**UNIT – 3**

**Cloud Platform Architecture**

1. **Cloud Computing and Service Models**: In recent days, the IT industry has moved from manufacturing to offering more services (service-oriented). As of now, 80% of the industry is ‘service-industry’. It should be realized that services are not manufactured/invented from time-to-time; they are only rented and improved as per the requirements.

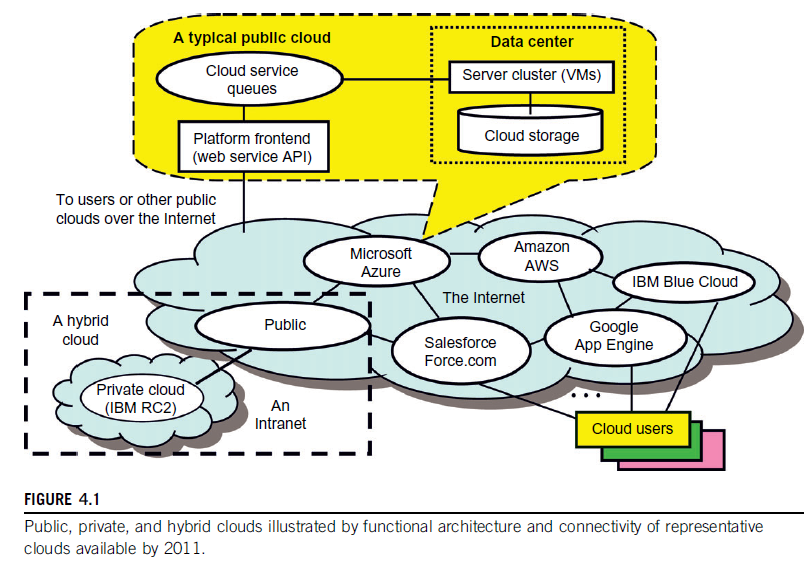
Clouds aim to utilize the resources of data centers virtually over automated hardware, databases, user interfaces and apps [1].

1. **Public, Private and Hybrid Clouds**: Cloud computing has evolved from the concepts of clusters, grids and distributed computing. Different resources (hardware, finance, time) are leveraged (use to maximum advantage) to bring out the maximum HTC. A CC model enables the users to share resources from anywhere at any time through their connected devices.

**Advantages of CC**: Recall that in CC, the programming is sent to data rather than the reverse, to avoid large data movement, and maximize the bandwidth utilization. CC also reduces the costs incurred by the data centers, and increases the app flexibility.

CC consists of a virtual platform with elastic resources [2] and puts together the hardware, data and software as per demand. Furthermore, the apps utilized and offered are heterogeneous.

* 1. **The Basic Architecture** of the types of clouds can be seen in Figure 4.1 [1] below.



* 1. **Public Clouds**: A public cloud is owned by a service provider, built over the Internet and offered to a user on payment. Ex: Google App Engine (GAE), AWS, MS-Azure, IBM Blie Cloud and Salesforce-Force.com. All these offer their services for creating and managing VM instances to the users within their own infrastructure.

**Private Clouds**: A private cloud is built within the domain of an **intranet** owned by a single organization. It is client-owned and managed; its access is granted to a limited number of clients only. Private clouds offer a flexible and agile private infrastructure to run workloads within their own domains. Though private cloud offers more control, it has limited resources only.

**Hybrid Clouds**: A hybrid cloud is built with both public and private clouds. Private clouds can also support a hybrid cloud model by enhancing the local infrastructure with computing capacity of a public external cloud.

**Summary**: **Public** clouds provide standardization, preserve the investment and offer flexibility. **Private** clouds attempt to achieve customization (modify to suit the current situation), higher efficiency, resilience (capacity to recover quickly from difficulties), security and privacy. **Hybrid** clouds work in the middle with many compromises in resource sharing.

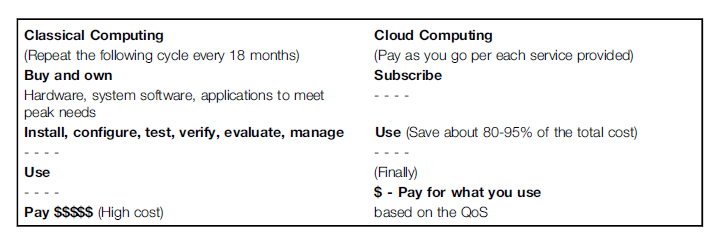
* 1. **Data Center Networking Architecture**: The core of a cloud is the server cluster and the cluster nodes are used as compute nodes. The scheduling of user jobs requires that virtual clusters are to be created for the users and should be granted control over the required resources. **Gateway nodes** are used to provide the access points of the concerned service from the outside world. They can also be used for security control of the entire cloud platform. It is to be noted that in physical clusters/grids, the workload is static; in clouds, the workload is dynamic and the cloud should be able to handle any level of workload on demand.

**Differences between Data Centers and Super Computers**: In data centers, scalability is a fundamental requirement. Note that data centers have multiple servers. Ex: MS-Chicago Data Center has 100,000 eight-core servers housed in 50 containers (2000 in each). In supercomputers, a separate **data farm** [3] is used; a data center uses disks on server nodes plus memory cache and DBs.

Data Centers and Supercomputers also possess different networking requirements. (bandwidth, routers used etc.)

NOTE: **Data Farm** => Data farming is the process of using computational experiments to ‘grow’ or increase data which can be utilized for statistical analyzing.

* 1. **Cloud Development Trends**: There is a good chance that private clouds will grow in the future since private clouds are more secure, and adjustable within an organization. Once they are matured and more scalable, they might be converted into public clouds. In another angle, hybrid clouds might also grow in the future.
  2. **Cloud Ecosystem and Enabling Technologies**: The differences between classical computing and cloud computing can be seen in the table [1] below. In traditional computing, a user has to buy the hardware, acquire the software, install the system, test the configuration and execute the app code. The management of the available resources is also a part of this. Finally, all this process has to be revised for every 1.5 or 2 years since the used methodologies will become obsolete.



On the other hand, CC follows a pay-as-you-go model [1]. Hence the cost is reduced significantly – a user doesn’t buy any resources but rents them as per his requirements. All S/W and H/W resources are leased by the user from the cloud resource providers. This is advantageous for small and middle business firms which require limited amount of resources only. Finally, CC also saves power.

