MOTHER TERESA WOMEN'S UNIVERSITY

DIRECTORATE OF DISTANCE EDUCATION

KODAIKANAL – 624 102

M.Sc. COMPUTER SCIENCE

(P.G.)

MATERIAL FOR

GRID AND CLOUD COMPUTING

SUBJECT CODE

Compiled by Dr. K.Kavitha

GRID AND CLOUD COMPUTING

UNIT I : Concepts and Architectures: Introduction - Parallel and Distributed Computing- Cluster computing – Grid Computing-anatomy and physiology of Grid – Review of web services – OGSA – WSRF.

UNIT II : **GRID Monitoring** : Grid Monitoring Architecture (GMA) – An overview of Grid Monitoring systems – Grid ICE – JAMM – MDs – Network Weather Service – R – GMA – other Monitoring systems Ganglia and Grid Mon

UNIT III: Grid Security and Resource Management: Grid Security – A Brief security primer – PRI – X509 Certificates – Grid security – Grid Scheduling and Resource management –Scheduling paradigms – Working principles of scheduling – A review of condor, SGE, PBS and LSF – Grid scheduling with QoS

UNIT IV: Examining the Value Proposition : Defining Cloud Computing, Assessing the Value Proposition, Understanding Cloud Architecture, Understanding Services and Applications by Type.

UNIT V : **Using Platforms :** Understanding Abstraction and Virtualization, Capacity, Planning, Exploring Platform as a Service, Using Google Web Services, Using Amazon Web Services, Using Microsoft Cloud Services.

TEXT BOOKS

1.The Grid : Core Technologies – Maozhen Li, Mark Baker, - John Wiley & Sum 2005
2. Cloud Computing Bible – Barrie Sosinky – Wiley Publishing Inc, 2011

REFERENCE BOOKS

1. Grid Computing – Josy Joseph & Craig Fellenstein – Pearson Education, 2004
2. The Little Book of Cloud Computing – A New Street Executive Summary
3. Lars Nielson – 2011 Edition

UNIT I

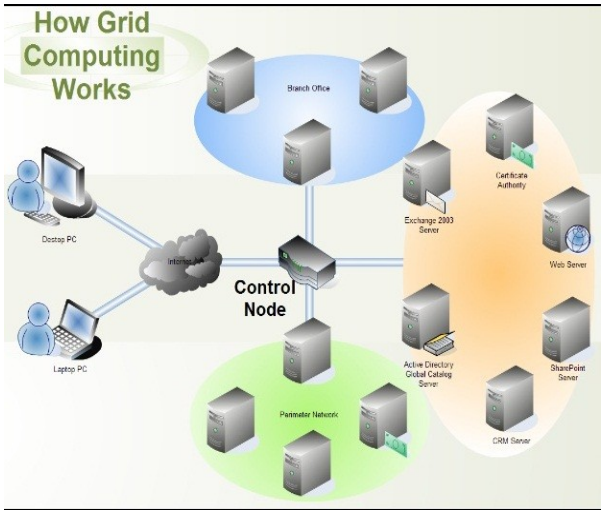
CONCEPTS AND ARCHITECTURE

Introduction

Grid computing is a group of computers physically connected (over a network or with Internet) to perform a dedicated tasks together, such as analyzing e-commerce data and solve a complex problem. **Grids are a form of "super virtual computer"** that solve a particular application. The grid size may vary from small to large enterprises network.

A *computing grid* is constructed with the help of grid middleware software that allows them to communicate. middleware is used to translates one node information passed stored or processed information to another into a recognizable format. **It is the form of "distributed computing" or "peer-to-peer computing".**

'Grid computing' is distinguished from the cluster computing, because in Grid computing each node has heterogeneous and geographically dispersed (such as a WAN) and its own resource manager and perform a different task and are loosely connected by the Internet or low-speed networks, but in cluster computing resources are managed in a single location (Like a LAN).



The **grid computing model** is a special kind of cost-effective distributed computing. In distributed computing, resources are shared by same network computers. In grid computing architecture, every computer in network turning into a powerful supercomputer that access to enormous processing power memory and data storage capacity.

