Unit - 3

CLOUD PLATFORM ARCHITECTURE

Cloud Computing Overview:

Definition: Cloud computing is the delivery of computing services (like servers, storage, databases, networking, software, analytics, and intelligence) over the internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale.

Benefits :

1. Cost efficiency (pay-as-you-go model)
2. Scalability (easily scale up or down based on demand)
3. Performance and speed (access to the latest technologies)
4. Security (robust security measures from providers)

Service Models: Cloud services are typically categorized into three primary models:

1. **Infrastructure as a Service (IaaS)** : It Provides virtualized computing resources over the internet.

   Examples : Amazon EC2, Microsoft Azure, Google Compute Engine.

   Use Cases: Hosting websites, storage, backup, and recovery, and development/testing environments.

2. **Platform as a Service (PaaS) :** Offers a platform allowing customers to develop, run, and manage applications without dealing with the complexity of building and maintaining the infrastructure.

   Examples : Google App Engine, Heroku, Microsoft Azure App Service

   Use Cases : Application development, middleware services, and API management.

3. **Software as a Service (SaaS) :** It Delivers software applications over the internet, on a subscription basis, eliminating the need for installation and maintenance.

   Examples : Google Workspace, Salesforce, Microsoft 365.

Use Cases : Email services, customer relationship management (CRM), and collaboration tools.

Additional Models:

Function as a Service (FaaS) : A serverless computing model where users run code in response to events without managing servers (e.g., AWS Lambda).

Container as a Service (CaaS) : Offers container-based virtualization to manage and deploy applications (e.g., Google Kubernetes Engine).

Deployment Models:

1. Public Cloud : Services offered over the public internet, available to anyone (e.g., AWS, Google Cloud).

2. Private Cloud : Cloud infrastructure dedicated to a single organization, offering more control and security.

3. Hybrid Cloud : Combines public and private clouds, allowing data and applications to be shared between them.

4. Multi-Cloud : Uses services from multiple cloud providers to avoid dependency on a single vendor.

Understanding cloud computing and its service models is essential for leveraging technology effectively in business and personal projects. Each model has distinct advantages tailored to different needs.

ARCHITECTURAL DESIGN OF COMPUTE AND STORAGE CLOUDS: Architecture of cloud computing is the combination of both SOA (Service Oriented Architecture) and EDA (Event Driven Architecture). Client infrastructure, application, service, runtime cloud, storage, infrastructure, management and security all these are the components of cloud computing architecture.

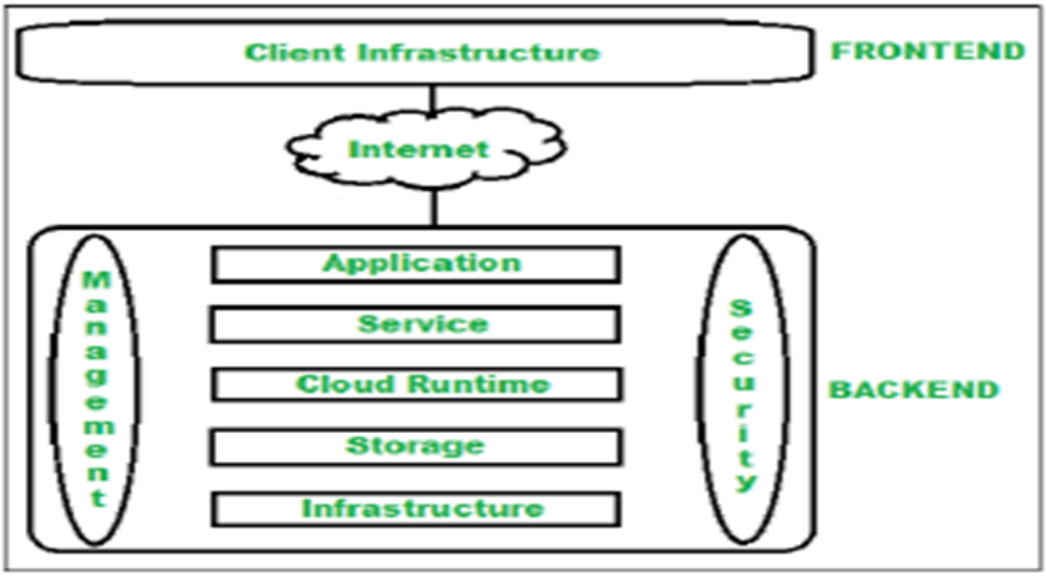
The cloud architecture is divided into 2 parts, i.e.

1. Frontend
2. Backend

Fig: The below figure represents an internal architectural view of cloud computing.

1. Frontend: Frontend of the cloud architecture refers to the client side of cloud computing system. Means it contains all the user interfaces and applications which are used by the client to access the

cloud computing services/resources. For example, use of a web browser to access the cloud platform.



2. Backend: Backend refers to the cloud itself which is used by the service provider. It contains the resources as well as manages the resources and provides security mechanisms. Along with this, it includes huge storage, virtual applications, virtual machines, traffic control mechanisms, deployment models, etc.

Components of Cloud Computing Architecture

Following are the components of Cloud Computing Architecture

1. **Client Infrastructure –** Client Infrastructure is a part of the frontend component. It contains the applications and user interfaces which are required to access the cloud platform. In other words, it provides a GUI(Graphical User Interface) to interact with the cloud.
2. **Application** : Application is a part of a backend component that refers to a software or platform to which a client accesses. Means it provides the service in the backend as per the client requirement.
3. **Service**: Service in backend refers to the major three types of cloud-based services like SaaS, PaaS and IaaS. Also manages which type of service the user accesses.
4. **Runtime Cloud**: Runtime cloud in backend provides the execution and Runtime platform/environment to the Virtual machine.
5. **Storage:** Storage in the backend provides flexible and scalable storage service and management of stored data.

6. **Infrastructure:** Cloud Infrastructure in backend refers to the hardware and software components of cloud like it includes servers, storage, network devices, virtualization software etc.
7. **Management:** Management in backend refers to management of backend components like application, service, runtime cloud, storage, infrastructure, and other security mechanisms etc.
8. **Security:** Security in backend refers to implementation of different security mechanisms in the backend for secure cloud resources, systems, files, and infrastructure to end-users.
9. **Internet:** Internet connection acts as the medium or a bridge between frontend and backend and establishes the interaction and communication between frontend and backend.
10. **Database:** Database in backend refers to a database for storing structured data, such as SQL and NOSQL databases. Examples of Databases services include Amazon RDS, Microsoft Azure SQL database and Google Cloud SQL.
11. **Networking:** Networking in backend services that provide networking infrastructure for application in the cloud, such as load balancing, DNS and virtual private networks.
12. **Analytics:** Analytics in backend service that provides analytics capabilities for data in the cloud, such as warehousing, business intelligence and machine learning.

Benefits of Cloud Computing Architecture:

1. Makes the overall cloud computing system simpler.
2. Improves data processing requirements.
3. Helps in providing high security.
4. Makes it more modularized.
5. Results in better disaster recovery.
6. Gives good user accessibility.
7. Reduces IT operating costs.
8. Provides high level reliability.
9. Scalability.

