h + O_{os} = 5\% + 8\% = 13\%$
- Effective Capacity $= 3000 \times \left(1 - \frac{13}{100}\right) = 3000 \times 0.87 = 2610$ MHz

Conclusion: The VM receives 2880 MHz of usable processing power under the Type 1 hypervisor and only 2610 MHz under the Type 2 hypervisor. The bare-metal approach is significantly more efficient for computational workloads.

Methodology:

Memory Ballooning Calculation Memory ballooning is a technique used by hypervisors to reclaim unused memory from active VMs and reallocate it to other VMs in need.

1. Identify total physical RAM and the RAM assigned to each VM.
2. Determine the active memory currently being used by each VM.
3. Define the minimum memory buffer the guest OS needs to prevent swapping.
4. Calculate the memory the balloon driver can inflate to reclaim:

$$\text{Reclaimed Memory} = \text{Assigned RAM} - \text{Active Memory} - \text{Guest OS Buffer}$$

5. Sum the reclaimed memory across all eligible VMs to find the total freed memory.

Worked Example:

Calculating Reclaimed Memory via Ballooning A hypervisor manages two VMs on a physical host. - VM 1 is assigned 8 GB of RAM but is actively using only 3 GB. - VM 2 is assigned 6 GB of RAM but is actively using only 2 GB. The hypervisor policies require a 1 GB buffer to be left untouched in each VM for immediate guest OS operations. Calculate the total amount of memory the hypervisor can reclaim using the balloon driver.

Step 1: Calculate Reclaimed Memory for VM 1

$$\text{Reclaimed}_1 = 8 \text{ GB (Assigned)} - 3 \text{ GB (Active)} - 1 \text{ GB (Buffer)} = 4 \text{ GB}$$

Step 2: Calculate Reclaimed Memory for VM 2

$$\text{Reclaimed}_2 = 6 \text{ GB (Assigned)} - 2 \text{ GB (Active)} - 1 \text{ GB (Buffer)} = 3 \text{ GB}$$

Step 3: Calculate Total Reclaimed Memory

$$\text{Total Reclaimed} = 4 \text{ GB} + 3 \text{ GB} = 7 \text{ GB}$$

Conclusion: The hypervisor successfully reclaims 7 GB of RAM, which can now be allocated to start new VMs without requiring additional physical memory modules.

CHAPTER 3

Data Center Architecture

This chapter focuses on the computational aspects and quantitative analysis of Data Center Architecture. We will explore mathematical models for evaluating data center efficiency, calculating network topology capacities such as oversubscription ratios in modern topologies, and utilizing algorithms for automated scaling and resource allocation.

3.1 Data Center Fundamentals

The fundamental operation of a data center revolves around efficient power utilization and spatial capacity planning.

Methodology:

Calculating Power Usage Effectiveness PUE and DCiE Power Usage Effectiveness (PUE) and Data Center Infrastructure Efficiency (DCiE) are the standard metrics used to evaluate the energy efficiency of a data center.

1. **Identify Total Facility Power:** This includes everything running in the data center (IT equipment, cooling, lighting, UPS inefficiencies).
2. **Identify IT Equipment Power:** The power consumed strictly by the computing, storage, and networking equipment.
3. **Calculate PUE:**
$$PUE = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$
Note: PUE is always ≥ 1.0. *A PUE closer to 1.0 indicates higher efficiency.*
4. **Calculate DCiE:**
$$DCiE = \left(\frac{1}{PUE}\right) \times 100\%$$

Worked Example:

PUE and DCiE Evaluation A data center consumes a total of 1500 kW of power. The cooling systems consume 400 kW, lighting and security systems consume 50 kW, and power delivery losses account for 50 kW. The rest of the power is used by the servers and networking gear. Calculate the PUE and DCiE.
Step 1: Determine IT Equipment Power The IT Equipment Power is the total power minus the non-IT overhead:

$$\text{Non-IT Power} = 400 \text{ kW (cooling)} + 50 \text{ kW (lighting)} + 50 \text{ kW (losses)} = 500 \text{ kW}$$

$$\text{IT Equipment Power} = 1500 \text{ kW} - 500 \text{ kW} = 1000 \text{ kW}$$

Step 2: Calculate PUE

$$PUE = \frac{1500 \text{ kW}}{1000 \text{ kW}} = 1.5$$

Step 3: Calculate DCiE

$$DCiE = \left(\frac{1}{1.5}\right) \times 100\% = 66.67\%$$

Conclusion: The data center has a PUE of 1.5, meaning 66.67% of its power goes to actual IT workloads.

Methodology:

Rack Space and Power Capacity Planning To design the physical layout of a data center, engineers must calculate spatial (Rack Units) and electrical constraints.

1. **Determine Total Required Rack Units RU:** Calculate total RU required for servers, switches, and patch panels. Standard rack size is often 42U.
2. **Determine Rack Count based on Space:**

$$N_{space} = \left\lceil \frac{\text{Total RU}}{\text{Usable RU per Rack}} \right\rceil$$

3. **Determine Rack Count based on Power Limit:**

$$N_{power} = \left\lceil \frac{\text{Total Power Load}}{\text{Max Power Capacity per Rack}} \right\rceil$$

4. **Final Rack Requirement:** The actual number of racks required is $\max(N_{space}, N_{power})$.

Worked Example:

Calculating Required Racks and Power Load You are deploying 150 new 2U servers. Each server consumes 450W of power. You are using standard 42U racks, but reserve 2U per rack for Top-of-Rack (ToR) switches. Each rack's Power Distribution Unit (PDU) can safely supply up to 10 kW. Calculate the minimum number of racks required.

Step 1: Calculate Spatial Requirements Usable RU per rack = 42U − 2U = 40U. Servers per rack based strictly on space = $\lfloor 40\text{U}/2\text{U} \rfloor = 20$ servers. Number of racks for space = $\lceil 150/20 \rceil = \lceil 7.5 \rceil = 8$ racks.

Step 2: Calculate Power Constraints Power per server = 450W = 0.45 kW. Servers per rack based strictly on power = $\lfloor 10 \text{ kW}/0.45 \text{ kW} \rfloor = \lfloor 22.22 \rfloor = 22$ servers. Number of racks for power = $\lceil 150/22 \rceil = \lceil 6.81 \rceil = 7$ racks.