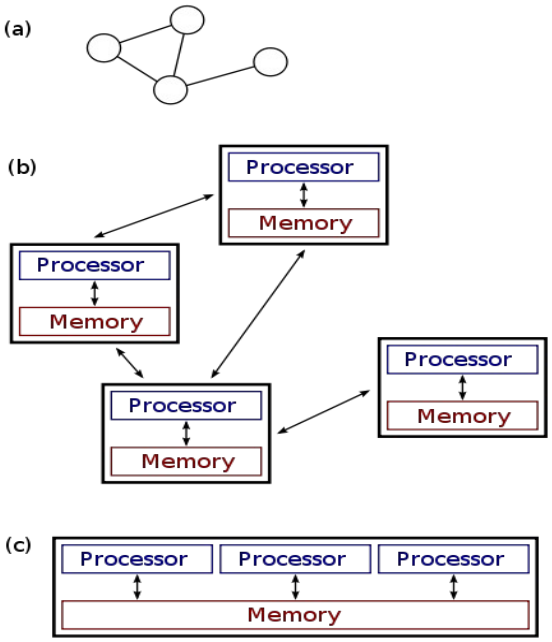
Grid computing solve challenging problems such as earthquake simulation and weather modeling.Grids computing is a way of using resources optimally inside an organization. Grid architecture can also be used for load balancing and redundant network connections.This Model use parallel processing software that divide a program among the many thousand computers and Collect and combine the results into a single solution.For the security reasons,grid computing is restricted within the same organisation.

Grid computing can be used in large networks where thousands of machines sit idle at any given moment. Even when a user is reading, it constitutes idle time. These idle power of computers can be used for large computational problems, these techniques is running in the background that is known as **cycle-scavenging**.

Parallel and Distributed Computing

Parallel Computing

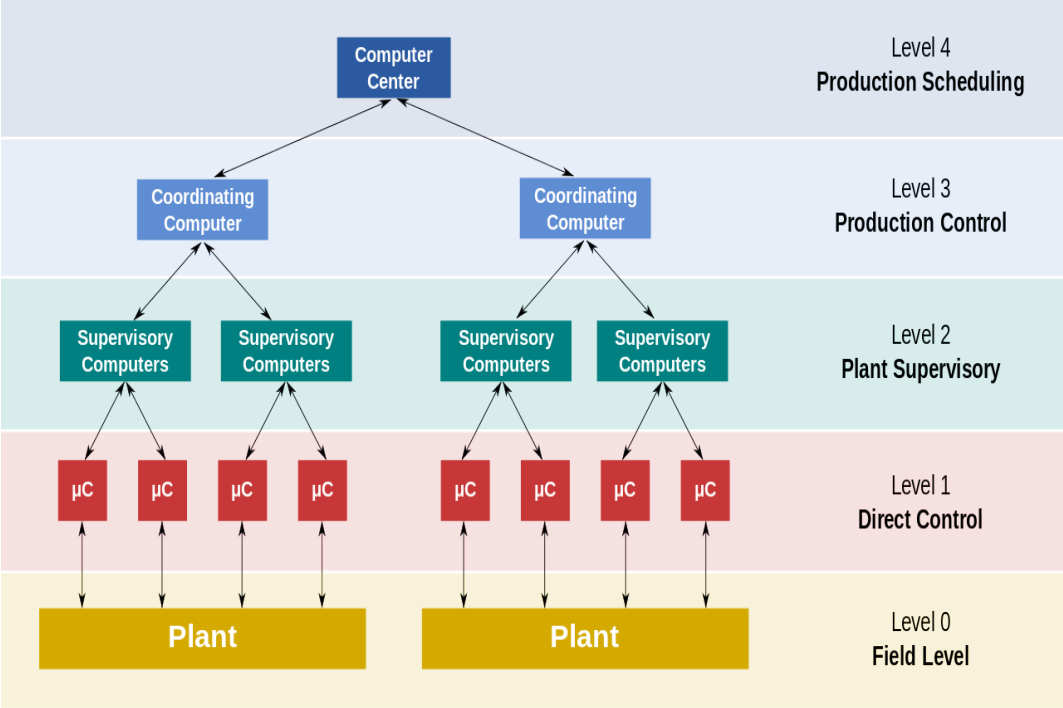
Parallel computing is also called **parallel processing**. There are multiple processors in parallel computing. Each of them performs the computations assigned to them. In other words, in parallel computing, multiple calculations are performed simultaneously. The systems that support parallel computing can have a shared memory or distributed memory. In shared memory systems, all the processors share the memory. In distributed memory systems, memory is divided among the processors.



