CS5224
AY23/24 SEM 2
francisyzy.com

Three Eras of Computing

The lecture begins by delineating the three eras of computing: the Tabulating era from the 1890s to the 1940s, characterized by mechanical tabulators for counting and summarizing information; the Programming era from the 1950s to the present, marked by programmable systems capable of performing different and multiple tasks; and the Cognitive era from 2011 to the future, focusing on systems that extend human cognition, learn, and adapt over time.

What and Why

Cloud computing is presented as a platform for hosting and delivering cognitive services, with examples including IBM Watson cognitive services over Bluemix, Amazon AI services over AWS, Microsoft AI tools over MS Azure, and Google AI services on Google Cloud. The essence of cloud computing is its ability to provide on-demand network access to a shared pool of configurable computing resources, rapidly provisioned with minimal management effort.

Cost Model

The lecture discusses the cost model of cloud computing, emphasizing its pay-per-use nature, which allows for the systematic evaluation of cloud services by combining compute, disk storage, and memory requirements. This model is demonstrated using a dataset derived from a real-world industrial data center workload.

History

The history of cloud computing is traced back to the 1960s with the concept of "time-sharing," leading to the emergence of cloud computing as a significant technological advancement. The lecture highlights key milestones, including the launch of Amazon Web Services (AWS) in 2002 and the introduction of Elastic Compute Cloud Commercial Service by Amazon in 2006.

Key Business Drivers

Cloud computing's key business drivers include cost optimization, increased competitiveness, and the ability to scale resources elastically to meet changing demands. These drivers underscore the strategic importance of cloud computing for organizations seeking to enhance their operational efficiency and agility.

Basic Concepts and Terminology

The lecture introduces basic concepts and terminology associated with cloud computing, such as elasticity, on-demand self-service, pay-per-use (measured service), multi-tenancy (location-independent resource pooling), cloud service (delivery) models, cloud deployment models, cloud actors, and trust boundary.

Technical and Non-Technical Challenges

Finally, the lecture addresses both technical and non-technical challenges of cloud computing. Technical challenges include data security and privacy, cost management, performance, interoperability, and reliance on network connectivity. Non-technical challenges involve organizational readiness, skill gaps, and the cultural shift required to embrace cloud computing technologies.

NIST Definition of Cloud Computing

The NIST definition of cloud computing is highlighted as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources. These resources can be rapidly provisioned and released with minimal management effort or service provider interaction. The definition encapsulates the essence of cloud computing, emphasizing its five essential characteristics, four deployment models, and three service models.

Essential Characteristics

- **On-demand self-service**: Users can provision computing resources as needed automatically without requiring human interaction with the service provider.
- **Broad network access**: Resources are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms.
- **Resource pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- **Rapid elasticity**: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.
- **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

Service Models

- **Software as a Service (SaaS)**: The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure.
- **Platform as a Service (PaaS)**: The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.
- **Infrastructure as a Service (IaaS)**: The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer can deploy and run arbitrary software, which can include operating systems and applications.

Deployment Models

- **Public Cloud**: The cloud infrastructure is provisioned for open use by the general public.
- **Private Cloud**: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units).
- **Community Cloud**: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns.
- **Hybrid Cloud**: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

Challenges and Considerations

The lecture also addresses challenges in cloud computing, including security, data privacy, compliance, and the importance of understanding these challenges to leverage cloud technologies effectively.

Cloud Computing Characteristics

Cloud computing is defined by five key characteristics:

1. On-demand self-service
2. Broad network access
3. Resource pooling
4. Rapid elasticity
5. Measured service

Service Models

Cloud computing offers three primary service models, each providing different levels of abstraction:

1. Infrastructure as a Service (IaaS) - provides virtualized computing resources over the internet.
2. Platform as a Service (PaaS) - offers hardware and software tools over the internet, typically for application development.
3. Software as a Service (SaaS) - delivers software applications over the internet, on a subscription basis.

Deployment Models

Cloud resources can be deployed in various models, each catering to different requirements of exclusivity and resource sharing:

1. Public Cloud - resources are shared among multiple tenants over the internet.
2. Private Cloud - resources are exclusively used by a single organization.
3. Community Cloud - resources are shared among a group of organizations with common concerns.
4. Hybrid Cloud - combines public and private clouds, allowing data and applications to be shared between them.

Managing Cloud Resources for Elasticity and Scalability

- **Workload Distribution**: Using load balancers to distribute workloads evenly across cloud resources to prevent over- or under-utilization.
- **Resource Pooling**: Aggregating cloud resources to serve diverse needs, with mechanisms like dedicated pools, sibling pools, and nested pools.
- **Dynamic Scalability**: Automatically allocating or deallocating resources based on predefined conditions to respond to fluctuations in demand.
- **Elastic Resource Capacity**: Managing the availability of cloud resources to ensure that consumer demands can be met even when resource availability fluctuates.
- **Service Load Balancing**: Distributing consumer requests across multiple instances of a cloud service to ensure fault tolerance and efficient resource utilization.

How to Organize (Partition) Resources?

1. Workload Distribution

- **Horizontal IT Resource Scaling**: Adding identical IT resources to handle increased workload.
- **Load Balancer**: Distributes workload to prevent over- or under-utilization of resources.
- **Supporting Mechanisms**: Include cloud usage monitor, audit monitor, hypervisor, logical network perimeter, resource cluster, and resource replication.

2. Resource Pooling

- **Aggregation of Cloud Resources**: Different types of resources are pooled to serve diverse needs.
- **Resource Pool Types**: Physical server pool, virtual server pool, CPU pool, memory pool, storage pool, and network pool.
- **Hierarchical Structure**: Dedicated pools, sibling resource pools, and nested resource pools are used to organize resources to meet consumer requirements.