Step 3: Determine Final Rack Count Since we are constrained by both space and power, we must take the maximum.

$$\text{Required Racks} = \max(8, 7) = 8 \text{ racks}$$

Conclusion: 8 racks are required to house the 150 servers without exceeding spatial or electrical constraints.

3.2 Data Center Networking

Data Center networking has shifted from traditional three-tier architectures (Core-Aggregation-Access) to Leaf-Spine (CLOS) architectures to minimize latency and handle east-west traffic.

Methodology:

Calculating Network Oversubscription Ratio Oversubscription is the ratio of peak downlink bandwidth capacity to peak uplink bandwidth capacity at a given network layer.

1. **Calculate Total Downlink Bandwidth:** Number of downward-facing ports × bandwidth per port.
2. **Calculate Total Uplink Bandwidth:** Number of upward-facing ports × bandwidth per port.

3. **Compute Ratio:**

$$\text{Oversubscription Ratio} = \text{Total Downlink Bandwidth} : \text{Total Uplink Bandwidth}$$

Reduce this fraction to a $X : 1$ format. If the ratio is $1 : 1$, the network is non-blocking.

Worked Example:

Oversubscription in an Access Switch A Top-of-Rack (ToR) access switch has forty-eight 10 Gigabit Ethernet (10 GbE) ports connected to servers. It connects to the aggregation layer using four 40 GbE uplink ports. Calculate the oversubscription ratio.

Step 1: Total Downlink Bandwidth

$$48 \text{ ports} \times 10 \text{ Gbps/port} = 480 \text{ Gbps}$$

Step 2: Total Uplink Bandwidth

$$4 \text{ ports} \times 40 \text{ Gbps/port} = 160 \text{ Gbps}$$

Step 3: Compute Ratio

$$\text{Ratio} = 480 : 160$$

Dividing both sides by 160 gives:

$$\text{Ratio} = 3 : 1$$

Conclusion: The oversubscription ratio is 3:1. If all servers transmit at line rate simultaneously, 66% of the traffic will be dropped or delayed.

Methodology:

Leaf Spine Capacity Planning In a two-tier Leaf-Spine topology, every leaf switch connects to every spine switch.

1. **Identify Switch Specifications:** Let P_{leaf} be the total ports on a leaf switch, and P_{spine} be the total ports on a spine switch.
2. **Uplinks and Downlinks:** A leaf switch typically reserves half its ports for server downlinks (D_{leaf}) and half for spine uplinks (U_{leaf}).
3. **Maximum Number of Leaves:** The maximum number of leaf switches is bound by the number of ports on a spine switch: Max Leaves $= P_{spine}$.
4. **Maximum Number of Spines:** The maximum number of spine switches is bound by the number of uplink ports on a leaf switch: Max Spines $= U_{leaf}$.
5. **Total Server Capacity:**

$$\text{Total Servers} = \text{Leaves} \times D_{leaf}$$

Worked Example:

Sizing a Leaf Spine Network You are building a non-blocking Leaf-Spine network. You have 64-port switches available to use as both leaves and spines. All ports are 100 GbE. Half of each leaf switch's ports are allocated for uplinks, and half for downlinks. Calculate the maximum number of servers this fabric can support and the total number of switches required.

Step 1: Determine Leaf Port Allocation Total leaf ports $P_{leaf} = 64$. Downlinks to servers $D_{leaf} = 32$. Uplinks to spines $U_{leaf} = 32$.

Step 2: Determine Maximum Switches Max Spine Switches $= U_{leaf} = 32$ spines. Max Leaf Switches $= P_{spine} = 64$ leaves.

Step 3: Calculate Total Servers

$$\text{Total Servers} = \text{Max Leaves} \times D_{leaf} = 64 \times 32 = 2048 \text{ servers}$$

Step 4: Calculate Total Switches

$$\text{Total Switches} = \text{Leaves} + \text{Spines} = 64 + 32 = 96 \text{ switches}$$

Conclusion: This architecture supports exactly 2048 servers at a 1:1 oversubscription ratio using 96 total switches.

3.3 Data Center Automation and Scaling

Automation in the data center enables dynamic scaling. Computational algorithms govern the placement of Virtual Machines (VMs) and the provisioning of new horizontal nodes.

Methodology:

Server Consolidation via First Fit Algorithm VM placement can be modeled as a Bin Packing problem. The First-Fit Decreasing (FFD) algorithm is a common heuristic used to minimize the number of active physical hosts.

1. **Sort VMs:** Order the list of requested VMs in descending order based on their dominant resource requirement (usually CPU or RAM).
2. **Initialize Hosts:** Start with an empty list of active physical hosts.
3. **Allocate:** For each VM in the sorted list:
   (a) Scan the active physical hosts sequentially.
   (b) Place the VM in the *first* host that has enough remaining capacity.
   (c) If no active host has sufficient capacity, boot a new physical host and place the VM there.
4. **Calculate Utilization:**

$$\text{System Utilization} = \frac{\text{Total VM Resources Required}}{\text{Active Hosts} \times \text{Capacity per Host}} \times 100\%$$

Worked Example:

VM Placement using First Fit Decreasing A data center uses physical hosts that each have 16 CPU cores. You need to provision 6 VMs with the following CPU core requirements: VM1: 8 cores, VM2: 4 cores, VM3: 10 cores, VM4: 6 cores, VM5: 4 cores, VM6: 2 cores. Calculate how many hosts are needed using the First-Fit Decreasing algorithm and calculate the CPU utilization.
Step 1: Sort VMs Descending Sorted list: VM3 (10), VM1 (8), VM4 (6), VM2 (4), VM5 (4), VM6 (2).
Step 2 and 3: Allocate to Hosts (Capacity = 16)

- **VM3 (10):** Requires 10. Open Host 1. Remaining capacity in Host 1 = 6.
- **VM1 (8):** Requires 8. Host 1 has 6 (insufficient). Open Host 2. Remaining in Host 2 = 8.
- **VM4 (6):** Requires 6. Host 1 has 6 (exact match). Place in Host 1. Remaining in Host 1 = 0.
- **VM2 (4):** Requires 4. Host 1 is full. Host 2 has 8 (sufficient). Place in Host 2. Remaining in Host 2 = 4.
- **VM5 (4):** Requires 4. Host 1 is full. Host 2 has 4 (exact match). Place in Host 2. Remaining in Host 2 = 0.