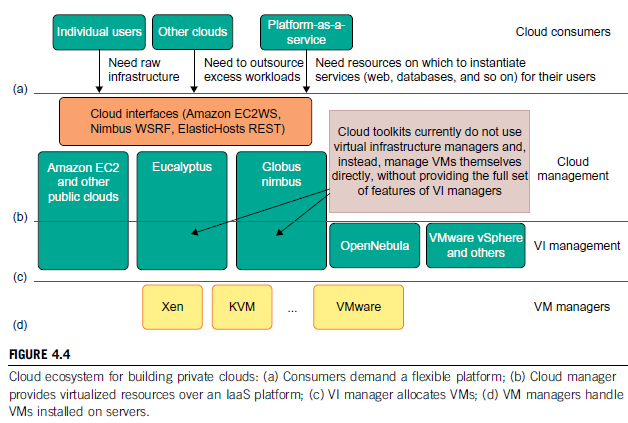
* 1. **Cloud Design Objectives**:
* Shifting computing from desktops to data centers
* Service provisioning and cloud economics
* Scalability in performance (as the no. of users increases)
* Data Privacy Protection
* High quality of cloud services (QoS must be standardized to achieve this)
* New standards and interfaces
  1. **Cost Model**:



The above Figure 4.3a [1] shows the additional costs on top of fixed capital investments in traditional computing. In CC, only pay-as-per-use is applied, and user-jobs are outsourced to data centers. To use a cloud, one has no need to buy hardware resources; he can utilize them as per the demands of the work and release the same after the job is completed.

* 1. **Cloud Ecosystems**: With the emergence of Internet clouds, an ‘ecosystem’ (a complex inter-connected systems network) has evolved. This consists of users, providers and technologies. All this is based mainly on the open source CC tools that let organizations build their own IaaS. Private and hybrid clouds are also used. Ex: Amazon EC2.

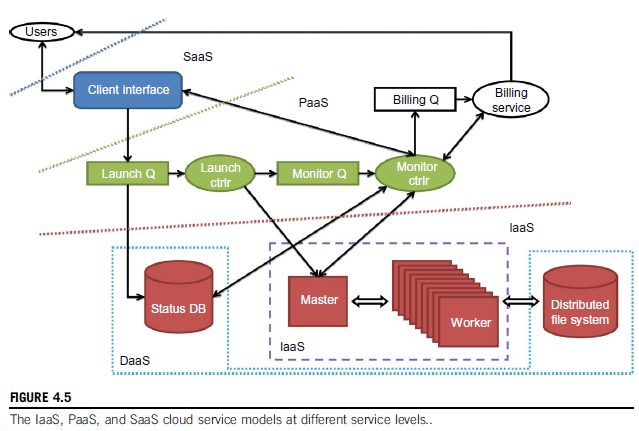
An **ecosystem for private clouds** was suggested by scientists as depicted in Figure 4.4 [1].



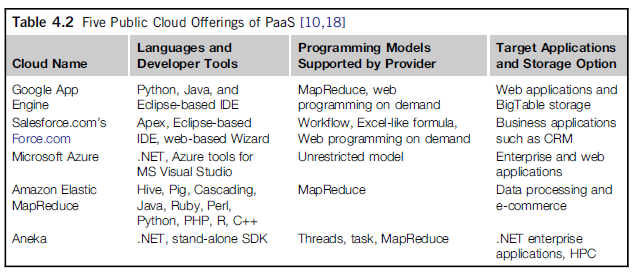
In the above suggested 4 levels, at the **user end**, a flexible platform is required by the customers. At the **cloud management level**, the VZ resources are provided by the concerned cloud manager to offer the IaaS. At the **VI management level**, the manager allocates the VMs to the available multiple clusters. Finally, at the **VM management level**, the VM managers handle VMs installed on the individual host machines.

* 1. **Increase of Private Clouds**: Private clouds influence the infrastructure and services that are utilized by an organization. Private and public clouds handle the workloads dynamically but public clouds handle them without communication dependency. On the other hand, private clouds can balance workloads to exploit the infrastructure effectively to obtain HP. The major advantage of private clouds is less security problems and public clouds need less investment.
  2. **Infrastructure-as-a-Service (IaaS)**: A model for different services is shown in Figure 4.5 [1], as shown below. The required service is performed by the rented cloud infrastructure. On this environment, the user can deploy and run his apps. Note that user doesn’t have any control over the cloud infrastructure but can choose his OS, storage, apps and network components.

Ex: Amazon EC2.



* 1. **Platform-as-a-Service (PaaS)**: To develop, deploy and manage apps with provisioned resources, an able platform is needed by the users. Such a platform includes OS and runtime library support. Different PaaS offered in the current market and other details are highlighted in the Table 4.2 [1] below:



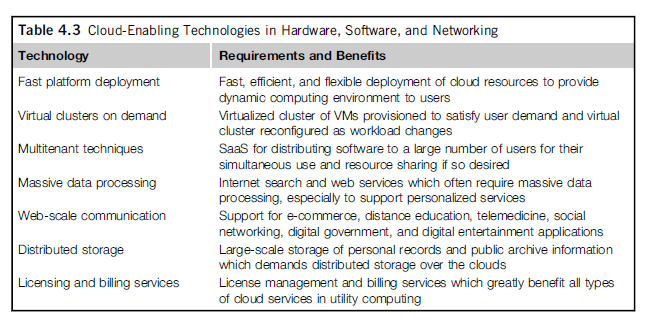
It should be noted that platform cloud is an integrated system consisting of both S/W and H/W. The user doesn’t manage the cloud infrastructure but chooses the platform that is best suited to his choice of apps. The model also encourages third parties to provide software management, integration and service monitoring solutions.

* 1. **Software as a Service (SaaS)**: This is about a browser-initiated app s/w over thousands of cloud customers. Services & tools offered by PaaS are utilized in construction and deployment of apps and management of their resources. The customer needs no investment and the provider can keep the costs low. Customer data is also stored in a cloud and is accessible through different other services. Ex: Gmail, Google docs, Salesforce.com etc.
  2. **Mixing of Cloud Services**: Public clouds are more used these days but private clouds are not far behind. To utilize the resources up to the maximum level and deploy/remove the apps as per requirement, we may need to mix-up the different parts of each service to bring out a chain of connected activities. Ex: Google Maps, Twitter, Amazon ecommerce, YouTube etc.

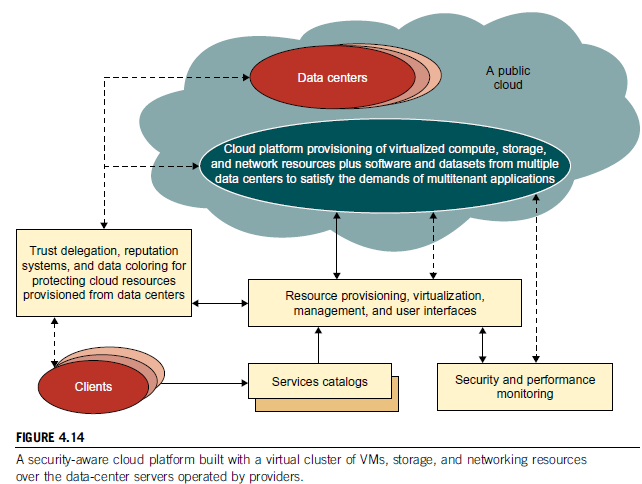
1. **Architectural Design of Compute and Storage Clouds**: An Internet cloud [4] (CC) is envisaged (imagined) as a public cluster of servers allocated on demand to perform collective web services or distributed apps using the resources of a data center.
   1. **Cloud Platform Design Goals**: The major goals of a cloud computing platform are scalability, efficiency, VZ, and reliability. A cloud platform manager receives the user requests, finds the resources, and calls the provisioning services to allocate the appropriate amount of resources for the job. Note that a manager supports both physical and virtual machines.