3. Dynamic Scalability

- **Automated Scaling**: Allocating or deallocating resources based on predefined conditions to handle fluctuations in demand.
- **Scalability Mechanisms**: Include automated scaling listener, intelligent automation engine, and hypervisor for dynamic provisioning of virtual resources.

4. Elastic Resource Capacity

- **Managing Resource Availability**: Ensuring that consumer demands can be met even when resource availability fluctuates.
- **Elastic Provisioning System**: Dynamically allocates and reclaims CPUs and RAM for virtual servers in response to processing requirements.

5. Service Load Balancing

- **Distributing Consumer Requests**: Across multiple instances of a cloud service to ensure fault tolerance and efficient resource utilization.
- **Load Balancing Services**: Such as Amazon Elastic Load Balancing (ELB) are used to automatically distribute incoming application traffic across multiple instances.

How to Operate/Manage Resources to Meet Certain Objectives?

Cloud Bursting

- **Handling Excess Demand**: Allows an organization to tap into additional cloud resources when on-premise resources are insufficient.
- **Redundant Copy**: A copy of the application/service is pre-deployed on the cloud and remains inactive until needed.

Elastic Disk Provisioning

- **Dynamic Storage Allocation**: Charges for disk storage based on actual usage rather than allocated storage.
- **Runtime Monitoring**: Monitors disk usage and reports for billing purposes.

The lecture emphasizes the importance of these architectures and mechanisms in enabling cloud providers to offer services that are both elastic and scalable. This ensures that cloud consumers can efficiently manage their resources in line with their fluctuating demands, ultimately leading to cost savings, improved performance, and operational efficiency.

Resource Hosting

- **On-premise vs. Cloud-based Hosting**: The lecture contrasts on-premise hosting, where organizations are responsible for deploying, operating, and maintaining their internet connectivity and quality of service (QoS), with cloud-based hosting. Cloud-based hosting involves using multiple Internet Service Providers (ISPs), which complicates QoS management and necessitates the use of multiple cloud carriers. It highlights the challenges in managing bandwidth and latency, as well as security concerns in a cloud environment.

Datacenter Infrastructure

- **Main Components in a Datacenter**: The lecture outlines the three key components within a datacenter: physical IT resources (servers, storage, and network), building infrastructure (including power and cooling systems), and security measures (fire protection and access control).
- **Server, Storage, and Network**: It discusses the modular architecture of datacenters, emphasizing the use of commodity hardware to facilitate scalability. The lecture also touches on the importance of a deep data storage hierarchy and efficient networking within and across datacenters.
- **Building, Power, and Cooling**: The infrastructure necessary to house physical resources is detailed, including the critical role of uninterruptible power supply systems and cooling mechanisms to maintain optimal operating temperatures for datacenter components.

Datacenter Management

- **Datacenter Tiers and Availability**: Different tiers of datacenters are introduced, categorized based on their availability and redundancy levels. Tier 4 datacenters are highlighted for their high reliability and minimal downtime.
- **Energy Usage and Efficiency**: The lecture addresses the significant energy consumption in datacenters, focusing on the distribution of energy usage among key hardware components. It introduces Power Usage Efficiency (PUE) as a measure of a datacenter's energy efficiency, emphasizing the goal of achieving a PUE as close to 1 as possible.

Innovations and Examples

- The lecture briefly mentions innovative approaches to datacenter design, including Google's multi-story cloud datacenters and Microsoft's undersea datacenter project. It also provides insights into the landscape of datacenters in Singapore, noting the market size, number of datacenters, and the total power capacity.

Summary

The lecture encapsulates the critical aspects of resource hosting and datacenter management within the realm of cloud computing. It underscores the transition from on-premise to cloud-based hosting, the architectural and operational complexities of managing datacenters, and the ongoing efforts to enhance energy efficiency and reduce operational costs in cloud infrastructure.

Availability Table

$$\text{Downtime} = (1 - \text{Availability}) \times \text{Total Time Period}$$

Where:

Availability is the percentage of uptime promised (e.g., 99.9% uptime is represented as 0.999). Total Time Period is the total duration for which the calculation is being made (e.g., a year, a month).

Availability (Number of Nines):	Downtime Per Year
90% (One Nine) :	36.5 Days
95% :	18.25 Days
99% (Two Nines) :	3.65 Days
99.5% :	1.83 Days
99.9% (Three Nines) :	8.76 Hours
99.95% :	4.38 Hours
99.99% (Four Nines) :	52.56 Minutes
99.995% :	26.28 Minutes
99.999% (Five Nines) :	5.26 Minutes
99.9999% (Six Nines) :	31.5 Seconds
99.99999% (Seven Nines) :	3.15 Seconds
99.999999% (Eight Nines) :	315.569 Milliseconds
99.9999999% (Nine Nines) :	31.5569 Milliseconds

PUE

Power Usage Effectiveness (PUE) is a metric used to determine the energy efficiency of a data center. It is calculated by dividing the total amount of energy used by a data center by the energy used by its IT equipment. The formula for calculating PUE is:

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

Here's a step-by-step guide on how to calculate PUE:

1. **Measure Total Facility Energy**: This is the total energy consumption of the data center, including all energy used for cooling, power conversion, lighting, and other support infrastructure. This measurement is typically taken at the utility meter.
2. **Measure IT Equipment Energy**: This is the energy consumed by all IT equipment, which includes servers, storage systems, network equipment, and any other devices directly involved in computing and data management tasks. This measurement is often taken from power distribution units (PDUs) within the data center.
3. **Calculate PUE**: Divide the total facility energy by the IT equipment energy to get the PUE value.