- **VM6 (2):** Requires 2. Host 1 full. Host 2 full. Open Host 3. Remaining in Host 3 = 14.

Step 4: Calculate Utilization Total required cores = 10 + 8 + 6 + 4 + 4 + 2 = 34 cores. Active Hosts = 3. Total capacity = $3 \times 16 = 48$ cores.

$$\text{Utilization} = \left(\frac{34}{48}\right) \times 100\% = 70.83\%$$

Conclusion: 3 physical hosts are required.

Methodology:

Horizontal Scaling and Target Utilization In automated scaling groups (Autoscaling), new nodes are provisioned based on traffic load and a target CPU/Memory utilization threshold to leave buffer room for burst traffic.

1. **Determine Total Required Capacity:** Given the total incoming Requests Per Second (RPS) and the max RPS a single node can handle at 100% load:

$$\text{Base Nodes} = \frac{\text{Total RPS}}{\text{Max RPS per node}}$$

2. **Apply Target Utilization Buffer:**

$$\text{Required Nodes} = \left\lceil \frac{\text{Base Nodes}}{\text{Target Utilization Percentage}} \right\rceil$$

Worked Example:

Calculating Autoscaling Cluster Size An e-commerce web application expects a peak load of 15,000 requests per second (RPS) during a flash sale. Performance testing shows that a single server instance can process 1,200 RPS before hitting 100% CPU utilization. The autoscaling policy mandates that average cluster CPU utilization must not exceed 70% to maintain latency SLAs. Calculate the number of server instances required.

Step 1: Calculate Base Nodes at 100% Load

$$\text{Base Nodes} = \frac{15000 \text{ RPS}}{1200 \text{ RPS/node}} = 12.5 \text{ nodes}$$

Step 2: Apply Target Utilization Buffer Target utilization = 70% = 0.70.

$$\text{Required Nodes} = \frac{12.5}{0.70} = 17.857$$

Step 3: Round Up Since we cannot deploy a fraction of a server, we round up to the nearest whole integer:

$$\text{Required Nodes} = \lceil 17.857 \rceil = 18 \text{ instances}$$

Conclusion: The autoscaling group must spin up 18 instances to handle the 15,000 RPS load while keeping average utilization under 70%.

CHAPTER 4

Cloud Storage and Database Services

This chapter focuses on the computational aspects of cloud storage and database services. We will explore quantitative methods for calculating storage costs estimating data transfer times provisioning NoSQL database read/write capacity and allocating data shards.

4.1 Cloud Storage Solutions

Cloud storage provides scalable infrastructure for storing objects blocks and files. The primary computational problems in cloud storage involve calculating the financial cost of storing and accessing data and estimating the time required to migrate large datasets over network links.

Methodology:

Calculating Cloud Storage Costs Cloud storage pricing models typically depend on three factors: storage volume data transfer (bandwidth) and request operations (API calls). To calculate the total monthly cost of a cloud storage service use the following steps:

1. **Storage Cost**: Multiply the total data stored (in GB) by the monthly cost per GB for the chosen storage tier.
$$\text{Storage Cost} = \text{Total Data (GB)} \times \text{Price per GB}$$
2. **Request Cost**: Identify the total number of operations (e.g. PUT GET requests). Divide by the billing unit (usually per 1000 or 10000 requests) and multiply by the rate.
$$\text{Request Cost} = \left(\frac{\text{Total Requests}}{\text{Billing Unit}} \right) \times \text{Price per Unit}$$
3. **Data Transfer Cost**: Calculate the cost of data moving *out* of the cloud (egress). Ingress (data moving in) is usually free.
$$\text{Transfer Cost} = \text{Total Data Out (GB)} \times \text{Price per GB Out}$$
4. **Total Monthly Cost**: Sum the storage request and data transfer costs.

Worked Example:

Calculate Monthly Object Storage Cost A company stores 5 TB of data in an object storage bucket. The storage pricing is \$0.023 per GB per month. During the month the application makes 150000 PUT requests (priced at \$0.005 per 1000 requests) and 2000000 GET requests (priced at \$0.0004 per 1000 requests). The application also transfers 500 GB of data out to the internet billed at \$0.09 per GB. Calculate the total monthly cloud storage bill.

Step 1: Calculate the Storage Cost Convert terabytes to gigabytes (using base-2 standard 1 TB = 1024 GB):
$$5 \text{ TB} = 5 \times 1024 \text{ GB} = 5120 \text{ GB}$$

$$\text{Storage Cost} = 5120 \text{ GB} \times \$0.023/\text{GB} = \$117.76$$

Step 2: Calculate the Request Costs For PUT requests:

$$\text{PUT Cost} = \left(\frac{150000}{1000}\right) \times \$0.005 = 150 \times \$0.005 = \$0.75$$

For GET requests:

$$\text{GET Cost} = \left(\frac{2000000}{1000}\right) \times \$0.0004 = 2000 \times \$0.0004 = \$0.80$$

$$\text{Total Request Cost} = \$0.75 + \$0.80 = \$1.55$$

Step 3: Calculate the Data Transfer Cost

$$\text{Transfer Cost} = 500 \text{ GB} \times \$0.09/\text{GB} = \$45.00$$

Step 4: Calculate Total Cost

$$\text{Total Monthly Cost} = \$117.76 + \$1.55 + \$45.00 = \$164.31$$

Methodology:

Calculating Network Data Transfer Time Migrating large volumes of data to the cloud requires significant network bandwidth. To calculate the time required to upload or download data:

1. **Convert Data Size to Bits**: Network speeds are measured in bits per second (bps) while data is measured in Bytes (B).
$$\text{Data (bits)} = \text{Total Data (Bytes)} \times 8$$
2. **Identify Effective Bandwidth**: Convert the network speed to bits per second. For example 1 Mbps $= 10^6$ bps and 1 Gbps $= 10^9$ bps.
3. **Calculate Time**:
$$\text{Time (seconds)} = \frac{\text{Data (bits)}}{\text{Effective Bandwidth (bps)}}$$
4. **Convert to Readable Units**: Divide by 60 for minutes 3600 for hours or 86400 for days.