The platform also needs to establish an infrastructure that can obtain HPC. Scalability can be obtained by adding more data centers or servers, which leads to more efficient data distribution and, usage of less power and bandwidth.

* 1. **Enabling Technologies for Clouds**: The important motives behind the growth of CC are the ubiquity (present everywhere) of broadband and wireless networking, falling costs of storage, remove unneeded storage. Service-providers like Amazon and Google can make the utilization of available resources more efficient through multiplexing [5] (incorporate into an existing system), VZ, and dynamic resource provisioning. In Table 4.3 [1], the enabling of clouds is summarized.



* 1. **Cloud Architecture**: A generic cloud architecture can be seen Figure 4.14 [1]. The Internet Cloud is imagined as a massive cluster of servers. The different resources (space, data, and speed) of the concerned servers are allocated as per demand dynamically.



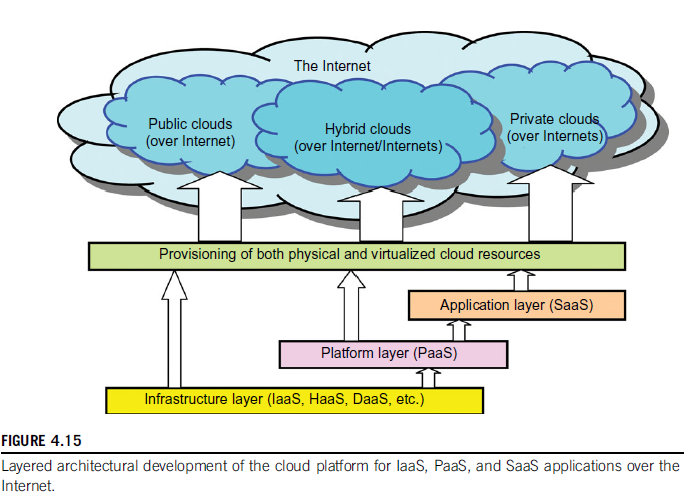
NOTE: **Data colouring [6] (like watermarking)** protects shared data objects and ensures the security level in the cloud. These techniques safeguard multi-way authentications, enable single sign-on in the cloud, and strengthen the security for accessing confidential data in both public and private clouds.

The cloud platform demands distributed storage and different services (PaaS, IaaS and SaaS). Though the resources and services do exist and work in parallel, the user need not know about the real-work behind the screen. Any software in the cloud is a service and any service demands high amount of trust on the data retrieved from the data. Other cloud resources include storage area networks (**SAN**s), firewalls, and security devices.

The **usage and performance of granted resources** are monitored and metered by special units. The **software infrastructure of a cloud platform** must automatically handle all the resource grants and management and note the status of each node system/server when it joins/leaves the cluster. The physical location of the data center, type of power used (general/solar/hydroelectric) and cooling required are also important points.

Typically, **private clouds are easier to manage and public clouds are easier to access**. In future the clouds which utilize the best resources from both the types (hybrid) are expected to grow. Finally, security becomes a critical issue in CC to grant the success of all the services.

* 1. **Cloud Architecture in Layers**: Cloud architecture is developed at three layers: infrastructure, platform and app. This can be noticed in Figure 4.15 [1].



Different VZ standards are framed and utilized in all these layers to provision the resources allocated for a cloud. The services offered to public, private and hybrid through different networking supports over the Internet and intranets.

* **Infrastructure layer** is deployed first to support the IaaS layer. It also serves as a foundation for the PaaS layer services.
* **Platform layer** itself is a foundation for the SaaS services.

The layers demand resource allocation as per demand and are granted.

* The **infrastructure layer is built with** virtualized compute, storage, and network resources. Proper utilization of these resources provides the flexibility demanded by the users. Note that **VZ demands automated provisioning** of the resources and minimum management time.
* The **platform layer** is for general purpose and repeated usage of the service resources. Proper environment is provided for the development, testing, deployment and monitoring the usage of apps. Indirectly, a virtualized cloud platform acts as a ‘system middleware’ between the infrastructure and application layers of a cloud.
* The **application layer** is formed with the collection of different modules of all software that are needed for the SaaS apps. The general service apps include those of information retrieval, doc processing, and authentication services. This layer also used in large-scale by the CRMs, financial transactions, and supply chain management.

Note that all the layers are built from the scratch (bottom-up) with dependence relations in between.

* 1. **NOTE**: In general, SaaS demands most work from the provider, PaaS in the middle, and IaaS demands the least. Ex: Amazon EC2. Services at app layer demand more work from the providers. Ex: Salesforce.com CRM service.
  2. **Market-Oriented Cloud Architecture**: This can be seen in the Figure 4.16 below.



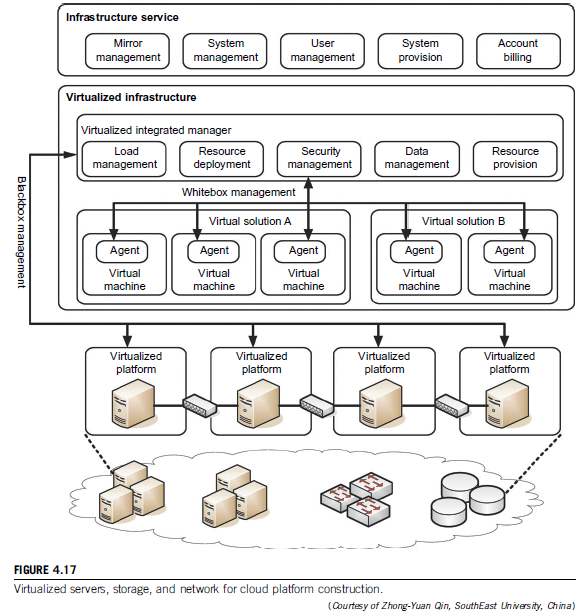
(SLA=> Service Level Agreements)

A high level architecture can be seen in the figure for supporting market oriented resource allocation in a CC environment. The entities here are users, brokers (acting on behalf of a set of similar users), and resource allocators. When a request is made, the service request examiner comes into picture and acts as an interface between the user and the data center resources.

* 1. **QoS factors**: In CC, different services being offered as commercial options in the market should take into account diverse factors for every service request: time, cost, reliability, and security. The QoS requirements can’t be static and might from time to time on demand. Importance must be given to the customer, his requests and requirements – he is paying for all these. For achieving all these accomplishments in the CC market, the CRM steps into picture and plays a crucial role to satisfy each and every customer.
  2. **VZ Support and Disaster Recovery**: System VZ is a much used feature in CC to improve provisioning of the resources to various services or customers. The provisioning tools, through the VMs (containers of services), try to find the best physical location wherein they plug the nodes into the data centers.