A Generic Cloud Architecture Design:

Cloud architecture is intended to process massive amounts of data with a high degree of parallelism.

The following are the major design goals of cloud architecture.

1. Scalability – cloud can be easily expanded by adding more servers and enlarging the network connectivity accordingly.
2. Virtualization - The cloud management software needs to support both physical and virtual machines.
3. Efficiency - The hardware and software systems are combined to make it easy and efficient to operate.

4. Reliability - Data can be put into multiple locations. In such a situation, even if one of the data centers crashes, the user data is still accessible.

Security in shared resources and shared access of data centers also pose another design challenge. Clouds support Web 2.0 applications. Cloud management receives the user request, finds the correct resources, and then calls the provisioning services which invoke the resources in the cloud.

Enabling Technologies for Clouds:

The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software.

Cloud users are able to demand more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity, whereas service providers can increase system utilization via multiplexing, virtualization, and dynamic resource provisioning.

Cloud-Enabling Technologies in Hardware, Software, and Networking:

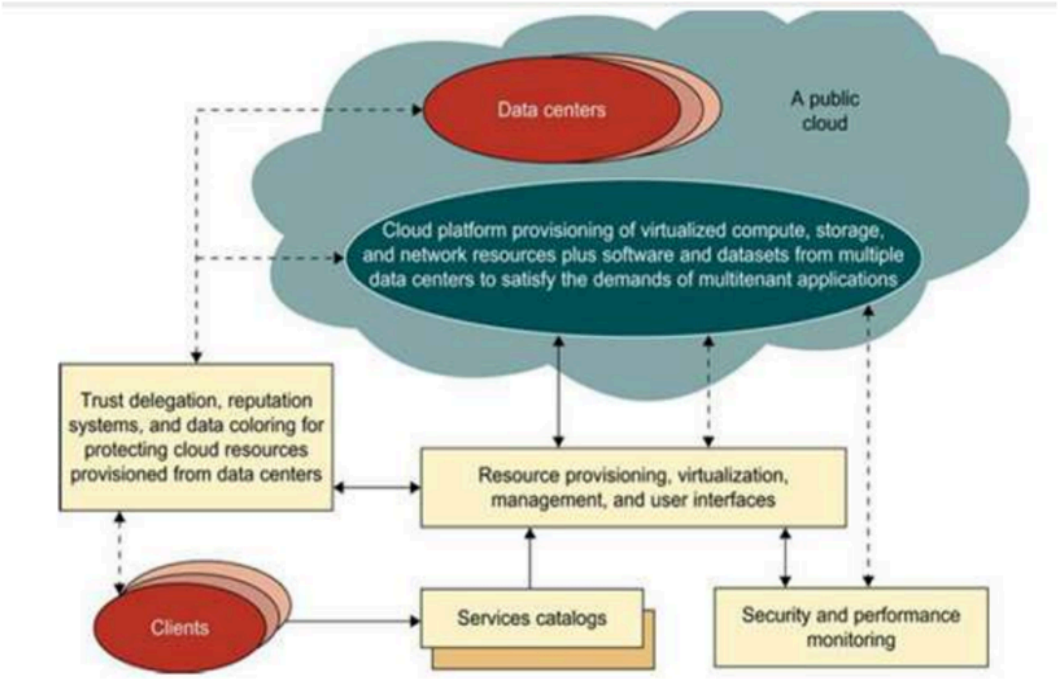
1. Fast platform Deployment – flexible deployment of cloud resources
2. Virtual clusters on demand – to be provided and cluster need to be reconfigured as workload changes.
3. Multitenant techniques –refers distributing software to a large no. of users simultaneously
4. Massive data processing –internet search and web services required this process
5. Web-scale communication – support for e-commerce, education, social networking, medical care etc.
6. Distributed storage –large scale storage of data in distributed storage
7. Licensing and billing services

Rapid progress in multicore CPUs, memory chips, and disk arrays has made it possible to build faster data centers with huge amounts of storage space. Resource virtualization enables rapid cloud deployment and disaster recovery. Service-oriented architecture (SOA) also plays a vital role.

Today's clouds are designed to serve a large number of tenants over massive volumes of data. The availability of large-scale, distributed storage systems is the foundation of today's data centers. Of course, cloud computing is greatly benefited by the progress made in license management and automatic billing techniques in recent years.

A Generic Cloud Architecture: Figure shows security-aware cloud architecture. The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources. The cloud platform is formed dynamically by provisioning or deprovisioning servers, software, and database resources. Servers in the cloud can be physical machines or VMs. In addition to building the server cluster, the cloud platform demands distributed storage and accompanying services.

The cloud computing resources are built into the data centers, which are typically owned and operated by a third-party provider.

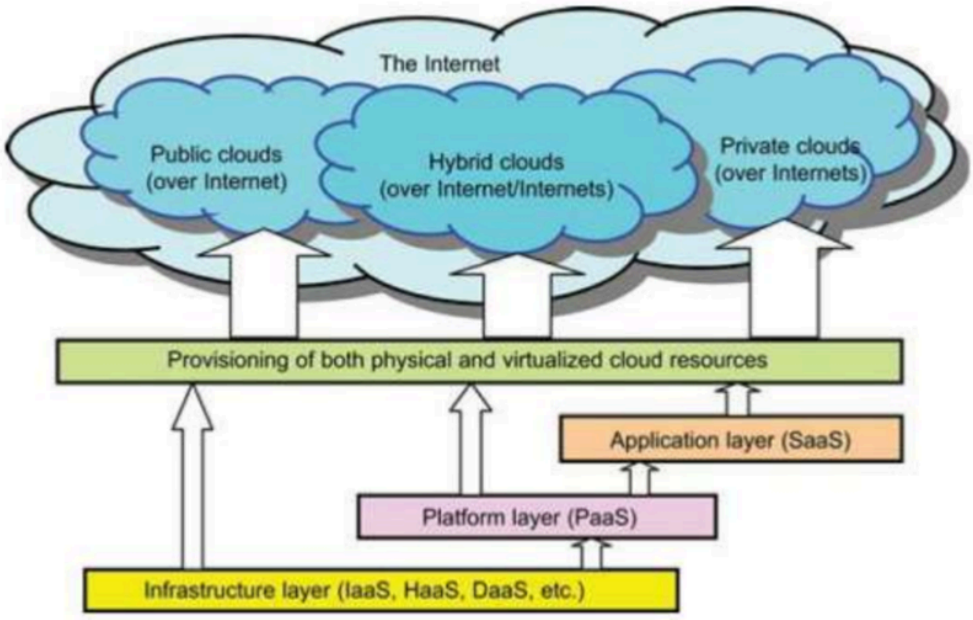


We need to build a framework to process large-scale data stored in the storage system. This demands a distributed file system over the database system. Other cloud resources are added into a cloud platform, including storage area networks (SANs), database systems, firewalls, and security devices. Web service providers offer special APIs that enable developers to exploit Internet clouds. Monitoring and metering units are used to track the usage and performance of provisioned resources.

The software infrastructure of a cloud platform must handle all resource management and do most of the maintenance automatically. So, VMware must detect the status of each node server joining and leaving, and perform relevant tasks accordingly. In general, private clouds are easier to manage, and public clouds are easier to access. The trends in cloud development are that more and more clouds will be hybrid.