There are multiple advantages to parallel computing. As there are multiple processors working simultaneously, it increases the CPUutilization and improves the performance. Moreover, failure in one processor does not affect the functionality of other processors. Therefore, parallel computing provides reliability. On the other hand, increasing processors is costly. Furthermore, if one processor requires instructions of another, the processor might cause latency.

Distributed Computing

Distributed computing divides a single task between multiple computers. Each computer can communicate with others via the network. All computers work together to achieve a common goal. Thus, they all work as a single entity. A computer in the distributed system is a node while a collection of nodes is a cluster.



There are multiple advantages of using distributed computing. It allows scalability and makes it easier to share resources easily. It also helps to perform computation tasks efficiently. On the other hand, it is difficult to develop distributed systems. Moreover, there can be network issues.

Difference Between Parallel and Distributed Computing

Definition

Parallel computing is a type of computation in which many calculations or execution of processes are carried out simultaneously. Whereas, a distributed system is a system whose components are located on different networked computers which communicate and coordinate their actions by passing messages to one another. Thus, this is the fundamental difference between parallel and distributed computing.

Number of computers

The number of computers involved is a difference between parallel and distributed computing. Parallel computing occurs in a single computer whereas distributed computing involves multiple computers.

Functionality

In parallel computing, multiple processors execute multiple tasks at the same time. However, in distributed computing, multiple computers perform tasks at the same time. Hence, this is another difference between parallel and distributed computing.

Memory

Moreover, memory is a major difference between parallel and distributed computing. In parallel computing, the computer can have a shared memory or distributed memory. In distributed computing, each computer has its own memory.

Communication

Also, one other difference between parallel and distributed computing is the method of communication. In parallel computing, the processors communicate with each other using a bus. In distributed computing, computers communicate with each other via the network.

Usage

Parallel computing helps to increase the performance of the system. In contrast, distributed computing allows scalability, sharing resources and helps to perform computation tasks efficiently. So, this is also a difference between parallel and distributed computing.

Conclusion

Parallel computing and distributed computing are two types of computations. The main difference between parallel and distributed computing is that parallel computing allows multiple processors to execute tasks simultaneously while distributed computing divides a single task between multiple computers to achieve a common goal.

Cluster computing

In its most basic form, a cluster is a system comprising two or more computers or systems **(called nodes)** which work together to execute applications or perform other

tasks, so that users who use them, have the impression that only a single system responds to them, thus creating an illusion of a single resource **(virtual machine)**. This concept is called transparency of the system. As key features for the construction of these platforms is included elevation : reliability, load balancing and performance.

Types of Clusters

High Availability **(HA)** and failover clusters, these models are built to provide an availability of services and resources in an uninterrupted manner through the use of implicit redundancy to the system. The general idea is that if a cluster node fail (failover), applications or services may be available in another node. These types are used to cluster data base of critical missions, mail, file and application servers.

Load balancing, this model distributes incoming traffic or requests for resources from nodes that run the same programs between machines that make up the cluster. All nodes are responsible to track orders. If a node fails, the requests are redistributed among the nodes available. This type of solution is usually used on farms of Web servers **(web farm)**.

HA & Load Balancing Combination, as its name says, it combines the features of both types of cluster, thereby increasing the availability and scalability of services and resources. This type of cluster configuration is widely used in web, email, news, or ftp servers.

Distributed Processing and Parallel Processing, this cluster model improves the availability and performance for applications, particularly large computational tasks. A large computational task can be divided into smaller tasks that are distributed around the stations **(nodes)**, like a massively parallel supercomputer. It is common to associate this type of Beowulf cluster at NASA project. These clusters are used for scientific computing or financial analysis, typical for tasks requiring high processing power.