For example, if a data center uses 2,000,000 kWh of total energy and the IT equipment uses 1,600,000 kWh of that energy, the PUE would be calculated as follows:

$$\text{PUE} = \frac{2,000,000 \text{ kWh}}{1,600,000 \text{ kWh}} = 1.25$$

A PUE value of 1.0 is ideal, indicating that all energy consumed is used directly by IT equipment with no additional energy for cooling or other infrastructure. The closer the PUE value is to 1.0, the more efficient the data center is considered to be. Average data centers typically have a PUE value around 1.5 to 2.0, while highly efficient data centers may achieve values closer to 1.2.
It's important to note that PUE should be measured over time to account for variations in energy use due to changes in IT load, weather, and other factors. Regular measurement allows data center managers to track efficiency improvements and identify areas for further optimization.

Singapore PUE/ Datacenter things

For three years, there were no new data centers built in Singapore. Why? It was because data centers consume a lot of power and water for cooling. I think it was in 2022 when they allowed the construction of new data centers, but with a new requirement. The Power Usage Effectiveness (PUE) had to be reduced from 1.58 to 1.3.
The PUE is calculated by dividing the total amount of power used for computing by the total power used for computing plus the supporting facility. A PUE of 1.3 means that there is a 30% power overhead, and the lower the PUE, the better. Data centers are usually not very common, and in Singapore, they are typically multi-storey buildings due to the high cost of land. So, how can you identify a data center building? If you see a building with no windows, it is likely to be a data center, as they are all concealed.

Virtualization

Virtualization is a foundational technology in cloud computing that allows for the creation of multiple simulated environments or resources from a single physical hardware system. This is achieved through software called a hypervisor, which manages and allocates the hardware's resources to virtual machines (VMs). The key benefits of virtualization include:

- **Hardware Independence**: Virtualization abstracts the operating system from the underlying hardware, allowing for greater flexibility in the deployment of resources.
- **Resource Consolidation**: By allowing different virtual resources to share one physical resource, virtualization increases hardware utilization and enables load balancing.
- **Resource Replication**: Virtual resources can be replicated as virtual disk images, facilitating rapid scaling and agility in migration.
- **Horizontal Scaling**: Supports the ability to scale out resources to handle increased load.
- **Business Continuity**: Virtualization supports backup and scaling strategies that contribute to business continuity and disaster recovery.

However, virtualization also introduces a performance overhead and can become a single point of failure if not managed correctly.

Types of Virtualization

The lecture outlines several types of virtualization, including processor, memory, storage, network, and data virtualization, each abstracting their respective physical resources to form a pool of virtual resources.

Multitenancy

Multitenancy is a key concept in cloud computing where a single instance of software serves multiple customer instances. This architecture is essential for sharing Platform as a Service (PaaS) platforms and Software as a Service (SaaS) services while maintaining tenant isolation. The benefits of multitenancy include:

- **Cost Efficiency**: Sharing resources across multiple tenants reduces costs for both the provider and the tenants.
- **Scalability**: It is easier to add new tenants without the need for additional software instances.
- **Tenant Isolation**: Despite sharing resources, each tenant's data and usage are kept isolated, ensuring privacy and compliance with security standards.

Cloud Service Models and Virtualization

The lecture emphasizes how virtualization is utilized across different cloud service models:

- **Infrastructure as a Service (IaaS)**: Uses virtualization for processor, memory, storage, and network resources to provide infrastructure services.
- **Platform as a Service (PaaS)**: Virtualizes the development platform, offering it as a service to developers.
- **Software as a Service (SaaS)**: Allows multiple end-users to share a single instance of centrally hosted software.

Practical Implications

For cloud computing to be effective, it is crucial to understand and implement virtualization and multitenancy correctly. These technologies enable cloud providers to offer scalable, cost-effective, and flexible services to their customers. Businesses looking to leverage cloud computing should consider these aspects when selecting cloud services and providers.

Pseudocode for MapReduce Word Count

[] # The map function processes input key-value pairs and produces intermediate key-value pairs def map(document_name, document_contents): for word in document_contents.split(): emit_intermediate(word, 1)

# The reduce function processes all intermediate values associated with the same intermediate key # and produces a set of merged output values (in this case, the sum of occurrences for each word) def reduce(word, counts): total_count = sum(counts) emit(word, total_count)

# The MapReduce framework would call the map function once for each document, where # 'document_name' is the name of the document, and 'document_contents' is the text of the document. # The framework collects all the outputs of the map function that have the same key (word in this case) # and passes them to the reduce function.

# For example, if the intermediate data produced by map is: # ("hello", 1), ("world", 1), ("hello", 1) # The framework will organize this data as follows before passing it to the reduce function: # ("hello", [1, 1]), ("world", [1])

# Then, the reduce function will be called as follows: # reduce("hello", [1, 1]) -> ("hello", 2) # reduce("world", [1]) -> ("world", 1)

# The final output would be: # ("hello", 2), ("world", 1)

Cloud Applications

Cloud applications are software programs that are designed to be accessed over the internet and hosted in a cloud computing environment. These applications leverage cloud resources to provide various services to users, including data storage, processing power, and application functionality.

Common Features

Cloud applications typically share several common features:

- **Scalability**: They can easily scale resources up or down based on demand.
- **Accessibility**: Users can access cloud applications from anywhere with an internet connection.
- **Multi-tenancy**: Multiple users or organizations can use the same application instance, sharing resources while keeping their data separate.
- **Elasticity**: Cloud applications can automatically allocate and deallocate resources to handle varying workloads efficiently.
- **Pay-per-use**: Users typically pay only for the resources they consume, rather than investing in hardware and software upfront.