In CC, VZ also means the resources and fundamental infrastructures are virtualized. The user need not care about the computing resources, where and how they are deployed and used. The user only uses the service offers as current situation demands.

* 1. **Hardware VZ**: System VZ is a special kind of technique that simulates the hardware execution, utilization and provisioning methods before they can be applied in the real world of CC. VZ software is used for simulations, platform-developing for clouds, and use any kind of OS that is preferred by a developer/user. The infrastructure needed by the servers to VZ the whole data center and utilize it for CC is given below in Figure 4.17 [1].



* 1. **Using VMs in CC** ensures maximum flexibility for the users. A proper methodology is required for correct provisioning of the resources, distribute the burdens of space and time evenly and bring out HP. Traditional sharing of cluster resources doesn’t confirm the above stated goals and an appropriate usage of all the hardware resources in all angles can be brought out by VZ of the same resources.
  2. **VZ Support in Public Clouds**: Public clouds like AWS, MS-Azure, GAE are the famous products in the market. AWS provides extreme flexibility through VMs for the users to execute their own applications. GAE provides limited app level VZ for users since it supports only Google’s services.MS provides programming level VZ for users to build their own apps.

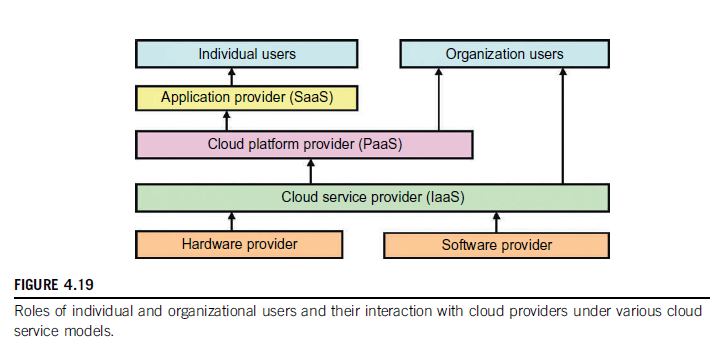
Continuing, the VMware tools apply to workstations, servers and virtual infrastructure. The MS tools are mainly used on PCs and some servers. **The entire IT industry is changing its look** and becoming more embedded in the cloud. **VZ leads to** HA (high availability), disaster recovery, dynamic load levelling, and commendable provisioning support. Both CC and utility computing leverage (use to the maximum advantage) the benefits of VZ to increase scalability and provide an autonomous computing environment.

* 1. **VZ for IaaS**: VM technology is ubiquitous (present everywhere) enabling the users to create customised environments atop physical infrastructure. **Advantages** are: The under-utilized servers can be removed and the workload can be evenly distributed among the existing servers, VMs can run their code without conflicting with other APIs, VMs can also be used to improve security through sandbox methodology (tightly controlled set of resources) and VZ cloud platforms can isolate their performance also, increasing the QoS.
  2. **VM Cloning for Disaster Recovery**: [Cloning => Make an identical copy] There exist two methods to recover from any disaster. In the first scheme, a physical machine is recovered by another physical machine. Apparently, this takes more time, energy and is more expensive. The needed hardware is to be setup, the OS is to be installed and the data recovery process has to be adjusted to other requirements too. In the other methodology, to recover a VM platform, no installation, configuration, OS setup etc. are needed – the time utilized becomes 40% less than the previous scheme.
  3. **Architectural Design Challenges**:

1. **Service Availability and Data Lock-in Problem**: If all the cloud services are functioning under a single company, that itself may be the reason of failure of the cloud. To achieve HA, it is advisable to use different services from multiple companies. Another obstacle for availability is DDoS attacks and ransomware.

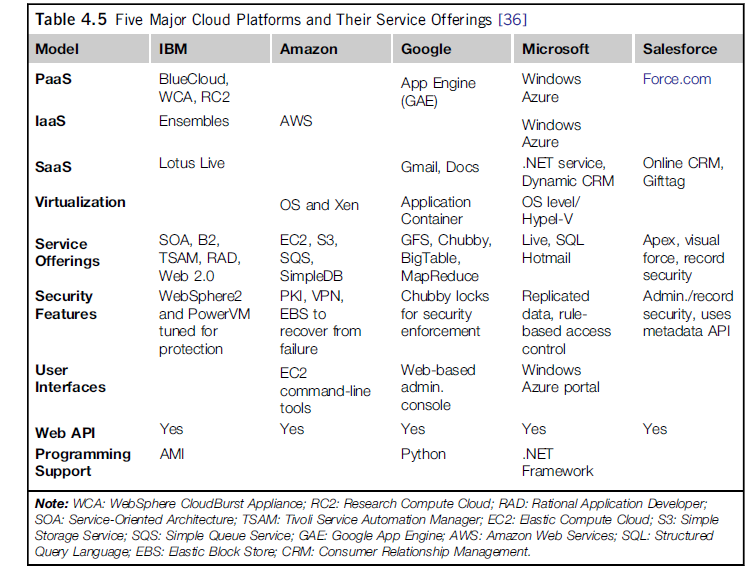
Software storage and usage in a distributed manner is being done systematically, but the APIs are still vulnerable to attacks. The solution to this challenge is to standardize the APIs that are used in SaaS; all this enables the usage of a new model in public and private clouds. All this leads to ‘**surge computing**’ where extra tasks are performed by public clouds, which can’t be done in the case of private clouds.

1. **Data Privacy and Security**: Present cloud offerings are public, but this makes them more exposed and prone to attacks. The steps that are to be taken are encrypted storage, virtual LANs, firewalls, and packet filters. The attacks that might try to intrude the cloud are malware, spyware, hijacking, DDoS, man in the middle (while migrating) and others.
2. **Unpredictable Performance and Bottlenecks**: Multiple VMs can share CPUs and main memory in CC, but I/O sharing is difficult and cumbersome. As a solution one might try to improve the I/O architectures and operating systems to virtualize the interrupts and I/O channels. Finally, in the clouds, the data bottlenecks must be removed or widened to obtain the efficient HP.
3. **Distributed Storage and Widespread Bugs**: DB usage is growing in CC and all of it can’t be stored at a single place. Distributed storage thus comes into picture, buts also brings new problems like requirement of efficient SANs (Storage Area Network), and data durability. Simulator is a nice way to understand the problem and propose a satisfactory solution.
4. **Cloud Scalability, Interoperability and Standardization**
5. **Software Licensing**: Since distributed computing is widely used, any single customer’s unsatisfactory usage of the concerned service may collapse the whole cloud
   1. **Public Cloud Platforms**: Cloud services are provided as per demand by different companies. It can be seen in Figure 4.19 [1] that there are 5 levels of cloud players.