Layered Cloud Architecture Development: The architecture of a cloud is developed at three layers: infrastructure, platform, and application, as demonstrated in figure. These three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud. The services to public, private, and hybrid clouds are conveyed to users through networking support over the Internet and intranets involved. It is clear that the infrastructure layer is deployed first to support IaaS services. This infrastructure layer serves as the foundation for building the platform layer of the cloud for supporting PaaS services. In

turn, the platform layer is a foundation for implementing the application layer for SaaS applications. Different types of cloud services demand application of these resources separately.

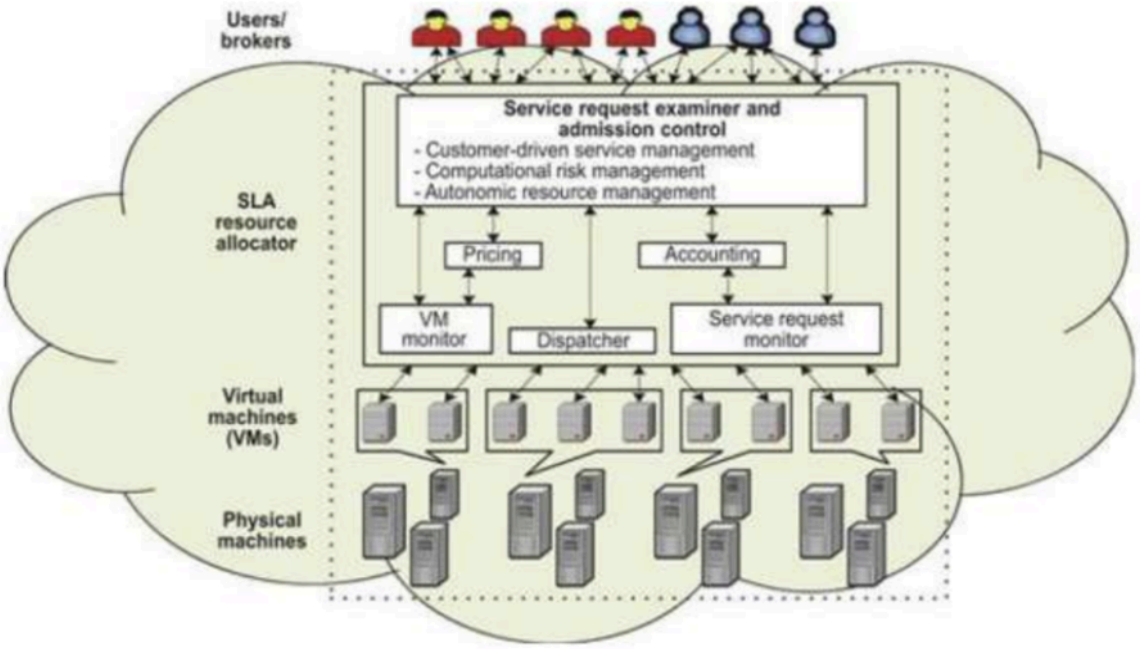


The infrastructure layer is built with virtualized compute, storage, and network resources. The platform layer is for general-purpose and repeated usage of the collection of software resources. This layer provides users with an environment to develop their applications, to test operation flows, and to monitor execution results and performance. Virtualized cloud platform serves as a "system middleware" between the infrastructure and application layers of the cloud. The application layer is formed with a collection of all needed software modules for SaaS applications. Service applications in this layer include daily office management work, such as information retrieval, document processing, and calendar and authentication services. The application layer is also heavily used by enterprises in business marketing and sales, consumer relationship management (CRM), financial transactions, and supply chain management. In general, SaaS demands the most work from the provider, PaaS is in the middle, and IaaS demands the least.

Market-Oriented Cloud Architecture:

Cloud providers consider and meet the different QoS parameters of each individual consumer as negotiated in specific SLAs. Market-oriented resource management is necessary to regulate the supply and demand of cloud resources. The designer needs to provide feedback on economic incentives for both consumers and providers. The purpose is to promote QoS-based resource allocation mechanisms. Figure shows the high-level architecture for supporting market-oriented resource allocation in a cloud computing environment. Users or brokers submit service requests to the data center and cloud to be processed. The SLA resource allocator acts as the interface between the data center/cloud service provider and external

users/brokers. When a service request is first submitted the service request examiner interprets the request for QoS requirements before determining whether to accept or reject the request.



The request examiner ensures that there is no overloading of resources, after that it assigns requests to VMs and determines resource entitlements for allocated VMs. The Pricing mechanism decides how service requests are charged. Pricing serves as a basis for managing the supply and demand of computing resources within the data center and facilitates in prioritizing resource allocations effectively. The Accounting mechanism maintains the actual usage of resources by requests so that the final cost can be computed and charged to users. In addition, the maintained historical usage information can be utilized by the Service Request Examiner and Admission Control mechanism to improve resource allocation decisions.

The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements. The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs. The Service Request Monitor mechanism keeps track of the execution progress of service requests. Multiple VMs can be started and stopped on demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service request. Quality of Service Factors There are critical QoS parameters to consider in a service request, such as time, cost, reliability, and trust/security. QoS requirements cannot be static and may change over time due to continuing changes in business operations. Negotiation mechanisms are needed to respond to alternate offers protocol for establishing SLAs.

Commercial cloud offerings must be able to support customer-driven service management based on customer profiles and requested service requirements. Commercial clouds define computational risk management tactics to identify, assess, and manage risks involved in the execution of applications with regard to service requirements and customer needs. The system incorporates autonomic resource management models that effectively self-manage changes in service requirements to satisfy both new service demands and existing service obligations, and leverage VM technology to dynamically assign resource shares according to service requirements.

Architectural Design Challenges: We will identify six open challenges in cloud architecture development.

1. **Service Availability and Data Lock-in Problem:**

The management of a cloud service by a single company is the source of single points of failure. Multiple cloud providers are to be used to achieve HA. Even if a company has multiple data centers located in different geographic regions, it may have common software infrastructure and accounting systems. Therefore, using multiple cloud providers may provide more protection from failures.

Another availability obstacle is distributed denial of service (DDoS) attacks which make services unavailable to intended users. Some utility computing services offer SaaS providers the opportunity to defend against DDoS attacks by using quick scale-ups. Software stacks have improved interoperability among different cloud platforms, but the APIs itself are still proprietary. The obvious solution is to standardize the APIs so that a SaaS developer can deploy services and data across multiple cloud providers.

This will rescue the loss of all data due to the failure of a single company. In addition to mitigating data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in both public and private clouds. Such an option could enable "surge computing," in which the public cloud is used to capture the extra tasks that cannot be easily run in the data center of a private cloud.

2. **Data Privacy and Security Concerns:**

Current cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks. Many obstacles can be overcome immediately with well understood technologies such as encrypted storage, virtual LANs, and network middleboxes (e.g., firewalls, packet filters). Many nations have laws requiring SaaS providers to keep customer data and copyrighted material within national boundaries.