Applications

Cloud applications span a wide range of uses, from consumer-oriented services like email and social media to enterprise applications such as customer relationship management (CRM) systems, enterprise resource planning (ERP), and more.

Challenges in Developing Applications

Developing cloud applications comes with its own set of challenges:

- **Performance optimization**: Ensuring that applications run efficiently and effectively in the cloud environment.
- **Scalability issues**: Addressing the potential limitations in data usage and resource allocation.
- **Multiple platform support**: Ensuring compatibility with various cloud environments and providers.
- **Downtime issues**: Minimizing the impact of service interruptions on application availability.
- **App security**: Protecting applications from vulnerabilities and cyber threats.

Architectural Styles for Cloud Applications

Cloud applications can be designed using various architectural styles:

- **Monolithic architecture**: A single-tiered software application where different components are interconnected and interdependent.
- **Microservices architecture**: An approach where an application is composed of small, independent services that communicate over well-defined APIs.
- **Serverless architecture**: A design pattern where the cloud provider dynamically manages the allocation of machine resources.

Cloud Application Development Models

The development of cloud applications can be categorized into three main service models:

- **Infrastructure as a Service (IaaS)**: Provides virtualized computing resources over the internet, offering the highest level of flexibility and management control over IT resources.
- **Platform as a Service (PaaS)**: Offers a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure.
- **Software as a Service (SaaS)**: Delivers software applications over the internet on a subscription basis, with providers managing the infrastructure and platforms.

Characteristics of Cloud Service Models

Each cloud service model has distinct characteristics:

- **IaaS**: Typically includes services like virtual machines, storage, and networking. It's cost-effective and offers a high degree of flexibility.
- **PaaS**: Provides development and deployment environments in the cloud, supporting various programming languages and tools.
- **SaaS**: Users access software applications through web browsers, with the provider handling all aspects of the application, including performance, security, and maintenance.

Examples of Setting up a Blog

IaaS:

1. Creates a compute instance running Linux
2. Installs and configures Apache,MySQL and PHP
3. Installs and configures Wordpress blog engine
4. Configures a new blog
5. Blogs happily about IaS development

PaaS:

1. Accesses a system with Linux, Apache, MySQL and PHP (LAMP)
2. Installs and configures Wordpress blog engine
3. Setup a new blog
4. Blogs happily about PaS provisioning

SaaS:

1. Accesses a server with Wordpress installed and configured
2. Setup a new blog
3. Blogs happily about SaaS development

Function-as-a-Service

Function-as-a-Service (FaaS) is a cloud computing service that allows developers to execute code in response to events without managing servers or runtime environments. It is a key component of serverless architectures.

How a Lambda Function Runs

AWS Lambda is an example of FaaS where functions are triggered by specific events and run in stateless containers that are fully managed by AWS, scaling automatically with the size of the workload.

How Event and Lambda Function Interact

In AWS Lambda, events from various AWS services or HTTP requests can trigger functions. The function runs and processes the event, then returns a response or triggers other processes within AWS.

Running Serverless Application

Serverless applications run on managed services like AWS Lambda, where the cloud provider executes the application code in response to events, scaling automatically and charging only for the compute time consumed.

MapReduce Programming Model

MapReduce is a programming model for processing and generating large datasets with a parallel, distributed algorithm on a cluster. It simplifies the distribution of tasks across multiple nodes and handles failures and inter-node communication.

Example: Word Count

A classic example of a MapReduce application is the word count program, where the map function processes input data to produce key-value pairs, and the reduce function aggregates the results to count the occurrences of each word.

Hadoop

Hadoop is an open-source framework that supports data-intensive distributed applications, implementing the MapReduce programming model. It provides a reliable, scalable platform for storage and processing of big data across clusters of computers.

Map Phase:

[] map(key, value): // key: document id // value: document contents for each document in value: for each cluster center: compute distance of document to cluster center find closest cluster center emit(closest cluster center, document)

Reduce Phase:

[] reduce(key, values): // key: cluster center // values: documents assigned to cluster center for each document in values: add document to cluster update cluster center based on added document emit(cluster center, new cluster center)

The overall K-means MapReduce algorithm:

1. Initialize K cluster centers randomly
2. Repeat until convergence:
   - Map phase: Assign each document to closest cluster center
   - Reduce phase: Recompute cluster centers based on document assignment
   - If cluster centers changed, repeat, else finish

In more detail:

- The map function takes a document as input and computes the distance to each of the K cluster centers
- It emits the document ID and contents, with the key being the closest cluster center
- The reduce function collects all documents assigned to a particular cluster center
- It recomputes the cluster center by averaging the documents
- The new cluster center is emitted
- This repeats until the cluster centers stop changing significantly between iterations

Some key aspects:

- Cluster centers can be stored in a file on HDFS accessible to all mappers
- Mappers load the cluster centers at the start of each iteration
- Reducers emit the updated cluster centers back to HDFS for the next iteration
- A combiner can be used to do partial cluster center updates to reduce data shuffled
- Convergence is checked between iterations by looking at the change in cluster centers

So in summary, the map phase does cluster assignment, the reduce phase does cluster center re-estimation, and this iterates until convergence, leveraging the scalability of MapReduce to parallelize K-means on large datasets. The cluster centers are shared via HDFS between map and reduce tasks.## SaaS is Different from Traditional Software

SaaS changes how software is delivered compared to traditional software:

- Pay-per-use vs. traditional licensing
- Web access to the software, no customer installation
- Multi-tenant (1-to-many) vs dedicated (1-to-1)
- Automated updates by the provider vs. customer managed

Different Perspectives of SaaS Development

There are 4 main perspectives, each with different levels of resource sharing:

1. SaaS with Self-managed Infrastructure and Platform - full control but lower utilization
2. SaaS with Self-managed Platform - shared infrastructure, self-managed platform

3. SaaS with Self-managed Infrastructure - shared platform, self-managed infrastructure for data governance
4. SaaS with Cloud-enabled IaaS and PaaS - fully shared, highest utilization but less control

Two key challenges in the Different Perspectives of SaaS Development are:

1. Choosing the correct multitenancy level(s)

- Multitenancy can be achieved at different levels such as infrastructure, platform and application
- The degree of multitenancy impacts resource utilization and data security
- Higher multitenancy leads to better resource utilization but lower data security
- SaaS developers need to carefully choose the right level of multitenancy to balance these trade-offs

2. Governance and security over user data

- Since SaaS applications are managed by the service provider, ensuring proper governance and security of sensitive customer data is critical
- SaaS developers need to implement strong access controls, encryption, backup and recovery to protect user data
- Compliance with data privacy regulations like GDPR, HIPAA etc. is also important when handling user data

Here are more details on the key considerations when developing cloud-aware SaaS applications using PaaS:

1. Requirement Analysis

- Carefully analyze the delivery model (SaaS) and deployment model (public/private/hybrid cloud)
- Understand the target users, expected workload patterns, data security needs
- Identify functional and non-functional requirements that impact the architecture

2. Multitenant Architecture

- Decide the degree of multitenancy at infrastructure, platform, database and application levels
- Higher multitenancy provides better resource utilization but lower data security and tenant isolation
- Design the application to be multitenant-aware, e.g. tenant context, data partitioning, customizations per tenant

3. Dynamic Scaling and Availability

- Unlike traditional web apps, SaaS apps must be designed upfront for dynamic scaling
- Leverage PaaS capabilities for automated scaling, load balancing, failover
- Use stateless components, distributed caching, eventual consistency for better scalability
- Design for multi-zone high availability

4. Database Design

- Database layer is critical for SaaS multitenancy and scalability
- Evaluate multi-tenancy models - separate database, shared database separate schema, shared schema
- Often a combination is used - e.g. shared tables for metadata, separate tables for tenant data
- NoSQL databases provide better horizontal scalability and availability compared to RDBMS

5. SaaS Development using PaaS

- PaaS provides built-in support for multitenancy, scalability, availability needed for SaaS
- Utilize PaaS services as much as possible - e.g. database, caching, identity management, analytics
- Follow cloud-native development practices - DevOps, microservices, API-first, automation

6. Monitoring and SLA Maintenance

- Robust monitoring is essential in SaaS to maintain SLAs and prevent issues
- Proactively monitor availability, performance, security, compliance of the SaaS application
- Use PaaS monitoring tools as well as custom tools for application-specific metrics
- Have automated alerts and incident response processes to minimize downtime

Achieving multi-tenancy at the database level is an important consideration for successful SaaS development. The multi-tenant database options include:

- Sharing the database instance across tenants
- Sharing database tables across tenants
- Sharing database schemas across tenants

NoSQL databases are a better option than relational databases for SaaS applications to achieve better scalability and availability.

Pricing Models:

- There are two main pricing models for SaaS and PaaS services: integrated and separate.
- In integrated pricing, the consumer pays the SaaS provider who in turn pays the IaaS provider for hosting.
- In separate pricing, the consumer pays the SaaS provider for the service and separately pays the IaaS provider for hosting.

Cost Metrics:

- Business costs include upfront capital expenses (CapEx) and ongoing operational expenses (OpEx). Cloud has low CapEx but potentially higher OpEx.
- Additional business costs to consider are cost of capital, sunk costs, integration costs, and lock-in costs.
- Consumer cloud usage costs include network usage, server usage, storage, and cloud services. These are metered and billed based on usage.

Total Cost of Ownership (TCO):

- TCO for a datacenter includes CapEx (construction, servers) depreciated over time and OpEx (electricity, maintenance, salaries).
- TCO = datacenter depreciation + datacenter OpEx + server depreciation + server OpEx
- The lecture compares TCO scenarios for high-end servers, lower-cost higher-power servers, and 50% datacenter utilization. Server amortization varies from 17-66% of 3-year TCO in the examples.

AWS Pricing:

- AWS offers on-demand, reserved, and spot pricing options to support different types of workloads.
  - On-demand is pay-per-use by the second.
  - Reserved requires an upfront fee for discounted rates.
  - Spot allows bidding for unused capacity at a discount but may terminate if outbid.
  - Dedicated has standard and reserved. Multi-Tenant Single Customer. For Regulatory and Compliant workloads

Key Terms

L01

Elastic Resource Elastic resources refer to the capability of cloud computing environments to dynamically allocate and deallocate resources based on demand. This elasticity allows for the scaling of resources, such as computing power and storage, to match the current needs of applications without manual intervention. Elastic resources are fundamental to cloud computing's value proposition, offering flexibility and cost-efficiency by ensuring that users pay only for the resources they consume.

Availability Availability in cloud computing denotes the degree to which a system, service, or data remains accessible and functional when required by users. High availability is achieved through redundant systems, failover mechanisms, and robust infrastructure design, ensuring minimal downtime and service disruption. This is critical for business continuity and user satisfaction, as it guarantees that cloud services are consistently accessible despite potential failures or maintenance activities.

Capacity Planning (Resource Provisioning) Capacity planning, or resource provisioning, involves estimating the resources required to meet current and future demands of applications or systems within the cloud. This process is essential for maintaining optimal performance and avoiding both over-provisioning, which leads to unnecessary costs, and under-provisioning, which can cause performance issues or service disruptions. Effective capacity planning requires understanding workload characteristics, monitoring resource utilization, and forecasting future needs.