Reasons to Use a Cluster

Clusters or combination of clusters are used when content is critical or when services have to be available and / or processed as quickly as possible. Internet Service Providers **(ISPs)** or e-commerce sites often require high availability and load balancing in a scalable manner. The parallel clusters are heavily involved in the film industry for rendering high quality graphics and animations, recalling that the Titanic was rendered within this platform in the Digital Domain laboratories. The Beowulf clusters are used in science, engineering and finance to work on projects of protein folding, fluid dynamics, neural networks, genetic analysis, statistics, economics, astrophysics among others. Researchers, organizations and companies are using clusters because they need to increase their scalability, resource management, availability or processing to supercomputing at an affordable price level.

High-Availability (HA) or Clusters Failover

The computers have a strong tendency to stop when you least expect, especially at a time when you need it most. It is rare to find an administrator who never received a phone call in the middle of the morning with the sad news that the system was down, and you have to go and fix the problem.

High Availability is linked directly to our growing dependence on computers, because now they have a critical role primarily in companies whose main functionality is exactly the offer of some computing service, such as e-business, news, web sites, databases, among others.

A High Availability Cluster aims to maintain the availability of services provided by a computer system by replicating servers and services through redundant hardware and software reconfiguration. Several computers acting together as one, each one monitoring the others and taking their services if any of them will fail. The complexity of the system must be software that should bother to monitor other machines on a network, know what services are running, those who are running, and what to do in case of a failure. Loss in performance or processing power are usually acceptable, the

main goal is not to stop. There are some exceptions, like real-time and mission critical systems.Fault tolerance is achieved through hardware like raid systems, supplies and redundant boards, fully connected network systems to provide alternative paths in the breaking of a link.

Cluster Load Balancing

The load balancing among servers is part of a comprehensive solution in an explosive and increasing use of network and Internet. Providing an increased network capacity, improving performance. A consistent load balancing is shown today as part of the entire Web Hosting and e-commerce project. But you cannot get stuck with the ideas that it is only for providers, we should take their features and bring into the enterprise this way of using technology to heed internal business customers.

The cluster systems based on load balancing integrate their nodes so that all requests from clients are distributed evenly across the nodes. The systems do not work together in a single process but redirecting requests independently as they arrive based on a scheduler and an algorithm.This type of cluster is specially used by e-commerce and Internet service providers who need to resolve differences cargo from multiple input requests in real time. Additionally, for a cluster to be scalable, must ensure that each server is fully utilized.

When we do load balancing between servers that have the same ability to respond to a client, we started having problems because one or more servers can respond to requests made and communication is impaired. So we put the element that will make balancing between servers and users, and configure it to do so, however we can put multiple servers on one side that, for the customers, they appear to be only one address. A classic example would be the Linux Virtual Server, or simply prepare a DNS load balancer. The element of balance will have an address, where customers try to make contact, called Virtual Server (VS), which redirects traffic to a server in the server pool. This element should be a software dedicated to doing all this management,

or may be a network device that combines hardware performance and software to make the passage of the packages and load balancing in a single device.

We highlight some key points for an implementation in an environment of success with load balancing on the powerful dedicated servers:

The algorithm used for load balancing, taking into consideration how balancing between servers is done and when a client makes a request to the virtual address **(VS)**, the whole process of choosing the server and the server response must occur transparent and imperceptible to the user mode as if no balancing.

Create a method to check if the servers are alive and working, vital if the communication is not redirected to a server that has just had a failure **(keepalive)**.

A method used to make sure that a client accessing the same server when you want.

Load balancing is more than a simple redirect client traffic to other servers. For proper implementation, the equipment you will need to have balancing characteristics as permanent communication check, verification of servers and redundancy. All of these items are necessary to support the scalability of the volume of traffic on the networks without eventually become a bottleneck or single point of failure.

Algorithms for balancing is one of the most important factors in this context, then we will explain three basic methods :

Least Connections

This technique redirects the requests to the lowest based on the number of requests / server connections. For example, if server 1 is currently handling 50 requests / connections, and server 2 controls 25 requests / connections, the next request / connection will be automatically directed to the second server, since the server currently has fewer requests / connections active.