Traditional network attacks include buffer overflows, DoS attacks, spyware, malware, rootkits, Trojan horses, and worms. In a cloud environment, newer attacks may result from hypervisor malware, guest hopping and hijacking, or VM rootkits. Another type of attack is the man-in-the-middle attack for VM migrations. In general, passive attacks steal sensitive data or passwords. Active attacks may manipulate kernel data structures which will cause major damage to cloud servers.

3. **Unpredictable Performance and Bottlenecks:**

Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic. It is required to improve I/O architectures and operating systems to efficiently virtualize interrupts and I/O channels. Internet applications continue to become more data-intensive. If we assume applications to be "pulled apart" across the boundaries of clouds, this may complicate data placement and transport. Data transfer bottlenecks must be removed, bottleneck links must be widened, and weak servers should be removed for minimizing the cost.

4. **Distributed Storage and Widespread Software Bugs:**

The database is always growing in cloud applications. The opportunity is to create a storage system that will not only meet this growth, but also combine it with the cloud advantage of scaling arbitrarily up and down on demand. This demands the design of efficient distributed SANs. Data centers must meet programmers' expectations in terms of scalability, data durability, and HA. Data consistency checking in SAN-connected data centers is a major challenge in cloud computing.

Large-scale distributed bugs cannot be reproduced, so the debugging must occur at a scale in the production data centers. No data center will provide such a convenience. One solution may be a reliance on using VMs in cloud computing. The level of virtualization may make it possible to capture valuable information in ways that are impossible without using VMs. Debugging over simulators is another approach to attacking the problem, if the simulator is well designed.

5. **Cloud Scalability, Interoperability, and Standardization:**

GAE automatically scales in response to load increases and decreases; users are charged by the cycles used. AWS charges by the hour for the number of VM instances used, even if the machine is idle. In order to save the money, scale up and down must happen quickly. Open Virtualization Format (OVF) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of VMs.

6. **Software Licensing and Reputation Sharing:**

Many cloud computing providers originally relied on open-source software because the licensing model for commercial software is not ideal for utility computing. An opportunity would be to create reputation-guarding services similar to the "trusted email" services currently offered (for a fee) to services hosted on smaller ISPs. Cloud providers want legal liability to remain with the customer, and vice versa. This problem must be solved at the SLA level.

Public Cloud Platforms:

Definition : Public cloud platforms provide cloud services (computing, storage, networking, etc.) over the internet to multiple customers, known as tenants. These services are owned and operated by third-party cloud service providers.

Key Characteristics:

1. Multi-Tenancy : Resources are shared among multiple customers while maintaining data isolation and security.
2. Scalability : Users can quickly scale resources up or down based on demand without significant infrastructure changes.
3. Cost-Effective : Users typically pay only for what they use (pay-as-you-go model), reducing upfront investment costs.
4. Accessibility : Services are accessible from anywhere with an internet connection, facilitating remote work and collaboration.
5. Automatic Updates : Providers manage maintenance and updates, ensuring users have access to the latest features and security enhancements.

Major Public Cloud Providers:

1. Amazon Web Services (AWS) :

Services : Compute (EC2), Storage (S3), Databases (RDS), Networking (VPC), and more.

Strengths : Extensive service offerings, global reach, strong ecosystem, and robust security.

2. Microsoft Azure :

Services : Compute (Azure VM), Storage (Azure Blob), Databases (Azure SQL Database), AI services, and more.

Strengths : Integration with Microsoft products, hybrid cloud capabilities, and a growing range of services.

3. Google Cloud Platform (GCP) :

Services : Compute (Google Compute Engine), Storage (Google Cloud Storage), Databases (Big Query), and AI/ML tools.

Strengths : Advanced data analytics, machine learning capabilities, and a strong focus on open-source technologies.

4. IBM Cloud :

Services : Infrastructure (VMs), Containers (Kubernetes), AI (Watson), and Blockchain.

Strengths : Strong enterprise focus, hybrid cloud offerings, and robust AI tools.

5. Oracle Cloud :

Services : Infrastructure (Compute), Storage, Database (Oracle Autonomous Database), and SaaS applications.

Strengths : Strong database services, enterprise applications, and hybrid cloud capabilities.

Use Cases :

1. Web Hosting : Hosting websites and applications with scalable infrastructure.
2. Data Backup and Recovery : Storing backups in the cloud for disaster recovery.
3. Development and Testing : Quickly provisioning resources for software development and testing environments.
4. Big Data and Analytics : Analyzing large datasets using cloud-native tools and services.
5. Machine Learning and AI : Leveraging cloud services for training and deploying machine learning models.

Considerations:

Security and Compliance : Understand the shared responsibility model and ensure compliance with industry regulations.

Performance : Evaluate latency and throughput requirements, especially for global applications.

Vendor Lock-In : Consider the implications of being tied to a single provider and explore multi-cloud strategies.

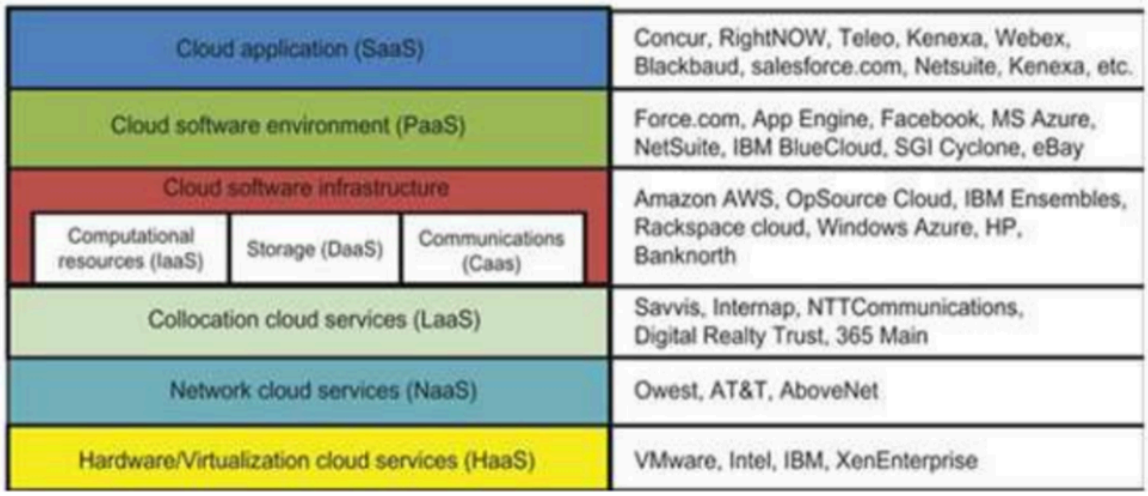
Cost Management : Monitor usage and optimize resources to avoid unexpected costs.

Public cloud platforms provide a flexible, scalable, and cost-effective solution for various computing needs. By leveraging the strengths of major providers, organizations can enhance their operations, drive innovation, and respond quickly to market demands.

Inter-cloud Resource Management

Cloud resource management and intercloud resource exchange schemes are reviewed in this section.

Extended Cloud Computing Services: Figure shows six layers of cloud services, ranging from hardware, network, and collocation to infrastructure, platform, and software applications. The cloud platform provides PaaS, which sits on top of the IaaS infrastructure. The top layer offers SaaS. These must be implemented on the cloud platforms provided.