Worked Example:

Calculate Time to Migrate Data to Cloud An organization needs to migrate 2 TB (Terabytes) of backup data to a cloud storage archive. Their office has a dedicated internet connection with an upload speed of 100 Mbps. Assuming 80% of the bandwidth is available for this transfer calculate the estimated time in hours required to complete the upload. (Use 1 TB $= 10^{12}$ Bytes).

Step 1: Convert Data Size to Bits Total Data $= 2$ TB $= 2 \times 10^{12}$ Bytes.

$$\text{Data in bits} = 2 \times 10^{12} \times 8 = 16 \times 10^{12} \text{ bits}$$

Step 2: Calculate Effective Bandwidth Total Bandwidth $= 100$ Mbps $= 100 \times 10^6$ bps $= 10^8$ bps. Effective Bandwidth (80% of total) $= 0.80 \times 10^8$ bps $= 8 \times 10^7$ bps.

Step 3: Calculate Time in Seconds

$$\text{Time (seconds)} = \frac{16 \times 10^{12} \text{ bits}}{8 \times 10^7 \text{ bps}} = 2 \times 10^5 \text{ seconds}$$

$$\text{Time (seconds)} = 200000 \text{ seconds}$$

Step 4: Convert to Hours

$$\text{Time (hours)} = \frac{200000}{3600} \approx 55.56 \text{ hours}$$

The transfer will take approximately 55.56 hours.

4.2 Cloud Databases

Cloud databases offer managed Relational (SQL) and NoSQL environments. A key engineering task when using cloud NoSQL databases (like Amazon DynamoDB or Azure Cosmos DB) is provisioning the exact read and write capacity to optimize performance and control costs.

Methodology:

Calculating Provisioned Capacity for Cloud NoSQL Many NoSQL cloud databases use Read Capacity Units (RCU) and Write Capacity Units (WCU) for billing and throttling. The capacity required depends on the size of the item and the desired operations per second.

1. **Calculate Read Capacity Units (RCU)**:
   - One RCU represents one strongly consistent read per second for an item up to 4 KB in size.
   - Round the item size *up* to the nearest multiple of 4 KB.
   - Divide by 4 KB to find RCUs per read.
   - Multiply by the number of reads per second.
   - *Note*: If the system uses "Eventually Consistent" reads divide the final RCU total by 2.
2. **Calculate Write Capacity Units (WCU)**:
   - One WCU represents one write per second for an item up to 1 KB in size.
   - Round the item size *up* to the nearest multiple of 1 KB.
   - Divide by 1 KB to find WCUs per write.
   - Multiply by the number of writes per second.

Worked Example:

Compute RCU and WCU for a Workload A cloud NoSQL database table stores user profile data. Each user profile item is 6.5 KB in size. The application requires 40 strongly consistent reads per second and 25 writes per second. Calculate the total provisioned RCU and WCU required for this table.

Step 1: Calculate RCU for the Table

- Item size: 6.5 KB.
- Round up to the nearest 4 KB multiple: The next multiple of 4 after 6.5 is 8 KB.
- Calculate RCUs per read operation: $\frac{8 \text{ KB}}{4 \text{ KB}} = 2$ RCUs per read.
- Multiply by operations per second: 40 reads/sec $\times$ 2 RCUs/read $= 80$ RCUs.

Step 2: Calculate WCU for the Table

- Item size: 6.5 KB.
- Round up to the nearest 1 KB multiple: The next multiple of 1 after 6.5 is 7 KB.
- Calculate WCUs per write operation: $\frac{7 \text{ KB}}{1 \text{ KB}} = 7$ WCUs per write.
- Multiply by operations per second: 25 writes/sec $\times$ 7 WCUs/write $= 175$ WCUs.

Result: The table must be provisioned with 80 RCUs and 175 WCUs.

Methodology:

Database Sharding and Key Distribution Sharding is a horizontal scaling technique that distributes data across multiple database nodes (shards). A common computational method is **Hash-based Sharding** which ensures an even distribution of data.

1. **Determine the Partition Key**: Select the attribute used to distribute data (e.g. UserID).
2. **Apply a Hash Function**: Convert the partition key into a numerical hash value $H(K)$.
3. **Apply the Modulo Operator**: Divide the hash value by the total number of shards (N) and take the remainder.
$$\text{Shard ID} = H(K) \pmod{N}$$
4. **Assign Data**: The resulting Shard ID (from 0 to $N-1$) indicates which physical server will store the record.

Worked Example:

Determine Shard Assignment for User IDs A cloud database cluster is configured with 4 shards ($N = 4$) numbered 0 1 2 and 3. The system uses a hash-based sharding algorithm where the hash function simply sums the ASCII values of the characters in the User ID. Determine which shards will store the data for 'UserA' and 'UserB'. (Assume ASCII values: U=85 s=115 e=101 r=114 A=65 B=66).

Step 1: Calculate Hash for 'UserA' Sum of ASCII values for U-s-e-r-A:

$$H(\text{UserA}) = 85 + 115 + 101 + 114 + 65 = 480$$

Step 2: Assign Shard for 'UserA' Apply modulo operation with $N = 4$:

$$\text{Shard ID} = 480 \pmod{4}$$

Since 480 is perfectly divisible by 4 ($480 = 4 \times 120$) the remainder is 0. 'UserA' is assigned to **Shard 0**.

Step 3: Calculate Hash for 'UserB' Sum of ASCII values for U-s-e-r-B:

$$H(\text{UserB}) = 85 + 115 + 101 + 114 + 66 = 481$$

Step 4: Assign Shard for 'UserB' Apply modulo operation with $N = 4$:

$$\text{Shard ID} = 481 \pmod{4}$$

$481 = (4 \times 120) + 1$. The remainder is 1. 'UserB' is assigned to **Shard 1**.

CHAPTER 5

Cloud Security and Compliance

Cloud security requires rigorous quantitative evaluation to assess risks and ensure data protection. This chapter covers the computational aspects of risk analysis, access control, encryption algorithms, and data redundancy configurations.