Scaling (Horizontal, Vertical) Scaling in cloud computing can be categorized into horizontal and vertical scaling. Horizontal scaling, or scaling out/in, involves adding or removing instances of resources (e.g., servers) to adjust capacity. This method enhances availability and fault tolerance by distributing workloads across multiple resources. Vertical scaling, or scaling up/down, entails increasing or decreasing the capacity of existing resources, such as upgrading CPU or memory of a server. Choosing between horizontal and vertical scaling depends on the application architecture, cost considerations, and performance requirements.

Cloud-based IT Resources Cloud-based IT resources refer to the computing infrastructure, platforms, and software that are delivered and managed over the internet by cloud service providers. These resources include virtual machines, storage, databases, networking, and applications, accessible on-demand to users. The cloud model enables organizations to utilize IT resources without the need for significant capital investment in physical hardware, offering scalability, flexibility, and cost savings.

Cloud Service A cloud service is a solution offered by cloud providers that delivers computing resources and capabilities over the internet. Cloud services are categorized into three primary models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These models provide varying levels of abstraction and control, from the underlying infrastructure to complete software applications, catering to different user needs and simplifying IT management.

Trust Boundary The trust boundary in cloud computing defines the logical perimeter within which IT resources and data are considered secure and trusted by an organization. It delineates the scope of security controls and policies applied to protect resources from unauthorized access and threats. In cloud environments, the trust boundary extends to include both the organization's and the cloud provider's infrastructure, raising considerations around security, compliance, and data privacy. Managing trust boundaries effectively is crucial for maintaining the integrity and confidentiality of data in the cloud.

L02

1. Elasticity Elasticity in cloud computing refers to the ability of a system to automatically scale computing resources up or down as needed to accommodate workload demands. It allows for the dynamic provisioning of resources to match demand, ensuring that users pay only for the resources they consume. Elasticity is crucial for efficiency and flexibility in cloud services, enabling rapid response to changing requirements.

2. On-demand self-service On-demand self-service is a characteristic of cloud computing that allows users to provision computing resources such as servers and storage as needed automatically without requiring human interaction with the service provider. This feature provides users with immediate access to resources, contributing to the agility and responsiveness of cloud services.

3. Pay-per-use (measured service) Pay-per-use, also known as measured service, is a billing model where users are charged based on their actual consumption of computing resources. This model aligns costs with usage, allowing organizations to optimize their cloud spending by paying only for the resources they use, rather than investing in excess capacity.

4. Multi-tenancy (location independent resource pooling) Multi-tenancy refers to a cloud architecture where multiple customers share the same physical resources, such as servers and storage, but are isolated from each other in terms of data and application instances. This approach maximizes resource utilization and cost-efficiency by pooling resources that can be dynamically allocated to different users based on demand.

5. Cloud service (delivery) models Cloud service models define how cloud services are delivered to users. The three primary models are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Each model offers different levels of abstraction and control over the computing resources, with SaaS providing the highest level of abstraction and IaaS offering the most control.

6. Cloud deployment models Cloud deployment models describe how cloud resources are made available to users. The four main models are public, private, community, and hybrid clouds. Each model offers different levels of resource exclusivity and is chosen based on organizational needs and regulatory requirements.

7. Cloud actors Cloud actors are the entities (individuals or organizations) that participate in cloud computing transactions or processes. The five major actors identified by NIST are cloud consumers, cloud providers, cloud auditors, cloud brokers, and cloud carriers. Each actor plays a specific role in the cloud ecosystem, from using services to managing and delivering them.

8. Trust boundary The trust boundary in cloud computing is a logical perimeter that defines the extent to which an organization trusts the IT resources provided by a cloud service. It encompasses the organizational boundaries of both the cloud provider and the cloud consumer. Trust boundaries are crucial for understanding security vulnerabilities and operational governance control in a multi-tenant cloud environment.

L03

1. Dynamic Scalability (Service Demand and Cloud Resource Fluctuations) Dynamic scalability refers to the cloud computing capability to automatically adjust computing resources based on the current demand without human intervention. This ensures that applications can handle increases or decreases in workload efficiently. Dynamic scalability can be achieved through two main strategies:

- **Horizontal Scaling (Scaling Out/In)**: Involves adding or removing instances of resources (such as virtual machines) to match demand. This strategy is effective for handling sudden spikes in traffic by distributing the load across multiple instances.
- **Vertical Scaling (Scaling Up/Down)**: Involves adding more power (CPU, RAM) to an existing instance rather than adding more instances. This is useful for applications that cannot be easily distributed across multiple servers.

2. Workload Distribution Workload distribution is the process of distributing incoming tasks or network traffic across a group of backend resources, such as servers, to optimize resource use, maximize throughput, minimize response time, and avoid overloading any single resource. Effective workload distribution relies on strategies like load balancing to ensure that no single server bears too much load, which can lead to performance degradation or service outages.

3. Load Balancer A load balancer is a device or software that acts as a reverse proxy and distributes network or application traffic across multiple servers. Load balancers are used to increase the capacity (concurrent users) and reliability of applications. They improve the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, or central processing units. Load balancers can be hardware-based or software-defined and are essential for achieving high availability and reliability by distributing the load efficiently.

4. Resource Pooling Resource pooling in cloud computing refers to the provider's ability to serve multiple consumers with provisioned resources that can be dynamically assigned and reassigned according to consumer demand. The resources in the pool, such as storage, processing, memory, and network bandwidth, are abstracted from the consumers who do not need to know the exact location of the provided resources. Resource pooling allows for cost-efficiency and scalability, as resources can be allocated based on demand in real-time.