The cloud players are divided into three classes: (1) cloud service providers and IT administrators, (2) software developers or vendors, and (3) end users or business users. These cloud players vary in their roles under the IaaS, PaaS, and SaaS models. From the software vendors' perspective, application performance on a given cloud platform is most important. From the providers' perspective, cloud infrastructure performance is the primary concern. From the end users' perspective, the quality of services, including security, is the most important.

Cloud Service Tasks and Trends: The top layer is for SaaS applications, mostly used for business applications. The approach is to widen market coverage by investigating customer behavior's and revealing opportunities by statistical analysis. SaaS tools also apply to distributed collaboration, and financial and human resources management.
PaaS is provided by Google, Salesforce.com, and Facebook, among others. IaaS is provided by Amazon, Windows Azure, and RackRack, among others. Colocation services require multiple cloud providers to work together to support supply chains in manufacturing. Network cloud services provide communications such as those by AT&T, Qwest, and Above Net. Details can be found in Clou's introductory book on business clouds.

Software Stack for Cloud Computing: The overall software stack structure of cloud computing software can be viewed as layers. Each layer has its own purpose and provides the interface for the upper layers just as the traditional software stack does. However, the lower layers are not completely transparent to the upper layers.

The platform for running cloud computing services can be either physical servers or virtual servers. By using VMs, the platform can be flexible, that is, the running services are not bound to specific hardware platforms. This brings flexibility to cloud computing platforms. The software layer on top of the platform is the layer for storing massive amounts of data. This layer acts like the file system in a traditional single machine. Other layers running on top of the file system are the layers for executing cloud computing applications. They include the database storage system,

programming for large-scale clusters, and data query language support. The next layers are the components in the software stack.

Runtime Support Services: As in a cluster environment, there are also some runtime supporting services in the cloud computing environment. Cluster monitoring is used to collect the runtime status of the entire cluster. One of the most important facilities is the cluster job management system. The scheduler queues the tasks submitted to the whole cluster and assigns the tasks to the processing nodes according to node availability. The distributed scheduler for the cloud application has special characteristics that can support cloud applications, such as scheduling the programs written in MapReduce style. The runtime support system keeps the cloud cluster working properly with high efficiency. Runtime support is software needed in browser-initiated applications applied by thousands of cloud customers. The SaaS model provides the software applications as a service, rather than letting users purchase the software.

Resource Provisioning and Platform Deployment: In this section, we will discuss techniques to provision computer resources or VMs and storage allocation schemes.

Provisioning of Compute Resources: Providers offer cloud services by signing SLAs with end users. The SLAs must commit sufficient resources such as CPU, memory, and bandwidth that the user can use for a preset period. Under provisioning of resources will lead to broken SLAs and penalties. Overprovisioning of resources will lead to resource underutilization, and consequently, a decrease in revenue for the provider. Deploying an autonomous system to efficiently provision resources to users is a challenging problem. The difficulty comes from the unpredictability of consumer demand, software and hardware failures, heterogeneity of services, power management, and conflicts in signed SLAs between consumers and service providers. Efficient VM provisioning depends on the cloud architecture and management of cloud infrastructures. Resource provisioning schemes also demand fast discovery of services and data in cloud computing infrastructures. In a virtualized cluster of servers, this demands efficient installation of VMs, live VM migration, and fast recovery from failures. To deploy VMs, users treat them as physical hosts with customized operating systems for specific applications. The provider should offer resource economic services. Power-efficient schemes for caching, query processing, and thermal management are mandatory due to increasing energy waste by heat dissipation from data centers.

Resource Provisioning Methods: There are three cases of static cloud resource provisioning policies. In case (a),overprovisioning with the peak load causes heavy resource waste (shaded area). In case(b), under provisioning (along the capacity line) of resources results in losses by both user and provider In case (c), the constant provisioning of resources with fixed capacity to a declining user demand could result in even worse resource waste.

Three resource-provisioning methods are presented here:

1. demand-driven method
2. event-driven method

3. popularity-driven method

Demand-driven method: This method adds or removes computing instances based on the current utilization level of the allocated resources. In general, when a resource has surpassed a threshold for a certain amount of time, the scheme increases that resource based on demand. When a resource is below a threshold for a certain amount of time,that resource could be decreased accordingly. Amazon implements such an auto-scale feature in its EC2 platform.

Event-driven method: This scheme adds or removes machine instances based on a specific time event. The scheme works better for seasonal or predicted events such as Christmastime in the West and the Lunar New Year in the East. During these events, the number of users grows before the event period and then decreases during the event period. This scheme anticipates peak traffic before it happens. The method results in a minimal loss of QoS, if the event is predicted correctly.

Popularity-driven method: In this method, the Internet searches for popularity of certain applications and creates the instances by popularity demand. The scheme anticipates increased traffic with popularity. Again, the scheme has a minimal loss of QoS, if the predicted popularity is correct. Resources may be wasted if traffic does not occur as expected.

Dynamic Resource Deployment: The cloud uses VMs as building blocks to create an execution environment across multiple resource sites. Dynamic resource deployment can be implemented to achieve scalability in performance. The InterGrid is a Java-implemented software system that lets users create execution cloud environments on top of all participating grid resources. Peering arrangements established between gateways enable the allocation of resources from multiple grids to establish the execution environment.

A scenario is illustrated by which an Intergrid gateway (IGG) allocates resources from a local cluster to deploy applications in three steps:

(1) requesting the VMs,

(2) enacting the leases, and

(3) deploying the VMs as requested. Under peak demand, this IGG interacts with another IGG that can allocate resources from a cloud computing provider.

Cloud security and trust management

Cloud security and trust management are critical components of cloud computing, focusing on protecting data and applications in the cloud while ensuring that users can trust the services they use.

Key Aspects of Cloud Security:

1. Data Protection:

Encryption: Encrypting data both at rest and in transit to prevent unauthorized access.

Access Control: Implementing role-based access controls (RBAC) to limit data access based on user roles.

2. Identity and Access Management (IAM):

Managing user identities and their access to cloud resources.

Multi-factor authentication (MFA) to enhance security.

3. **Compliance and Governance:**

Ensuring adherence to regulatory requirements such as GDPR, HIPAA, or PCI-DSS.

Regular audits and assessments to ensure compliance.

4. Network Security:

Firewalls, intrusion detection/prevention systems (IDS/IPS), and secure configurations to protect cloud networks.

5. Incident Response:

Having a robust incident response plan to quickly address security breaches or vulnerabilities.

6. Security Best Practices:

Use multi-factor authentication (MFA) and regularly update software to patch vulnerabilities.

Trust Management in the Cloud:

1. Service Level Agreements (SLAs):

Clearly defined SLAs to set expectations for security, availability, and performance.

2. Third-Party Risk Management:

Assessing the security practices of third-party vendors to ensure better security.

3. Transparency and Accountability:

Cloud providers should be transparent about their security practices and data handling policies.

Regular reports and audits can enhance trust.

4. Reputation and Reviews:

Research the reputation of cloud providers through reviews and third-party assessments.