5.1 Security in the Cloud

Security in cloud computing involves assessing potential threats quantitatively and designing logical access controls to authorize users dynamically.

Methodology:

Quantitative Risk Assessment Quantitative risk assessment involves assigning numerical values to assets, threats, and potential losses to prioritize security investments in cloud infrastructure.
Formulas:

1. **Single Loss Expectancy (SLE):** The monetary loss expected from a single incident.

$$\text{SLE} = \text{Asset Value (AV)} \times \text{Exposure Factor (EF)}$$

   where EF is the percentage of asset loss caused by the identified threat.

2. **Annualized Rate of Occurrence (ARO):** The estimated frequency of the threat occurring within a year.

3. **Annualized Loss Expectancy (ALE):** The total expected financial loss per year.

$$\text{ALE} = \text{SLE} \times \text{ARO}$$

Worked Example:

Calculating Annualized Loss Expectancy A company hosts its primary database in a cloud environment. The total value of the database (Asset Value) is \$500000. A security analyst determines that a DDoS attack would compromise 20% of the database's value (Exposure Factor). Historical data suggests that such an attack is likely to occur once every 4 years. Calculate the Single Loss Expectancy (SLE) and Annualized Loss Expectancy (ALE).
Solution:

1. **Identify given values:**
   - Asset Value (AV) = \$500000
   - Exposure Factor (EF) = 20% = 0.20
   - ARO = 1 attack / 4 years = 0.25 attacks per year
2. **Calculate Single Loss Expectancy (SLE):**

$$\text{SLE} = \text{AV} \times \text{EF}$$

$$\text{SLE} = 500000 \times 0.20 = 100000$$

The expected loss per incident is \$100000.

3. **Calculate Annualized Loss Expectancy (ALE):**

$$\text{ALE} = \text{SLE} \times \text{ARO}$$

$$\text{ALE} = 100000 \times 0.25 = 25000$$

The annual expected loss is \$25000. This is the maximum amount the company should economically spend annually to mitigate this specific risk.

Methodology:

Evaluating Role Based Access Control In cloud environments, Role-Based Access Control (RBAC) determines user permissions based on roles. Access rights are computed using logical matrices.
Evaluation Steps:

1. Identify the user and map them to their assigned roles.
2. Retrieve the permissions (Read, Write, Execute) granted to each assigned role.
3. Compute the **Effective Permission** by applying the logical OR operation across all roles for a given resource. If an explicit `DENY` policy exists, it overrides all `ALLOW` policies.

Worked Example:

Determining Effective User Permissions A cloud storage system has three roles: `Guest`, `Developer`, and `Admin`. User *Alice* is assigned the roles of `Guest` and `Developer`.
The Role-to-Permission mapping for a specific configuration file is:

Role	Read	Write	Execute
Guest	ALLOW	DENY	DENY
Developer	ALLOW	ALLOW	DENY
Admin	ALLOW	ALLOW	ALLOW

Compute Alice's effective permissions for the configuration file.
Solution:

1. **Identify Alice's Roles:** `Guest` and `Developer`.
2. **Retrieve Permissions:**
   - `Guest` permissions: {Read: ALLOW, Write: DENY, Execute: DENY}
   - `Developer` permissions: {Read: ALLOW, Write: ALLOW, Execute: DENY}
3. **Calculate Effective Permissions:**
   - **Read:** `Guest` (ALLOW) OR `Developer` (ALLOW) → **ALLOW**
   - **Write:** The system processes explicit Denies. `Guest` (DENY) overrides `Developer` (ALLOW) → **DENY**.
   - **Execute:** `Guest` (DENY) OR `Developer` (DENY) → **DENY**
4. **Final Permissions:** Alice can **Read** the file but cannot **Write** or **Execute** it due to the overriding explicit DENY from her Guest role.

5.2 Data Security in Cloud

Protecting data at rest and in transit requires cryptographic algorithms and data redundancy techniques to ensure both confidentiality and availability.

Methodology:

RSA Algorithm for Data Encryption The RSA (Rivest-Shamir-Adleman) algorithm is widely used in cloud computing to secure data transmissions. It involves generating a public key for encryption and a private key for decryption.

Steps for Key Generation and Encryption:

1. Choose two distinct prime numbers, p and q.
2. Compute $n = p \times q$. The value n is used as the modulus for both keys.
3. Compute Euler's totient function, $\phi(n) = (p-1) \times (q-1)$.
4. Choose an integer e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$. The Public Key is (e, n).
5. Compute the private key exponent d such that $(d \times e) \equiv 1 \pmod{\phi(n)}$. The Private Key is (d, n).
6. **Encryption:** For a plaintext message M (where $M < n$), the ciphertext C is calculated as:
$$C = M^e \pmod{n}$$
7. **Decryption:** To recover the message, compute:
$$M = C^d \pmod{n}$$

Worked Example:

RSA Encryption and Decryption Steps Perform RSA key generation, encrypt the message $M = 9$, and then decrypt the ciphertext given the prime numbers $p = 5$ and $q = 11$, and the public exponent $e = 3$.

Solution:

1. **Compute n:**
$$n = p \times q = 5 \times 11 = 55$$
2. **Compute $\phi(n)$:**
$$\phi(n) = (p-1) \times (q-1) = 4 \times 10 = 40$$
3. **Find Private Exponent d:** We need $d \times e \equiv 1 \pmod{40}$, which means $3d = 1 + 40k$ for some integer k.
   - If $k = 1$: $1 + 40(1) = 41$, not divisible by 3.
   - If $k = 2$: $1 + 40(2) = 81$, divisible by 3.
   - $d = 81/3 = 27$.

   The Private Key is $(d = 27, n = 55)$.
4. **Encrypt the Message ($M = 9$):**
$$C = M^e \pmod{n}$$
$$C = 9^3 \pmod{55}$$
$$C = 729 \pmod{55}$$
   Divide 729 by 55: $55 \times 13 = 715$.
$$C = 729 - 715 = 14$$
   The ciphertext is **14**.