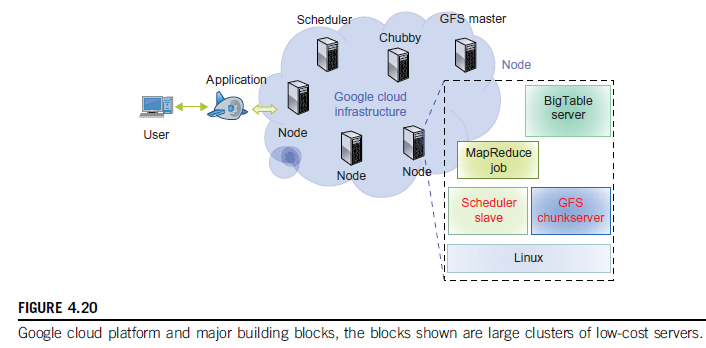
The app providers at the SaaS level are used mainly by the individual users. Most business organisations are serviced by IaaS and PaaS providers. IaaS provides compute, storage, and communication resources to both app providers and organisational users. The cloud environment is defined by PaaS providers. Note that PaaS provides support both IaaS services and organisational users directly.

Cloud services depend upon machine VZ, SOA, grid infrastructure management and power efficiency. The provider service charges are much lower than the cost incurred by the users when replacing damaged servers. The Table 4.5 shows a summary of the profiles of the major service providers.



PKI=> Public Key Infrastructure; VPN=> Virtual Private Network

* 1. **Google App Engine (GAE)**: The Google platform is based on its search engine expertise and is applicable to many other areas (Ex: MapReduce). The **Google Cloud Infrastructure** consists of several apps like Gmail, Google Docs, and Google Earth and can support multiple no. of users simultaneously to raise the bar for HA (high availability). Other technology achievements of Google include Google File System (GFS) [like HDFS], MapReduce, BigTable, and Chubby (A Distributed Lock Service). GAE enables users to run their apps on a large number of data centers associated with Google’s search engine operations. The GAE architecture can be seen in Figure 4.20 [1] below:



The building blocks of Google’s CC app include GFS for storing large amounts of data, the MapReduce programming framework for developers, Chubby for distributed lock services and BigTable as a storage service for accessing structural data.

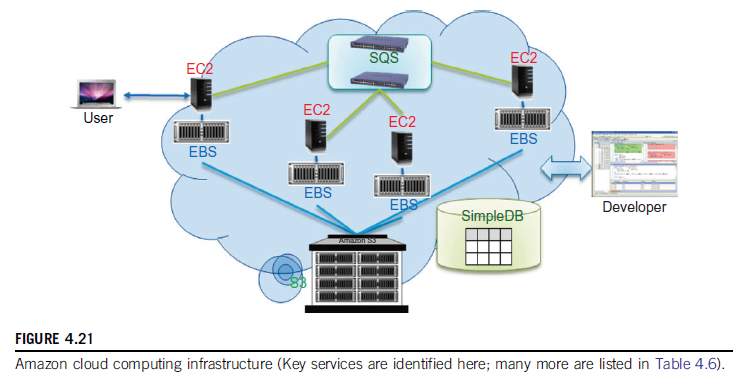
GAE runs the user program on Google’s infrastructure where the user need not worry about storage or maintenance of data in the servers. It is a combination of several software components but the frontend is same as ASP (Active Server Pages), J2EE and JSP.

* 1. **Functional Modules of GAE**:

1. **Datastore** offers OO, distributed and structured data storage services based on BigTable techniques. This secures data management operations.
2. **Application Runtime Environment:** It is a platform for scalable web programming and execution. (Supports the languages of Java and Python)
3. **Software Development Kit:** It is used for local app development and test runs of the new apps.
4. **Administration Console**: Used for easy management of user app development cycles instead of physical resource management.
5. **Web Service Infrastructure** provides special interfaces to guarantee flexible use and management of storage and network resources.

The well-known GAE apps are the search engine, docs, earth and Gmail. Users linked with one app can interact and interface with other apps through the resources of GAE (synchronise and one login for all services).

* 1. **Amazon Web Services (AWS)**: Amazon applies the IaaS model in providing its services. The Figure 4.21 [1] below shows the architecture of AWS:

****

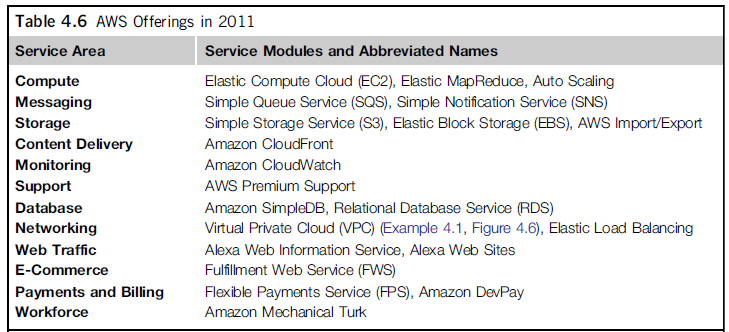
**EC2** provides the virtualized platforms to host the VMs where the cloud app can run.

**S3** (Simple Storage Service) provides the OO storage service for the users.

**EBS** (Elastic Block Service) provides the block storage interface which can be used to support traditional apps.

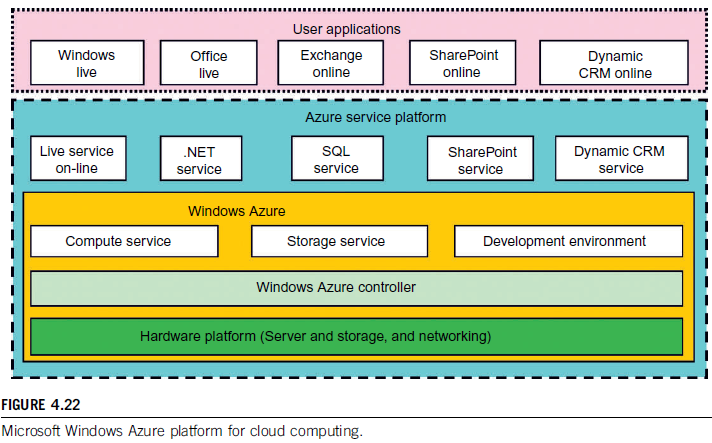
**SQS** (Simple Queue Service) ensures a reliable message service between two processes.

Amazon offers a **RDS** (relational database service) with a messaging interface. The AWS offerings are given below in Table 4.6 [1].



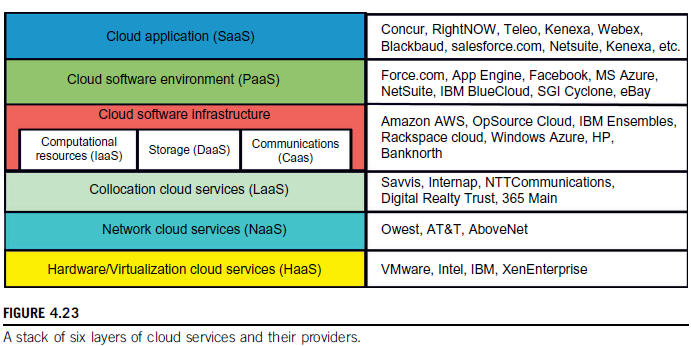
* 2. **MS-Azure**: The overall architecture of MS cloud platform, built on its own data centers, is shown in Figure 4.22 [1]. It is divided into 3 major component platforms as it can be seen. Apps are installed on VMs and Azure platform itself is built on Windows OS
  3. .