5. User Education:

Educate users on best practices for cloud security and the importance of maintaining strong passwords and recognizing phishing attempts.

6. Continuous Monitoring: Implement continuous monitoring and analytics to detect anomalies and ensure compliance with security policies.

Effective cloud security and trust management require a proactive approach that combines technology, policies, and user awareness. By focusing on these areas, organizations can safeguard their cloud environments and build trust with users and clients.

Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration. It defines a way to make software components reusable using the interfaces.

Formally, SOA is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a network call over the internet. It uses common communication standards to speed up and streamline the service integrations in applications. Each service in SOA is a complete business function in itself. The services are published in such a way that it makes it easy for the developers to assemble their apps using those services. Note that SOA is different from microservice architecture.

- SOA allows users to combine a large number of facilities from existing services to form applications.
- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

The different characteristics of SOA are as follows :

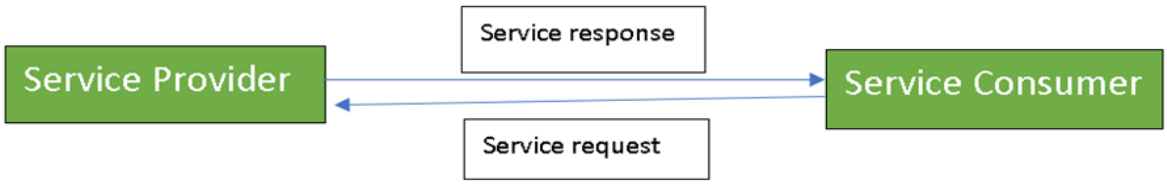
1. Provides interoperability between the services.
2. Provides methods for service encapsulation, service discovery, service composition, service reusability and service integration.
3. Facilitates QoS (Quality of Services) through service contract based on Service Level Agreement (SLA).
4. Provides loosely coupled services.

5. Provides location transparency with better scalability and availability.
6. Ease of maintenance with reduced cost of application development, a registry, together with a service contract and deployment.

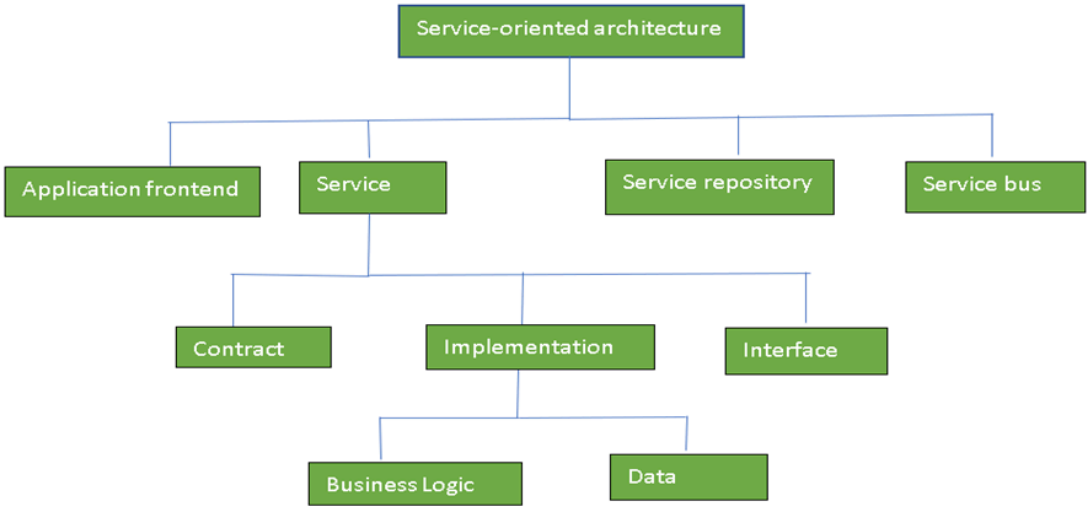
There are two major roles within Service-oriented Architecture:

1. **Service provider:** The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.

2. **Service consumer:** The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.



Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer. This practice is known as service orchestration Another important interaction pattern is service choreography, which is the coordinated interaction of services without a single point of control. **Components of SOA:**



Service-Oriented Architecture (SOA) is a design paradigm that allows for the creation of services

that can be used and reused across different applications. In the context of cloud computing, SOA plays a pivotal role in enabling flexibility, scalability, and interoperability of services. Here's a breakdown of its key components and benefits:

Key Components of SOA in Cloud Computing

1. **Services:**

Modular and self-contained units that perform specific functions.

Can be independently developed, deployed, and maintained.

2. **Interoperability:**

Services can communicate over standardized protocols (e.g., HTTP, REST, SOAP).

Enables integration between different systems and platforms, facilitating seamless data exchange.

3. **Loose Coupling:**

Services are designed to minimize dependencies on one another, allowing for easier updates and modifications without affecting the entire system.

4. **Reusability:**

Services can be reused across different applications, reducing development time and costs.

5. **Discoverability:**

Services can be easily discovered and accessed via service registries, promoting reuse and efficiency.

6. **Scalability:**

Cloud environments can scale services up or down based on demand, allowing for efficient resource utilization.

Benefits of SOA in Cloud Computing

1. **Agility:** Organizations can quickly adapt to changing business needs by leveraging existing services or creating new ones without extensive rework.

2. **Cost Efficiency:** Reduces redundancy by promoting service reuse, which can lower development and operational costs.

3. **Improved Collaboration:** Teams can work on different services independently, facilitating parallel development and faster delivery.

4. **Easier Integration:** SOA simplifies the integration of legacy systems with modern cloud applications, enhancing overall system interoperability.

5. Dynamic Scaling: In a cloud environment, services can scale dynamically based on demand, ensuring performance and reliability during peak loads.

6. Enhanced Maintainability: With a modular architecture, maintaining and updating services becomes easier and less disruptive to the overall system.

Challenges of SOA in Cloud Computing:

1. **Complexity:** Managing multiple services can introduce complexity in architecture and orchestration.

2. **Security:** Ensuring secure communication and data protection between services can be challenging, requiring robust security measures.

3. **Performance:** Network latency and service dependencies can impact performance; careful design is needed to mitigate these issues.

4. **Governance:** Establishing clear governance and policies for service development, deployment, and usage is essential for maintaining control.

SOA in cloud computing enhances flexibility and efficiency by promoting modular, reusable services. While there are challenges to overcome, the benefits in agility, cost savings, and improved integration make SOA a valuable approach for organizations leveraging cloud technologies.

Practical applications of SOA: SOA is used in many ways around us whether it is mentioned or not.

1. SOA infrastructure is used by many armies and air forces to deploy situational awareness systems.

2. SOA is used to improve healthcare delivery.
3. Nowadays many apps are games and they use inbuilt functions to run. For example, an app might need GPS so it uses the inbuilt GPS functions of the device. This is SOA in mobile solutions.
4. SOA helps maintain museums with a virtualized storage pool for their information and content.

Message Oriented Middleware

The full form of MOM is Message-Oriented Middleware which is an infrastructure that allows communication and exchanges the data (messages). It involves the passing of data between applications using a communication channel that carries self-contained units of information (messages).In a MOM-based communication environment, messages are sent and received asynchronously.