5. **Decryption ($C = 14$):**

$$M = C^d \pmod{n}$$

$$M = 14^{27} \pmod{55}$$

Using modular exponentiation rules to simplify: $14^2 = 196 \equiv 31 \pmod{55}$
$14^4 = 31^2 = 961 \equiv 26 \pmod{55}$
$14^8 = 26^2 = 676 \equiv 16 \pmod{55}$
$14^{16} = 16^2 = 256 \equiv 36 \pmod{55}$
Now construct $14^{27} = 14^{16} \times 14^8 \times 14^2 \times 14^1$:
$14^{27} \equiv 36 \times 16 \times 31 \times 14 \pmod{55}$
$36 \times 16 = 576 \equiv 26 \pmod{55}$
$31 \times 14 = 434 \equiv 49 \pmod{55}$
$26 \times 49 = 1274 \equiv 9 \pmod{55}$
The decrypted message is **9**, matching the original plaintext.

Methodology:

Data Availability using RAID Redundant Array of Independent Disks (RAID) provides data security through redundancy and hardware fault tolerance. Let N be the total number of identical drives and S be the storage capacity of a single drive.
Capacity Formulas:

1. **RAID 0 (Striping):** No redundancy. Usable Capacity $= N \times S$. Fault tolerance $= 0$ drives.
2. **RAID 1 (Mirroring):** Complete redundancy. Usable Capacity $= (N \times S)/2$ (assuming N is even). Fault tolerance $= 1$ drive per mirrored pair.
3. **RAID 5 (Striping with Parity):** Distributed parity. Usable Capacity $= (N - 1) \times S$. Fault tolerance $= 1$ drive. Minimum 3 drives required.
4. **RAID 6 (Striping with Double Parity):** Double distributed parity. Usable Capacity $= (N-2) \times S$. Fault tolerance $= 2$ drives. Minimum 4 drives required.

Worked Example:

Calculating RAID Storage Capacity A cloud storage cluster is being provisioned with 6 hard drives, each having a capacity of 4 TB. Calculate the total usable capacity and the number of drives that can fail without data loss for RAID 1, RAID 5, and RAID 6 configurations.
Solution:

1. **Identify given values:** Number of drives $(N) = 6$, Capacity per drive $(S) = 4$ TB.
2. **RAID 1 Analysis:**
   - **Usable Capacity:** $(N \times S)/2 = (6 \times 4)/2 = 24/2 = 12$ TB.
   - **Fault Tolerance:** Safely guaranteed for **1 drive failure** (though it can survive up to 3 failures if they occur in different mirrored pairs).
3. **RAID 5 Analysis:**
   - **Usable Capacity:** $(N - 1) \times S = (6 - 1) \times 4 = 5 \times 4 = 20$ TB.
   - **Fault Tolerance:** Any **1 drive** can fail without losing data.
4. **RAID 6 Analysis:**
   - **Usable Capacity:** $(N - 2) \times S = (6 - 2) \times 4 = 4 \times 4 = 16$ TB.
   - **Fault Tolerance:** Any **2 drives** can fail simultaneously without losing data.

CHAPTER 6

Emerging Technologies with Cloud Computing

6.1 Introduction to Emerging Technologies in the Cloud

The integration of emerging technologies such as Internet of Things (IoT), Edge Computing, and Serverless architectures with cloud platforms requires rigorous computational analysis to ensure performance, cost-efficiency, and scalability. While these technologies introduce new architectural paradigms, planning and managing them relies heavily on mathematical modeling. This section focuses on the models and algorithms used to quantify these cloud integrations.

Methodology:

Edge Computing Latency and Bandwidth Analysis Edge computing reduces the latency and bandwidth required to send data to a centralized cloud by processing data closer to the source. The computational analysis involves calculating data rates and the resulting bandwidth savings.

1. **Determine Total Raw Data Generation Rate (R_{raw}):**

$$R_{raw} = N \times S \times f$$

   where N is the number of devices, S is the data size per payload, and f is the transmission frequency (messages per second).

2. **Calculate Data Filtered at the Edge (R_{edge}):**

$$R_{edge} = R_{raw} \times \text{Filtration Rate}$$

3. **Calculate Effective Bandwidth to Cloud (B_{cloud}):**

$$B_{cloud} = R_{raw} - R_{edge}$$

4. **Calculate Bandwidth Savings (ΔB):**

$$\Delta B = R_{raw} - B_{cloud} = R_{edge}$$

Worked Example:

Calculating Bandwidth Savings with Edge Computing A smart factory deploys 2000 IoT vibration sensors. Each sensor generates a 50 KB data payload every 2 seconds. An edge gateway is installed that processes the data locally and filters out 95% of the routine data, sending only anomalies and summarized reports to the central cloud. Calculate the raw bandwidth required without edge computing, the effective bandwidth with edge computing, and the total bandwidth saved in Megabytes per second (MB/s). (Assume 1 MB = 1000 KB).

Step 1: Determine Total Raw Data Generation Rate (R_{raw})

- Number of sensors (N) = 2000

- Payload size (S) = 50 KB
- Frequency (f) = 1 message / 2 seconds = 0.5 messages/sec

$$R_{raw} = 2000 \times 50 \text{ KB} \times 0.5 \text{ /sec} = 50,000 \text{ KB/s}$$

Convert to MB/s:

$$R_{raw} = \frac{50,000}{1000} = 50 \text{ MB/s}$$

Step 2: Calculate Data Filtered at the Edge (R_{edge}) The edge gateway filters out 95% of the data.

$$R_{edge} = 50 \text{ MB/s} \times 0.95 = 47.5 \text{ MB/s}$$

Step 3: Calculate Effective Bandwidth to Cloud (B_{cloud})

$$B_{cloud} = R_{raw} - R_{edge} = 50 \text{ MB/s} - 47.5 \text{ MB/s} = 2.5 \text{ MB/s}$$

Step 4: Calculate Bandwidth Savings (ΔB)

$$\Delta B = 47.5 \text{ MB/s}$$

Conclusion: By implementing edge computing, the factory reduces its required cloud uplink bandwidth from 50 MB/s to just 2.5 MB/s, saving 47.5 MB/s of continuous bandwidth.

Methodology:

Serverless Computing Cost Estimation Serverless platforms Function-as-a-Service charge based on the number of requests and the duration of code execution multiplied by the memory allocated to the function.