* **Live Service**: Through this, the users can apply MS live apps and data across multiple machines concurrently.
* **.NET Service**: This package supports app development on local hosts and execution on cloud machines.
* **SQL Azure**: Users can visit and utilized the relational database associated with a SQL server in the cloud.
* **SharePoint Service**: A scalable platform to develop special business apps.
* **Dynamic CRM Service**: This provides a business platform for the developers to manage the CRM apps in financing, marketing, sales and promotions.

1. **Inter-Cloud Resource Management**
   1. **Extended CC Services**: This can be viewed in Figure 4.23 [1]:



The top three service layers are SaaS, PaaS and IaaS. The bottom three layers are related to physical requirements and are as Hardware as a Service (HaaS), Network as a Service (NaaS), Location as a Service (LaaS), and Security as a Service (SaaS).

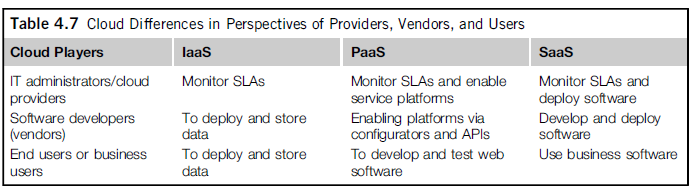


Table 4.7 [1] shows that cloud players are into three classes.

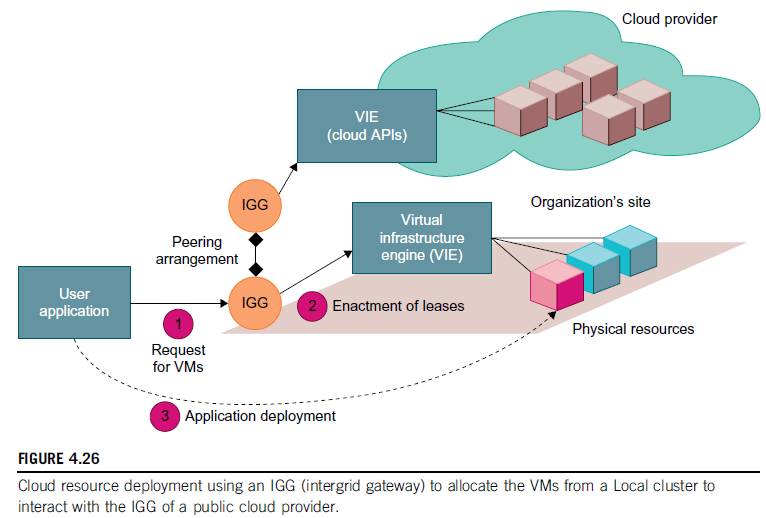
* 1. **Software Stack for CC**: A software stack [7] is a group of programs that work in tandem (in order) to produce a common goal. It may also refer to any set of apps that works in a specific order toward a common goal. Ex: Like a set in maths or a cluster in DM. The system has to be designed to meet goals like HT, HA, and fault tolerance. Physical or virtual servers can be used making the platform more flexible and be able to store and utilize large amount of data.
  2. **Resource Provisioning and Platform Deployment:**

1. **Provisioning of Compute Resources (VMs)**: The provisioning of resources like CPU, memory, and bandwidth are distributed among the users as per the service level agreements (SLAs) signed before the start of the work. The problem here is the ever-changing levels of requests from the user, power management and conflicts in the SLAs.

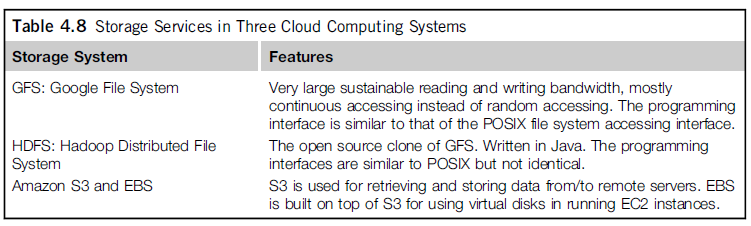
Efficient VM provisioning depends on the cloud architecture and management of cloud infrastructures. Resource provisioning also demands fast discovery of services and data in the provided infrastructure. Ex: Efficient installation of VMs, live VM migration, and fast recovery from failures. Providers like Amazon, IBM and MS-Azure use VM templates, automation of provisioning and power-efficient schemes.

1. **Resource Provisioning Methods**:
2. **Demand**-**Driven Resource Provisioning**: This method adds or removes computing instances based on the current utilization level for the allocated resources. This method automatically allocates two processors for the user app, if the user utilizes more than 60% of time for an extended period. That is, if **the resource utilization has crossed a threshold** of the concerned resource, **extra resources will be allocated**. This methodology is implemented by Amazon in EC2.
3. **Event**-**Driven Resource Provisioning**: This scheme adds or removes machine instances based on an event like festival season. At this time, the no. of users peaks and so does the traffic. This anticipation results in good QoS and customer satisfaction.
4. **Popularity-Driven Resource Provisioning:** In this method, The Internet searches for popularity of certain apps and creates extra instances if the popularity has risen.
5. **Dynamic Resource Deployment:** This can be implemented to achieve scalability in performance through efficient allocation of resources at every place in the grid as the situation demands. To achieve this, we need an **inter-grid gateway** (IGG) between different grids that allocates the resources from a local cluster to deploy apps by **requesting the VMs**, **enacting** (endorse) **the leases**, and **deploying the VMs as per requests**.

The Inter-Gridprovidesandallocates a **distributed virtual environment (DVE**). It is a virtual cluster of VMs that runs in isolation from other virtual clusters. This process is carried out by a component called DVE manager. Received massages are handled in parallel in a thread pool. All these methodologies are depicted in Figure 4.26.



1. **Provisioning of Storage Resources**: The data in CC is stored in the clusters of the cloud provider and can be accessed anywhere in the world. Ex: email. For data storage, distributed file system, tree structure file system, and others can be used. Ex: GFS, HDFS, MS-Cosmos. This method provides a convenient coding platform for the developers. The storage methodologies and their features can be found in Table 4.8 [1].



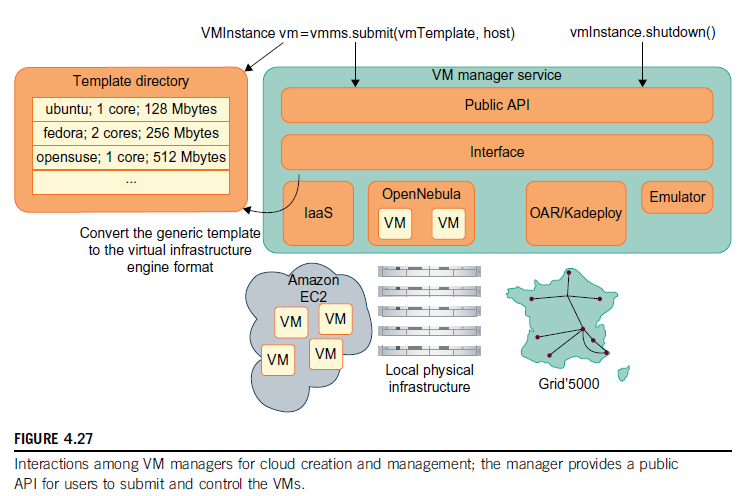
POSIX => Portable OS Interface

EBS => Elastic Block Storage

EC2 => Elastic Compute Cloud

S3 => Amazon Simple Storage Service

* 1. **Virtual Machine Creation & Management**: Figure 4.27 [1] shows the interactions among VM managers for cloud creation and management.