Round Robin

This method uses the technique of always direct requests to the next available server in a circular fashion. For example, incoming connections are directed to the server 1, server 2 and then finally server 3 and then the server 1 returns.

Weighted Fair

This technique directs the requests to the load based on the requests of each and the responsiveness of the same **(performance)** For example, if the servers server 1 is four times faster in servicing requests from the server 2, the administrator places a greater burden of work for the server 1 to server 2.

Combined Cluster High Availability and Load Balancing

This combined solution aims to provide a high performance solution combined with the possibility of not having critical stops. This combined cluster is a perfect solution for ISPs and network applications where continuity of operations is very critical.

Some features of this platform :

- Redirection of requests to node failures reservations for us ;
- Improved quality of service levels for typical network applications ;
- Transparent integration for stand-alone applications and non-clustered together in a single virtual network ;
- Provide a highly scalable architecture framework.

Grid Computing- Anatomy and physiology of Grid

A domain-specific software architecture (DSSA) represents an effective, generalized, reusable solution to constructing software systems within a given application domain. In this paper, we revisit the widely cited DSSA for the domain of grid computing. We have studied systems in this domain over the last ten years. During this time, we have repeatedly observed that, while individual grid systems are widely used and deemed successful, the grid DSSA is actually underspecified to the point where providing a precise answer regarding what makes a software system a grid system is nearly

impossible. Moreover, every one of the existing purported grid technologies actually violates the published grid DSSA. In response to this, based on an analysis of the source code, documentation, and usage of eighteen of the most pervasive grid technologies, we have significantly refined the original grid DSSA. We demonstrate that this DSSA much more closely matches the grid technologies studied. Our refinements allow us to more definitively identify a software system as a grid technology, and distinguish it from software libraries, middleware, and frameworks.

Review of web services

Yet another distributed middleware technology (just like CORBA, RMI, EJBs, etc.) What sets it apart: Clean separation between interface (what the service does) and implementation (how it does it). Based on standard interoperable languages (XML). Widespread use and abundant software available

A web service 'publishes' the operations it is capable of performing through its interface. The interface is written in a specific language. The implementation and the interface are kept separate. The implementation is done using a programming language (Java, C++...)Both the implementation and the interface are placed in a running environment, which is in charge of handling all incoming calls, crafting the responses, etc

Separation of interface and implementation Interface is defined in an XML language called WSDL (Web Services Description Language). WSDL is language and platform-neutral, and allows the interface to be defined separately from the particular transport protocol or data encoding used in the actual message passing. e.g. A GNU/Linux client can access a web service in a Windows server using protocol A, while a different client using Solaris might use protocol B

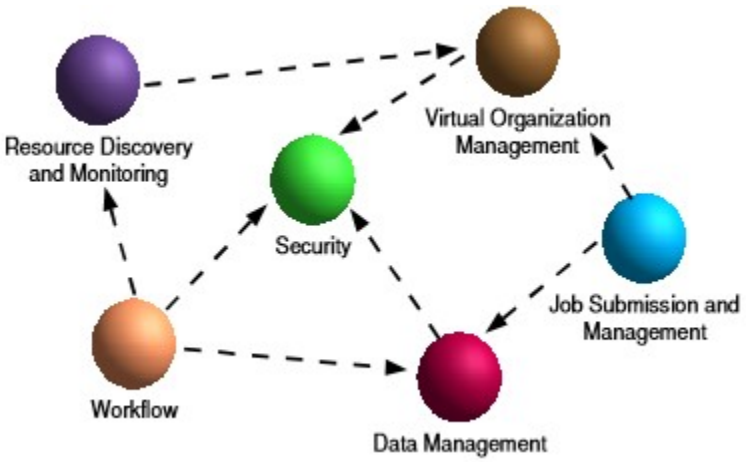
We can use web services to build SOAs (ServiceOriented Architectures) Architecture in which an application uses several independent services (or loosely coupled services) that cooperate to undertake a common task. “loosely coupled”: A change in the

implementation of one service doesn't affect the other services. SOAs improve interoperability and reusability.