MOM provides asynchronous communication, and it just sends the message and performs its asynchronous operations. It consists of inter-application communication software that relies on asynchronous message passing which would oppose request-response architecture. So an asynchronous system consists of a message queue that provides a temporary stage so that the destination program becomes busy or might not be connected. Message Queue helps in storing the message on a MOM platform. MOM clients can send and receive the message through the queue.

Queues act as a central component for implementing asynchronous interaction within MOM.

- Middleware is software that acts as a link between two or more objects
- Middleware simplifies complex distributed applications,
- It consists of web servers, application servers, and more, it is integral to modern information technology based on XML, SOAP, service-oriented architecture.

Block Representation of Middleware:

Features and Capabilities:

1. Unified messaging
2. Provisioning and monitoring
3. Dynamic scaling
4. Management and control tools
5. Dynamic scaling
6. Flexible service quality
7. Secure communication
8. Integration with other tools

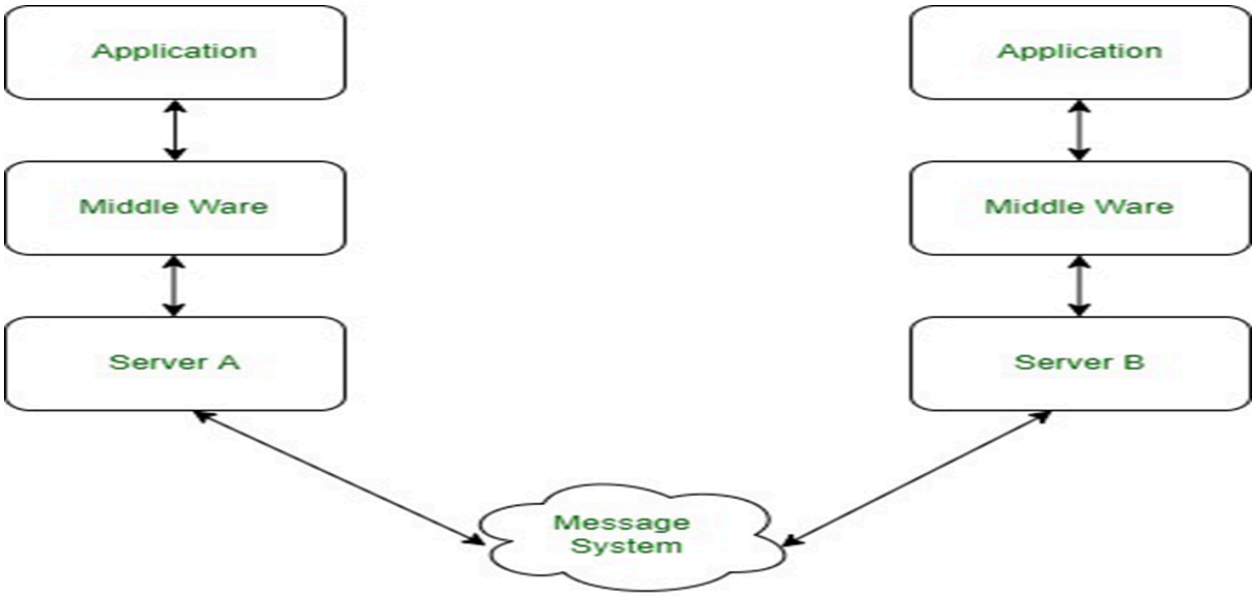


Fig: Middleware

Structure and working of the Message Queue in MOM:

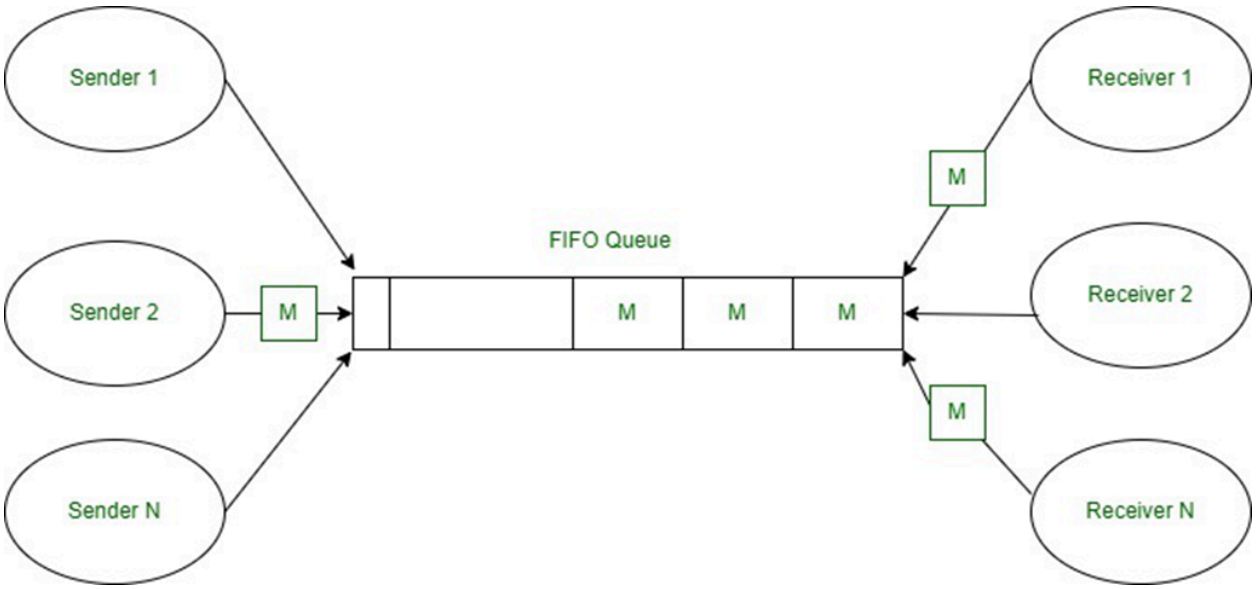
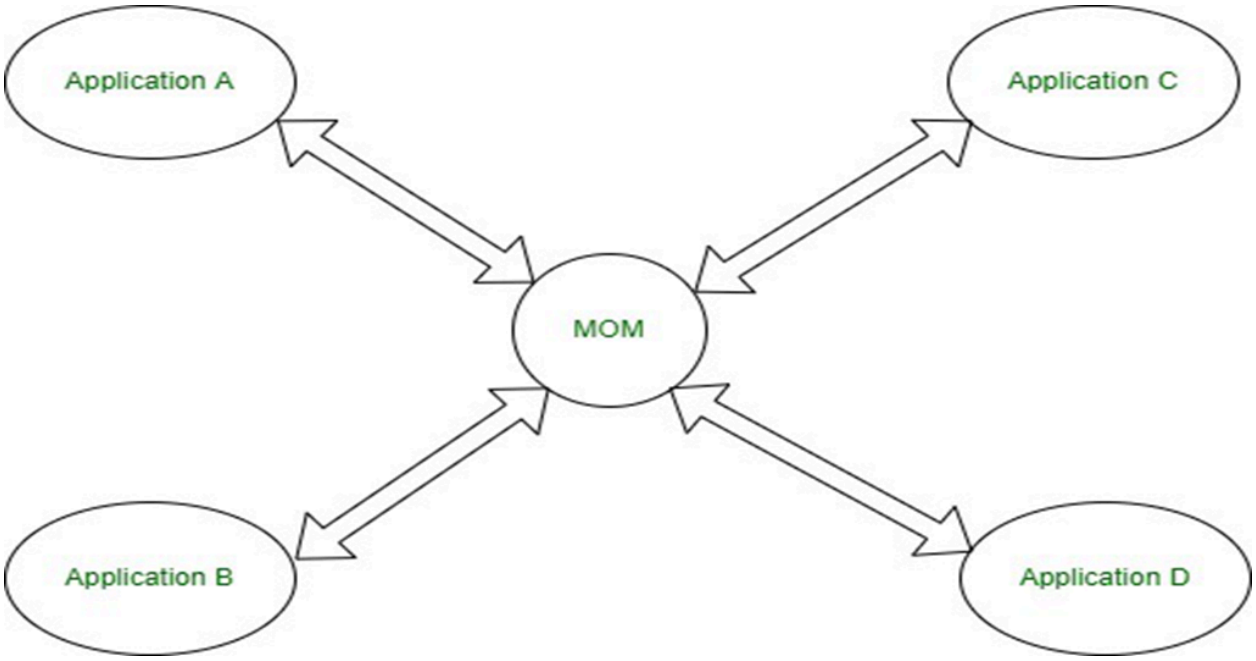