5. Cloud Bursting Cloud bursting is a configuration set up between a private cloud and a public cloud to deal with peaks in IT demand. When an organization using a private cloud reaches its resource capacity, the overflow traffic is directed to a public cloud so there's no interruption of services. Cloud bursting is beneficial for handling unexpected surges in demand, providing a cost-effective solution for scaling resources. It allows organizations to maintain their data and applications on a private cloud while taking advantage of the scalable, on-demand resources of a public cloud during peak times.

L04

1. On-premise (vs cloud-based) **On-premise** refers to the deployment and hosting of resources such as servers, storage, and networking hardware within an organization's physical premises. This model allows organizations to have full control over their IT infrastructure, including the physical security of the hardware, the configuration of software and systems, and the management of data. On-premise solutions require significant capital investment in hardware and infrastructure, as well as ongoing costs for maintenance, power, and cooling. Organizations are responsible for ensuring the security, compliance, and availability of their on-premise systems.
In contrast, **cloud-based** hosting involves utilizing computing resources (servers, storage, networking) provided by a third-party service provider over the internet. Cloud services are typically offered on a pay-as-you-go or subscription basis, reducing the need for large upfront capital expenditures. Cloud computing offers scalability, allowing organizations to easily adjust their resource usage based on demand. It also shifts the responsibility for maintaining and upgrading hardware and software to the cloud service provider. However, organizations have less control over the physical security and configuration of cloud-based resources compared to on-premise solutions.

2. Datacenter tiers Datacenter tiers are a standardized classification system used to describe the infrastructure and operational capabilities of a data center. The tier system, developed by the Uptime Institute, ranges from Tier I to Tier IV. Each tier represents a level of redundancy, fault tolerance, and uptime.

- **Tier I**: The simplest form of a data center with no redundancy. It has an expected uptime of 99.671% per year, translating to about 28.8 hours of downtime annually.
- **Tier II**: Offers partial redundancy in power and cooling systems. It has an expected uptime of 99.741% per year, or about 22 hours of downtime annually.
- **Tier III**: Features multiple power and cooling distribution paths, with one path active and others on standby, allowing for maintenance without downtime. It has an expected uptime of 99.982% per year, or about 1.6 hours of downtime annually.
- **Tier IV**: Provides full fault tolerance with 2N+1 redundancy for all components, ensuring no single point of failure. It has an expected uptime of 99.995% per year, or about 26.3 minutes of downtime annually.

3. PUE (Power Usage Effectiveness) PUE is a metric used to measure the energy efficiency of a data center. It is defined as the ratio of the total amount of energy used by a data center to the energy delivered to computing equipment. A PUE value of 1.0 indicates perfect efficiency, where all the energy is used by IT equipment with no additional energy for cooling, lighting, or other overhead. Lower PUE values indicate higher energy efficiency. The average PUE for data centers has been improving over the years, but achieving a PUE of 1.0 is challenging and requires innovative cooling and power management strategies.

4. Energy proportional systems Energy proportional systems refer to computing systems and components that consume power in proportion to their workload. In an ideal energy-proportional system, the energy usage would scale linearly with the workload, meaning that a system doing no work would consume no power, and a system at full capacity would consume maximum power. This concept is crucial for improving the energy efficiency of data centers, as it minimizes wasted energy during periods of low demand. Achieving energy proportionality requires advancements in hardware design and power management techniques. It is a key factor in reducing the overall PUE of data centers and making them more sustainable.

L05

Virtualization Virtualization refers to the process of creating a virtual version of something, such as a hardware platform, operating system, storage device, or network resources. It involves using software to simulate the existence of hardware and create a virtual system that can execute programs as if they were running on an actual, physical machine. This allows for the abstraction of physical resources, making it possible to run multiple operating systems and applications on a single physical machine, thereby improving resource utilization and flexibility.

Virtual Machine (VM) A virtual machine (VM) is a software-based emulation of a computer system that provides the functionality of a physical computer. It runs an operating system and applications, making it possible to perform tasks as if they were being executed on an actual hardware machine. VMs are created and managed by hypervisors and can operate independently, with their own virtual hardware resources (such as CPU, memory, and storage), which are mapped to the physical resources of the host machine.

Hypervisor A hypervisor, also known as a virtual machine monitor (VMM), is a layer of software, firmware, or hardware that creates and runs virtual machines. It sits between the hardware and the virtual machines and is responsible for managing the system's resources so that each VM operates efficiently and remains isolated from other VMs. Hypervisors enable multiple operating systems to share a single hardware host, with each OS appearing to have the host's processor, memory, and other resources all to itself.

Container A container is a lightweight, executable package that includes everything needed to run a piece of software, including the code, runtime, system tools, libraries, and settings. Containers are isolated from each other and the host system, but they share the host system's kernel, making them more efficient than virtual machines in terms of resource usage. Containers provide a consistent environment for software to run, regardless of where the container is deployed, making them ideal for microservices and cloud-native applications.

Multitenancy Multitenancy is an architecture where a single instance of software serves multiple users, or "tenants." In a multitenant cloud environment, multiple customers can use the same cloud resources, such as software or infrastructure, but their data and configurations are kept separate. This allows for cost savings and efficiency because resources can be shared without compromising security or privacy. Multitenancy is a common feature in cloud computing services, such as Software as a Service (SaaS) platforms, where different customers access the same application but see only their own data.

L06

1. Divisible Workload

A divisible workload refers to tasks or computational jobs that can be broken down into smaller, independent units of work. These smaller units can then be processed in parallel across multiple computing resources. This concept is crucial in cloud computing and distributed systems, where dividing a large task into smaller chunks enables efficient utilization of available resources, leading to faster processing times and improved scalability. Divisible workloads are particularly relevant in scenarios involving large datasets or computationally intensive tasks, such as data analysis, scientific simulations, and image processing.