Common interface language (WSDL) allows: Virtualization: from a pool of services (with the same interface) I can access any service in the same fashion regardless of vendor, platform, etc. Dynamic service indexing and discovery.Dynamic access to services.

OGSA

The Open Grid Services Architecture (OGSA), set to become the standard architecture for most grid applications, depends on Web Services as the underlying middleware. OGSA first spawned the Open Grid Services Infrastructure which, despite improving Web Services in several ways, failed to converge with existing Web Services standards. The Web Services Resource Framework (WSRF), improves on OGSI and will eventually replace it. The presentation will cover the evolution and current state of OGSA, OGSI, WSRF, and the next version of the Globus Toolkit (GT4), which will be the first implementation of the WSRF specification.A Grid systems is built with several different subsystems. For example...



The Open Grid Services Architecture (OGSA) aims to standardize the different services that are commonly found in a Grid application. Job management, security, VO management, data management, workflow, deployment, etc. OGSA was introduced in

the classic paper "Physiology of the Grid" the actual standardization process is being carried out by the Global Grid Forum (GGF)

"A set of core interfaces and behaviors that address key concerns in Grid systems." Each particular service can be accessed in the same fashion, regardless of vendor, organization, internal implementation, etc.

Web Services Resource Framework (WSRF)

The Web Services Resource Framework (WSRF) defines a generic and open framework for modeling and accessing stateful resources using Web Services. This includes mechanisms to describe views on the state, to support management of the state through properties associated with the Web Service, and to describe how these mechanisms are extensible to groups of Web Services. It defines the means by which:

- Web Services can be associated with one or more stateful resources (named, typed, state components).
- Service requestors access stateful resources indirectly through Web Services that encapsulate the state and manage all aspects of Web Service based access to the state.
- Stateful resources can be destroyed, through immediate or time based destruction.
- The type definition of a stateful resource can be associated with the interface description of a Web Service to enable well-formed queries against the resource via its Web Service interface.
- The state of the stateful resource can be queried and modified via Web Service message exchanges.
- Endpoint references to Web Services that encapsulate stateful resources can be renewed when they become invalid, for example due to a transient failure in the network.
- Stateful resources can be aggregated for domain-specific purposes.

Additional related specifications that have been developed that will be considered by OASIS for the WSRF. These were developed by Computer Associates, Fujitsu, Globus Alliance, Hewlett-Packard, IBM, and the University of Chicago. The motivation for these specifications is that while Web service implementations typically do not maintain state information during their interactions, their interfaces must frequently allow for the manipulation of state, that is, data values that persist across and evolve as a result of Web service interactions

WS-ResourceProperties:

This defines how the data associated with a stateful resource can be queried and changed using Web Services technologies. This allows a standard means by which data associated with a WS-Resource can be accessed by clients. The declaration of the WS-Resource's properties represents a projection of or a view on the WS-Resource's state. This projection represents an implied resource type which serves to define a basis for access to the resource properties through Web service interfaces

UNIT II

GRID MONITORING

The INFN-**Grid**` infrastructure includes several **monitoring** systems for different purposes and granularity. Monitor activity can focus on resources status and services availability at each site or across the **grid** as a whole and display useful data aggregated per site, service and VO.

Characteristics for Grid Monitoring:

- Scalable
- Dynamic
- Robust
- Flexible
- Should be integrated with other Grid Technologies and middleware (security infrastructure, resource brokers, schedulers, ...)
- Must Perform

Grid Monitoring Architecture (GMA)