1. **Calculate Total Execution Time (T_{exec}):**

$$T_{exec} \text{ (in seconds)} = \text{Total Requests} \times \text{Average Duration (seconds)}$$

2. **Calculate Compute Allocation (C_{alloc} in GB-seconds):**

$$C_{alloc} = T_{exec} \times \text{Allocated Memory (in GB)}$$

3. **Calculate Compute Cost:**

$$\text{Compute Cost} = C_{alloc} \times \text{Price per GB-second}$$

4. **Calculate Request Cost:**

$$\text{Request Cost} = \left(\frac{\text{Total Requests}}{1,000,000}\right) \times \text{Price per Million Requests}$$

5. **Calculate Total Monthly Cost:**

$$\text{Total Cost} = \text{Compute Cost} + \text{Request Cost}$$

Worked Example:

Estimating Monthly Cost for a Serverless Architecture A cloud-native application uses a serverless function for image processing. The function is invoked 8,000,000 times per month. The average execution time per invocation is 400 milliseconds. The function is allocated 1024 MB (1 GB) of memory. The cloud provider pricing is: \$0.000016 per GB-second of compute and \$0.20 per 1 million requests. Calculate the total monthly cost for this serverless function.

Step 1: Calculate Total Execution Time (T_{exec})

- Total Requests $= 8,000,000$
- Average Duration $= 400$ ms $= 0.4$ seconds

$$T_{exec} = 8,000,000 \times 0.4 = 3,200,000 \text{ seconds}$$

Step 2: Calculate Compute Allocation (C_{alloc})

- Memory Allocated $= 1$ GB

$$C_{alloc} = 3,200,000 \text{ seconds} \times 1 \text{ GB} = 3,200,000 \text{ GB-seconds}$$

Step 3: Calculate Compute Cost

$$\text{Compute Cost} = 3,200,000 \text{ GB-s} \times \$0.000016/\text{GB-s} = \$51.20$$

Step 4: Calculate Request Cost

$$\text{Request Cost} = \left(\frac{8,000,000}{1,000,000}\right) \times \$0.20 = 8 \times \$0.20 = \$1.60$$

Step 5: Calculate Total Monthly Cost

$$\text{Total Cost} = \$51.20 + \$1.60 = \$52.80$$

Conclusion: The total monthly cost for running the serverless image processing function is \$52.80.

Methodology:

IoT Data Ingestion and Storage Calculation For large-scale IoT systems connected to the cloud, planning data lake storage capacity requires exact calculations of incoming data volume over the retention lifecycle.

1. **Calculate Daily Message Volume (V_{daily}):**

$$V_{daily} = N \times \left(\frac{86400}{T_{interval}}\right)$$

where N is the number of devices, 86400 is the number of seconds in a day, and $T_{interval}$ is the transmission interval in seconds.

2. **Calculate Daily Data Size (S_{daily}):**

$$S_{daily} = V_{daily} \times \text{Message Payload Size}$$

3. **Calculate Total Storage Requirement (S_{total}):**

$$S_{total} = S_{daily} \times \text{Retention Period (Days)}$$

Worked Example:

IoT Sensor Network Storage Requirements An agricultural cloud platform monitors 15,000 soil moisture sensors. Each sensor transmits a 4 KB payload containing moisture, temperature, and pH data every 5 minutes. The data must be retained in a cloud data lake for 5 years (1825 days) for machine learning analysis. Calculate the total storage capacity required in Terabytes (TB), assuming 1 TB $= 10^9$ KB.

Step 1: Calculate Daily Message Volume (V_{daily})

- Number of sensors (N) $= 15,000$
- Transmission interval ($T_{interval}$) $= 5$ minutes $= 300$ seconds

$$\text{Messages per sensor per day} = \frac{86400}{300} = 288 \text{ messages}$$

$$V_{daily} = 15,000 \times 288 = 4,320,000 \text{ messages/day}$$

Step 2: Calculate Daily Data Size (S_{daily})

- Payload Size = 4 KB

$$S_{daily} = 4,320,000 \times 4 \text{ KB} = 17,280,000 \text{ KB/day}$$

Step 3: Calculate Total Storage Requirement (S_{total})

- Retention Period = 1825 days

$$S_{total} = 17,280,000 \text{ KB/day} \times 1825 \text{ days} = 31,536,000,000 \text{ KB}$$

Step 4: Convert to Terabytes (TB) Using the base-10 network standard conversion 1 TB = 10^9 KB:

$$S_{total} = \frac{31,536,000,000}{10^9} = 31.536 \text{ TB}$$

Conclusion: The agricultural platform requires 31.536 TB of cloud storage capacity to retain the data for 5 years.

Appendix: Key Formulae

This appendix provides a quick reference to the key formulae and mathematical relationships discussed in this book.

Unit 1: Introduction to Cloud Computing

Cost Savings (Economic Benefit)

Calculates the economic benefit of moving from local infrastructure to cloud computing.

$$E = C_{\text{local}} - C_{\text{cloud}}$$

Variability Ratio

Measures the fluctuation in resource demand by comparing peak utilization to average utilization.

$$V = \frac{\text{Peak Demand}}{\text{Average Demand}}$$

Unit 2: Virtualization

Consolidation Ratio

Determines the efficiency of virtualizing physical servers into fewer virtualized hosts.

$$\text{Consolidation Ratio} = \frac{\text{Number of Original Physical Servers}}{\text{Number of New Virtualized Hosts}}$$

Effective Capacity (Type 1 Hypervisor)

Calculates the available capacity on a bare-metal hypervisor by subtracting hypervisor overhead (O_h).

$$\text{Effective Capacity} = \text{Allocated Capacity} \times \left(1 - \frac{O_h}{100}\right)$$

Effective Capacity (Type 2 Hypervisor)

Calculates available capacity on a hosted hypervisor, accounting for both hypervisor overhead (O_h) and host OS overhead (O_{os}).

$$\text{Effective Capacity (Type 2)} = \text{Allocated Capacity} \times \left(1 - \frac{O_h + O_{os}}{100}\right)$$

Reclaimed Memory

Calculates the amount of memory that can be reclaimed from a VM.