Fig: Message Queue in MOM

MOM enables communication between distributed components. Middleware makes programming easier in a distributed environment. It acts like an Operating System (OS) for distributed computing architecture and provides transparency for the applications.

A message-oriented middleware acts as a middleware for different applications for example:



MOM Deployment:

The above diagram is a message-oriented middleware-based distributed system deployment that offers a service-based approach to inter-process communication. The messaging of MOM is the same as the postal service.

The Architecture of Message Oriented Middleware:

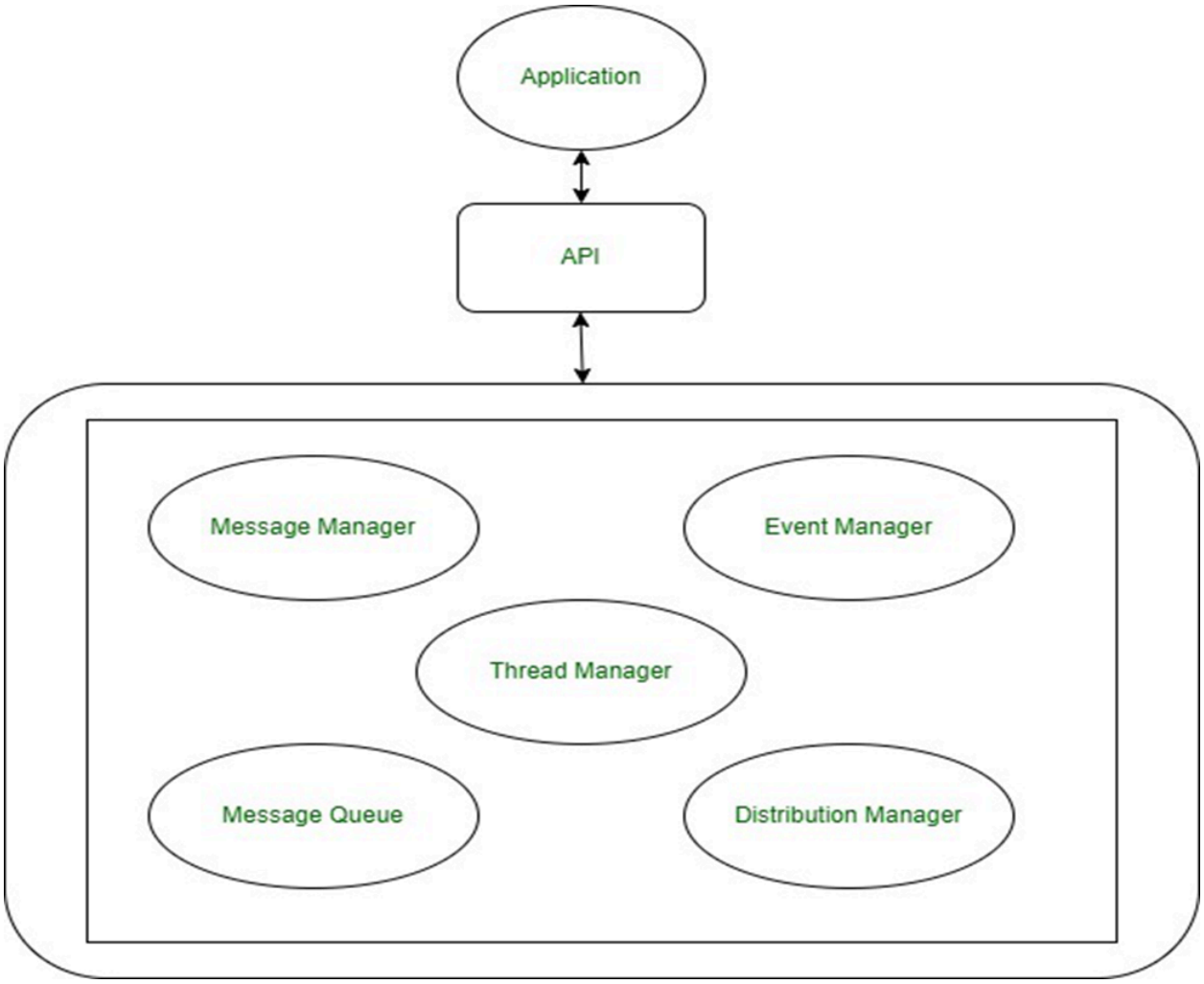


Fig: Architecture of MOM

Types of middleware:

1. Database Middleware
2. Application Server Middleware
3. Messaging middleware
4. Message-oriented Middleware
5. Transaction processing middleware

Roles of message-oriented middleware

1. The message distribution is enabled over complex IT systems.
2. It serves as a connector for two different applications or platforms.
3. MOM helps in implementing the delivery of messages across different IT organizations.
4. It will create a distributed product that is compatible with the various OS.

5. MOM allows various software components can talk to each other.
6. It is a type of middleware that consists of several lines that are connected to different applications.
7. It connects different technologies involving message origination and delivery destination.
8. It links front and back-end systems.

Example:

MQTT(Message Queuing for Telemetry Transport): Most MQ systems and protocols are aimed at back-end and enterprise applications and these types of technologies are not suited for constrained devices like sensor nodes. Such devices are typically constrained in terms of memory, bandwidth, and power.

MQTT is a message-oriented protocol aimed at applications like wireless sensor networks, M2M(mobile 2 mobile) and ultimately the internet of things(a large number of nodes and applications loosely through a messaging system).

Advantages

1. Loose coupling
2. Scalability
3. Fast
4. Reliability
5. Availability

Disadvantages

1. Requires extra component in the architecture
2. Poor programming abstraction
3. One-to-one communication for queue abstraction
4. Not implemented for some platform