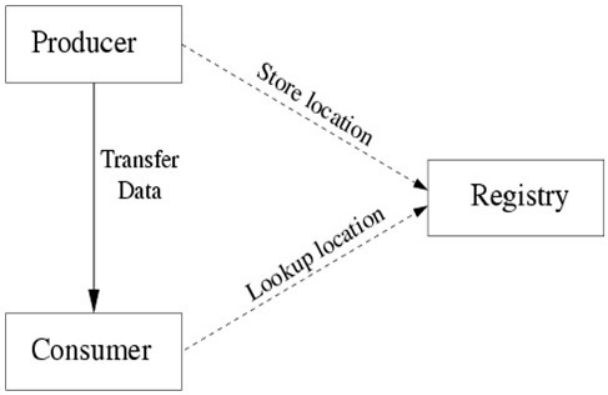


Figure: Grid Monitoring Architecture

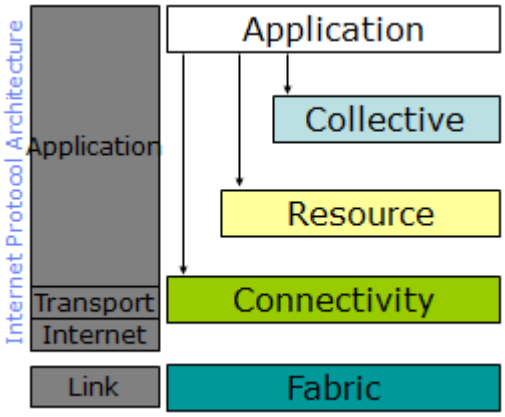
- **Descriptive**
  - Provide a common vocabulary for use when describing Grid systems

- **Guidance**
  - Identify key areas in which services are required
- **Prescriptive**
  - Define standard protocols and APIs to facilitate creation of interoperable Grid systems and portable applications
- A grid architecture identifies fundamental system components, specifies the purpose and function of these components, and indicate how these components interact.
- Grid's protocols allow **VO** users and resources to negotiate, establish, manage and exploit sharing relationships.
  - Interoperability a fundamental concern
  - The protocols are critical to interoperability
  - Services are important
  - We need to consider APIs and SDKs

  VO: Virtual Organization.

The components are

- numerous
- owned and managed by different, potentially mutually distrustful organisations and individuals
- may be potentially faulty
- have different security requirements and policies
- heterogeneous
- connected by heterogeneous, multilevel networks
- have different resource management policies
- are likely to be geographically separated



Fabric Layer

- Just what you would expect: the diverse mix of resources that may be shared
  - Individual computers, Condor pools, file systems, archives, metadata catalogs, networks, sensors, etc., etc.
- Defined by interfaces, not physical characteristics

Connectivity Layer

- Communication
  - Internet protocols: IP, DNS, routing, etc.
- Security: Grid Security Infrastructure (GSI)
  - Uniform authentication, authorization, and message protection mechanisms in multi-institutional setting
  - Single sign-on, delegation, identity mapping
  - Public key technology, SSL, X.509, GSS-API
  - Supporting infrastructure: Certificate Authorities, certificate & key management.

Resource Layer

- The architecture is for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources.
  - Information Protocols (inform about the structure and state of the resource)
  - Management Protocols (negotiate access to a shared resource)

- Grid Resource Allocation Mgmt (GRAM)
  - Remote allocation, reservation, monitoring, control of compute resources
- GridFTP protocol (FTP extensions)
  - High-performance data access & transport
- Grid Resource Information Service (GRIS)
  - Access to structure & state information
- Network reservation, monitoring, control
- All built on connectivity layer: GSI & IP

Collective Layer

- Coordinating multiple resources
- Contains protocols and services that capture interactions among a collection of resources
- It supports a variety of sharing behaviours without placing new requirements on the resources being shared
- Sample services: directory services, co-allocation, brokering and scheduling services, data replication services, workload management services, collaboratory services
- Index servers aka metadirectory services
  - Custom views on dynamic resource collections assembled by a community
- Resource brokers (e.g., Condor Matchmaker)
  - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management servicesEtc.

Application Layer

- There are user applications that operate within the VO environment
- Applications are constructed by calling upon services defined at any layer

- Each of the layers are well defined using protocols, provide access to services
- Well-defined APIs also exist to work with these services

An overview of Grid Monitoring systems

A grid environment is potentially a complex globally distributed system that involves large sets of diverse, geographically distributed components used for a number of applications. The goal of grid monitoring is to measure and publish the state of resources at a particular point in time. This includes software such as applications, services, processes and operating systems. Host hardware such as CPUs, disks memory and sensors.Networks such as routers, switches, bandwidth and latency.