$$\text{Reclaimed Memory} = \text{Assigned RAM} - \text{Active Memory} - \text{Guest OS Buffer}$$

Power Savings

Computes the energy savings achieved through server consolidation.

$$\text{Power Savings} = (\text{Original Servers} \times \text{Watts/Server}) - (\text{New Hosts} \times \text{Watts/Host})$$

Cost Analysis

Calculates the cost of traditional physical infrastructure, the cost of a new virtualized environment, and the resulting financial savings.

$$\text{Old Cost} = \text{Original Physical Servers} \times \text{Cost per Server}$$

$$\text{New Cost} = (\text{New Hosts} \times \text{Cost per Host}) + \text{Hypervisor License Costs}$$

$$\text{Total Savings} = \text{Old Cost} - \text{New Cost}$$

Unit 3: Cloud Infrastructure and Data Center

Power Usage Effectiveness (PUE)

Measures the energy efficiency of a data center. An ideal PUE is 1.0.

$$PUE = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$

Data Center Infrastructure Efficiency (DCiE)

The inverse of PUE, expressed as a percentage.

$$DCiE = \left(\frac{1}{PUE}\right) \times 100\%$$

System Utilization

Measures the percentage of active host capacity currently being used by virtual machines.

$$\text{System Utilization} = \frac{\text{Total VM Resources Required}}{\text{Active Hosts} \times \text{Capacity per Host}} \times 100\%$$

Oversubscription Ratio

Ratio of downlink bandwidth (facing servers) to uplink bandwidth (facing the core network) in a switch.

$$\text{Oversubscription Ratio} = \text{Total Downlink Bandwidth} : \text{Total Uplink Bandwidth}$$

Capacity Planning: Nodes Required

Estimates the required number of compute nodes based on workload requests per second (RPS) and a target utilization threshold.

$$\text{Base Nodes} = \frac{\text{Total RPS}}{\text{Max RPS per node}}$$

$$\text{Required Nodes} = \left\lceil \frac{\text{Base Nodes}}{\text{Target Utilization Percentage}} \right\rceil$$

Rack Sizing Requirements

Determines the number of server racks needed based on physical space (Rack Units) and power constraints.

$$N_{space} = \left\lceil \frac{\text{Total RU}}{\text{Usable RU per Rack}} \right\rceil$$

$$N_{power} = \left\lceil \frac{\text{Total Power Load}}{\text{Max Power Capacity per Rack}} \right\rceil$$

Clos Network (Leaf-Spine) Scale

Calculates the total number of switches and the maximum number of servers in a leaf-spine architecture.

$$\text{Total Switches} = \text{Leaves} + \text{Spines}$$

$$\text{Total Servers} = \text{Leaves} \times D_{leaf}$$

(Where D_{leaf} is the number of downlink ports per leaf switch).

Unit 4: Cloud Storage and Data Transfer

Data Conversion

Converts data sizes to standard bit notation for bandwidth calculations.

$$\text{Data (bits)} = \text{Total Data (Bytes)} \times 8$$

Data Transfer Time

Estimates the time required to transfer a given amount of data over a network connection.

$$\text{Time (seconds)} = \frac{\text{Data (bits)}}{\text{Effective Bandwidth (bps)}}$$

Data Sharding and Partitioning

Calculates the target shard (or partition) for a given key (K) across a cluster of N nodes using a hash function H.

$$\text{Shard ID} = H(K) \pmod N$$

Cloud Storage Costs

Calculates the standard cost components of storing and serving data in the cloud.

$$\text{Storage Cost} = \text{Total Data (GB)} \times \text{Price per GB}$$

$$\text{Request Cost} = \left(\frac{\text{Total Requests}}{\text{Billing Unit}} \right) \times \text{Price per Unit}$$

$$\text{Transfer Cost} = \text{Total Data Out (GB)} \times \text{Price per GB Out}$$

Unit 5: Cloud Security

RSA Cryptography

Core mathematical foundations for the RSA public-key encryption algorithm.

$$n = p \times q$$

$$\phi(n) = (p-1) \times (q-1)$$

$$C = M^e \pmod n \quad \text{(Encryption)}$$

$$M = C^d \pmod n \quad \text{(Decryption)}$$

Quantitative Risk Assessment

Calculates the expected financial loss from security incidents to help justify security investments.

$$\text{SLE} = \text{Asset Value (AV)} \times \text{Exposure Factor (EF)}$$

$$\text{ALE} = \text{SLE} \times \text{Annualized Rate of Occurrence (ARO)}$$

(Where SLE is Single Loss Expectancy and ALE is Annualized Loss Expectancy).

Unit 6: Edge Computing and Serverless

Edge Computing Data Rates

Models the flow of data from IoT edge devices to cloud storage and the bandwidth saved via edge processing.

$$R_{raw} = N \times S \times f$$

$$R_{edge} = R_{raw} \times \text{Filtration Rate}$$

$$B_{cloud} = R_{raw} - R_{edge}$$

$$\Delta B = R_{raw} - B_{cloud} = R_{edge}$$

(Where N is number of devices, S is payload size, and f is transmission frequency).

IoT Data Storage Requirements

Estimates the volume and storage requirements for telemetry data over a given retention period.

$$V_{daily} = N \times \left(\frac{86400}{T_{interval}} \right)$$

$$S_{daily} = V_{daily} \times \text{Message Payload Size}$$

$$S_{total} = S_{daily} \times \text{Retention Period (Days)}$$

(Where V_{daily} is daily message volume, and $T_{interval}$ is the transmission interval in seconds).

Serverless Compute (FaaS) Pricing

Calculates the billing components of serverless function execution.

$$T_{exec} \text{ (in seconds)} = \text{Total Requests} \times \text{Average Duration (seconds)}$$

$$C_{alloc} = T_{exec} \times \text{Allocated Memory (in GB)}$$

$$\text{Compute Cost} = C_{alloc} \times \text{Price per GB-second}$$

$$\text{Request Cost} = \left(\frac{\text{Total Requests}}{1,000,000} \right) \times \text{Price per Million Requests}$$

$$\text{Total Cost} = \text{Compute Cost} + \text{Request Cost}$$