2. Performance Isolation

Performance isolation in cloud computing environments ensures that the performance of one tenant's applications does not adversely affect the performance of another tenant's applications, despite sharing underlying physical resources. This is achieved through various techniques, including resource allocation policies, virtualization, and containerization, which create logical boundaries between tenants' environments. Performance isolation is critical in multi-tenant architectures to maintain service quality and prevent "noisy neighbor" issues, where the resource-intensive activities of one tenant can degrade the performance for others.

3. Web Application Architecture Layers

Web application architecture layers refer to the structured approach of dividing a web application into separate, interconnected components, each responsible for a specific aspect of the application's functionality. The typical layers include:

- **Presentation Layer**: The user interface and user experience components, including HTML, CSS, and JavaScript, responsible for displaying information to the user and capturing user inputs.
- **Application Layer (or Business Logic Layer)**: Contains the core functionality of the application, processing user inputs, executing business rules, and making logical decisions.
- **Data Layer**: Manages data storage and retrieval, interacting with databases or other data storage solutions to save and fetch application data.

This layered architecture promotes separation of concerns, making applications more manageable, scalable, and maintainable.

4. Application Development Models (IaaS, PaaS, SaaS)

- **Infrastructure as a Service (IaaS)**: Provides virtualized computing resources over the internet. Users have control over operating systems, storage, and deployed applications but do not manage the underlying cloud infrastructure.
- **Platform as a Service (PaaS)**: Offers a development and deployment environment in the cloud, with resources enabling users to develop, run, and manage applications without dealing with the underlying infrastructure.
- **Software as a Service (SaaS)**: Delivers software applications over the internet, on a subscription basis. SaaS providers manage the infrastructure, platforms, and software, while users access the software from web browsers.

5. Function-as-a-Service and Serverless FaaS

Function-as-a-Service (FaaS) is a cloud computing service that allows developers to execute code in response to events without managing servers or runtime environments. FaaS is a key component of serverless architectures, where the cloud provider dynamically manages the allocation of machine resources. Serverless computing encompasses more than just FaaS, including backend services (BaaS) and other cloud services where infrastructure management is abstracted away from the developer. FaaS is particularly suited for microservices-based applications and event-driven processing.

6. MapReduce & Hadoop

MapReduce is a programming model and an associated implementation for processing and generating large datasets with a parallel, distributed algorithm on a cluster. It simplifies the process of distributing tasks across multiple nodes, handling failures, and managing inter-node communication. Hadoop is an open-source framework that supports data-intensive distributed applications, implementing the MapReduce programming model. It provides a reliable, scalable platform for storage and processing of big data across clusters of computers using simple programming models.

L09

Costs (up-front, on-going)

Costs associated with acquiring and using a product or service can be categorized into up-front and ongoing costs.

1. **Up-front costs** refer to the initial expenses required to acquire a product or service. These costs are incurred before the product or service is put into use and can include purchase price, installation fees, and any other one-time expenses necessary for setup. Up-front costs are typically capital expenses (CapEx) that are incurred once and may be amortized over the life of the product or service.
2. **Ongoing costs** are expenses that continue to be incurred as long as the product or service is in use. These can include operational expenses (OpEx) such as maintenance fees, subscription costs, utility charges, and any other recurring payments necessary for the continued use and upkeep of the product or service. Ongoing costs represent the long-term financial commitment to a product or service beyond the initial purchase.

Cloud Usage Costs

Cloud usage costs are the expenses associated with using cloud computing services. These costs can vary widely depending on the type of cloud service model employed (IaaS, PaaS, SaaS), the resources consumed (e.g., compute power, storage, bandwidth), and the pricing model of the cloud provider. Key components of cloud usage costs include:

1. **Network usage** costs, which encompass data transfer fees for inbound, outbound, and intra-cloud network traffic.
2. **Server usage** costs, related to the allocation and use of virtual machines or compute instances, often charged on a pay-per-use basis.
3. **Storage costs**, which are incurred for data storage and I/O operations within the cloud.
4. **Service costs**, which may include fees for using specific cloud-based applications, platforms, or additional services such as databases, monitoring tools, and more.

Pricing Models (Integrated, Separate)

Pricing models in cloud computing, particularly for SaaS and PaaS services, can be categorized into integrated and separate models:

1. **Integrated pricing** involves a single payment that covers both the use of the SaaS or PaaS service and the underlying IaaS resources required to host it. In this model, the consumer pays the SaaS or PaaS provider, who in turn pays the IaaS provider for the hosting services. This model simplifies billing for the consumer but may offer less transparency regarding the costs of individual components.
2. **Separate pricing** requires the consumer to pay separately for the SaaS or PaaS service and the IaaS hosting. This model offers greater visibility into the costs associated with each component and may allow for more flexibility in selecting and optimizing IaaS resources. However, it can also introduce complexity in billing and management.

Total Cost of Ownership (TCO)

Total Cost of Ownership (TCO) is a comprehensive assessment of all costs associated with acquiring, deploying, using, and eventually retiring a product or service over its entire lifecycle. TCO includes both direct costs (such as purchase price and maintenance fees) and indirect costs (such as administrative overhead, training, and opportunity costs). The goal of TCO analysis is to provide a complete picture of the financial impact of a purchasing decision, enabling organizations to make more informed choices that consider long-term expenses beyond the initial acquisition cost.
TCO analysis is particularly useful in cloud computing to compare the costs of on-premises infrastructure versus cloud services, taking into account factors such as scalability, flexibility, and the shifting of CapEx to OpEx. By understanding the TCO, organizations can better assess the true value and financial implications of different computing models and services.