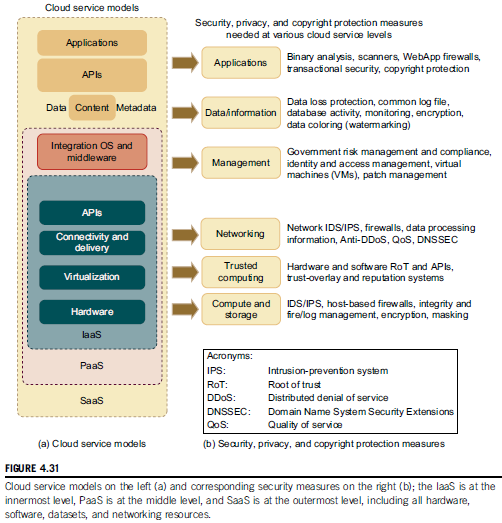


1. **Independent Service Management**: By using independent service providers, the cloud apps can run different services at the same time. Some other services are used for providing data other than the compute or storage services.
2. **Running Third Party Apps**: IN this case, the cloud platforms have to provide support for apps constructed by third-party app providers. The concerned APIs are in the form of services provided by another company. (Ex: Dropbox + Gmail + User).
3. **VM Manager**: It is a link between the gateway and resources. The physical resources aren’t shared directly, but in a virtualized method. The VMs themselves become the actual resources. Ex: OpenNebula (an OS). Users submit VMs on physical machines using hypervisors, which enables the running of several operating systems on the same host concurrently.
4. **VM Templates**: A VM template is analogous (similar) to the configuration of a computer and contains the description for a VM. Information provided is:

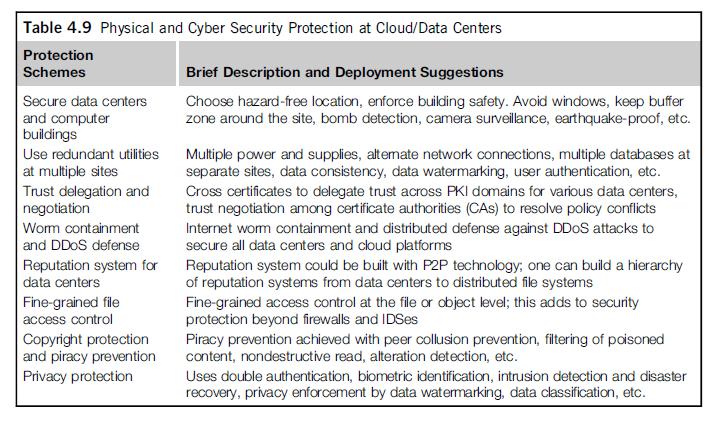
* The no. of processors allocated to the VM
* Memory required by a VM
* The kernel used by the VM’s OS
* The disk image containing the VM’s file system
* The price per hour

The gateway administrator provides the VM template information and can add, update and delete the templates at any time. Before starting an instance, scheduler gives the network configuration and address of the host. The MAC and IP addresses are also allocated. It also contains the path to the disk image storage.

1. **Distributed VM Management**: A distributed VM manager requests for VMs and gets their status and obtains a list containing the IP addresses of the VMs with secure shell (SSH) tunnels. The managers also obtains the template to be used by the VM, schedules the task for the VM, sets up the tunnel, and executes the tasks for each of the VM.
2. **Cloud Security and Trust Management**: Lacking of trust between service providers and clients has been a major problem in the field and much more since the advent of ecommerce. Cloud platforms are a concern for some users for lack of privacy protection, security assurance, and so on. All these can be solved with a technical approach.
   1. **Cloud Security Defence Strategies**:
   2. **Basic Cloud Security**: The basic cloud security enforcements are: security measures in data centers (like biometric readers, CCTV, man-traps etc.), fault-tolerant firewalls, IDS Intrusion Detection System), data encryption, strict password policies, and so on. The Figure 4.31 [1] shows the security measures at various levels:



* 1. **Cloud Defence Methods**: Virtualization enhances cloud security, but VMs add an additional layer of software that might lead to a single point of failure. So the VMs should be isolated in their deployment and work – the failure of one VM will not affect another. The Table 4.9 [1] below lists the protection schemes to secure public clouds and data centers.

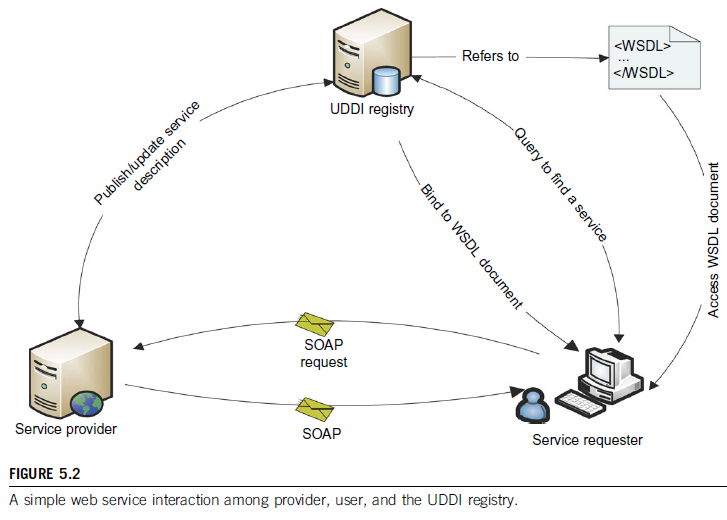


* 1. **Defence against DDoS Flooding attacks**: A DDoS defence system must be designed to cover multiple network domains in a cloud platform. The DDoS causes an abnormal surge in the network traffic by a hidden attacker which leads of the crash of the service/website or disk exhaustion or connection saturation.
  2. **Data and Software Protection Techniques**:

1. Data Integrity and Privacy Protection
2. Data Colouring and Cloud Watermarking
3. Data Lock-in Problems and Solutions: Once the data is moved into the cloud, users cannot easily extract their data and programs from the cloud servers to run on another platform. This is known as data lock-in. The solution possible here is to build platform-independent APIs where migration from one platform to another is easier.
4. **Service-Oriented Architecture**: SOA is concerned about how to design a software system that makes use of services or apps through their interfaces. These apps are distributed over the networks. The World Wide Web Consortium (W3C) defines SOA as a form of distributed architecture characterized by:

* **Logical View**: The SOA is an abstracted, logical view of actual programs, DBs etc. defined in terms of the operations it carries out. The service is formally defined in terms of messages exchanged between providers and requests.
* **Message Orientation**
* **Description Orientation**

1. **Services and Web Services**: In an SOA concept, the s/w capabilities are delivered & consumed through loosely coupled and reusable services using messages. ‘Web Service’ is a self-contained modular application designed to be used by other apps across the web. This can be seen in Figure 5.2 [1].



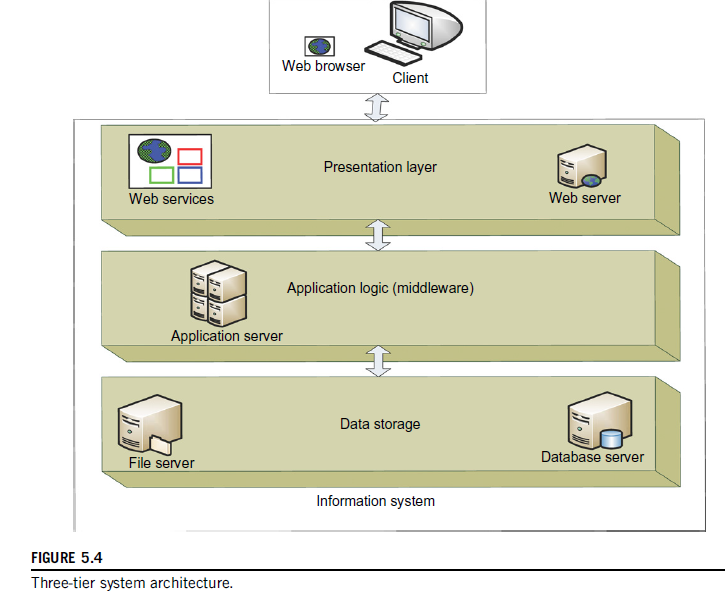
WSDL => Web Services Description Language

UDDI => Universal Description, Discovery and Integration

SOAP => Simple Object Access Protocol

* 1. **SOAP**: This provides a standard packaging structure for transmission of XML documents over various IPs. (HTTP, SMTP, FTP). A SOAP message consists of an **envelope** (root element), which itself contains a **header**. It also had a **body** that carries the payload of the message.
  2. **WSDL**: It describes the interface and a set of operations supported by a web service in a standard format.
  3. **UDDI**: This provides a global registry for advertising and discovery of web services by searching for names, identifiers, categories.
  4. Since SOAP can combine the strengths of XML and HTTP, it is useful for heterogeneous distributed computing environments like grids and clouds

1. **Enterprise Multitier Architecture**: This is a kind of client/server architecture application processing and data management are logically separate processes. As seen below in Figure 5.4 [1], it is a three-tier information system where each layer has its own important responsibilities.



**Presentation Layer**: Presents information to external entities and allows them to interact with the system by submitting operations and getting responses.

**Application Logic (Middleware)**: These consist of programs that implement actual operations requested by the client. The middle tier can also be used for user authentication and granting of resources, thus removing some load from the servers.

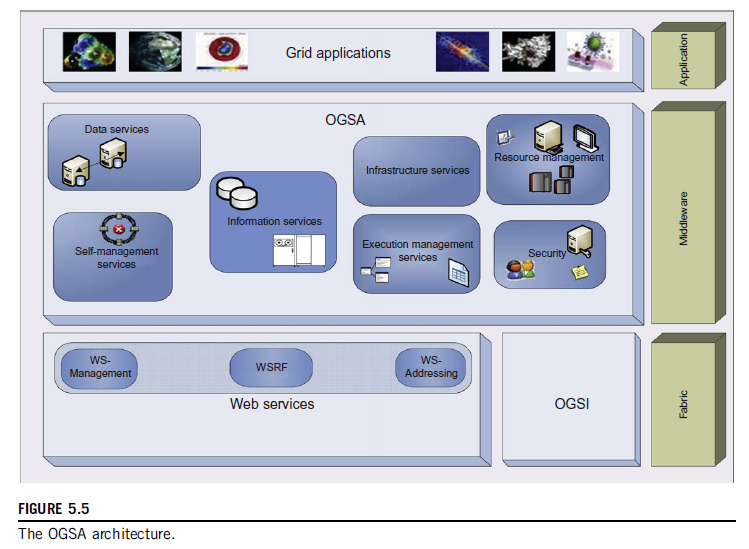
**Resource Management Layer (Data Layer)**: It deals with the data sources of an information system.

1. **OGSA Grid**: Open Grid Services Architecture is intended to

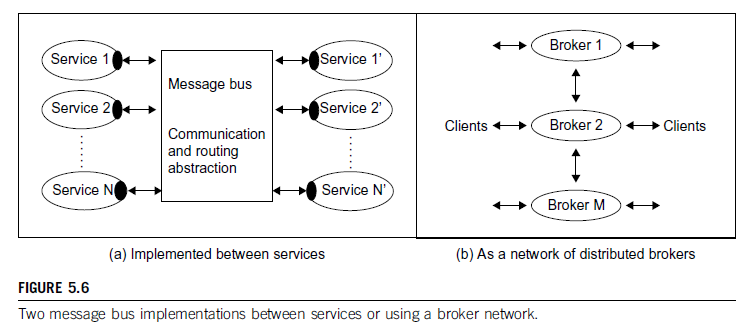
* Facilitate the usage of resources across heterogeneous environments
* Deliver best QoS
* Define open interfaces between diverse resources
* Develop inter-operable standards

1. OGSA architecture falls into seven broad areas, as shown in Figure 5.5 [1].

Infrastructure Services, Execution Management Services, Data Management Services, Resource Management Services, Security Services, Security Services, Information Services and Self-management Services (automation).



1. **Message-Oriented Middleware**:
   1. **Enterprise Bus**: Figure 5.6 [1]



Enterprise Service Bus (ESB) refers to the case where the bus supports the integration of many components in different styles as shown above. No source and destination channel is opened but only messages are induced from different services. A message bus is shown linking the services by receiving and sending messages but this methodology can work with any software or hardware.

* 1. **Queuing and Message Systems**: The best known standard in this field is the Java Message Service (JMS) which specifies a set of interfaces utilized in communication queuing systems. Advanced Message Queuing Protocol (AMQP) specifies a set of wire formats for communications.

**References**

1. Kai Hwang et al, Distributed and Cloud Computing – From Parallel Processing to the Internet of Things, Morgan Kaufmann, Elsevier, 2012.
2. <https://www.techopedia.com/definition/26598/elastic-computing-ec>
3. <https://en.wikipedia.org/wiki/Data_farming>
4. <http://www.webopedia.com/TERM/C/cloud_computing.html>
5. <http://searchnetworking.techtarget.com/definition/multiplexing>
6. <https://www.researchgate.net/post/What_is_data_coloring>
7. <https://www.techopedia.com/definition/27268/software-stack>