Grid ICE

The **Internet Communications Engine**, or **Ice**, is an open-source RPC framework developed by ZeroC. It provides SDKs for C++, C#, Java, JavaScript, MATLAB, Objective – C, PHP, Python, Ruby, Swift and TypeScript, and can run on various operating systems, including Linux, Windows, macOS, iOS and Android.

JAMM

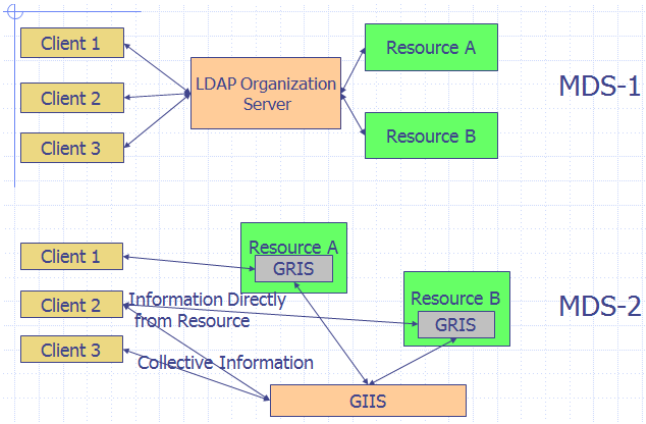
Java Agents for Monitoring and Management use a collection of software agents that can perform various administrative tasks in a monitoring network

MDS

MDS (Metacomputing Directory Service)

- Uses LDAP (Lightweight Directory Access Protocol). It Provides uniform means of querying system information from a rich variety of system components
- uniform namespace for resource information across a system that may involve many organizations.
- MDS provides directory services for Grids using the Globus Toolkit.
- Provides a mechanism for publishing and discovering resource stats and configuration infoBased on OpenLDAP. Decentralized and Scalable

- Security provided by combining GSI (Grid Security Infrastructure) with OpenLDAP ACLs
- GRIS runs on each resource and provides resource specific info.
- GRIS invokes Info Providers to collect Resource data.
- Each GRIS supports multiple Info Providers.
- Data gets cached for a period of time (Cachettl parameter)
- GRIS registers with one or more GIIS to form a hierarchy.



- Service
  - GRIS Grid Resource Information Service
  - GIIS Grid Index Information Service
- Protocol
  - GRRP Grid Resource Registration Protocol
  - GRIP Grid information Protocol

Both GRRP and GRIP are extended from LDAP (Light Weight Directory Access Protocol)

Registrar configuration (GIIS)

Every Registrar (GIIS) determines whether to accept incoming registration requests (grid-info-site-policy.conf)

Registrant (GRIS/GIIS)

Every Registrant determines which GIIS's will register to (grid-info-resource-register.conf) and which providers will be available to send data to the GIIS's this GRIS is registered (grid-info-resource-ldif.conf).

regperiod:

How often this GRIS will send a message to GIIS announcing its existence

ttl:

How long the registration info will be good for, before assuming that this GRIS is no longer available (typically ttl=2xregperiod)

cachettl:

How long info from this GRIS will be kept in cache.

bindmethod:

What method will be used for mutual authentication

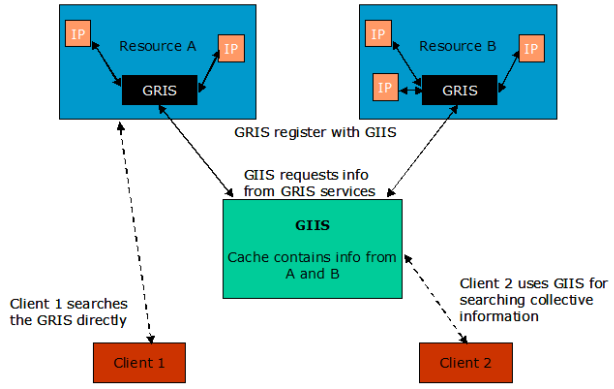


Figure: MDS GIIS

MDS data can be accessed with a wide range of utilities:

Command Line Tools:

grid-info-search: a grid enabled ldapsearch

Programmatically:

LDAP Client API for Java, Python and Perl

Java uses the JNDI package for accessing LDAP directories.

Java CoG uses it.

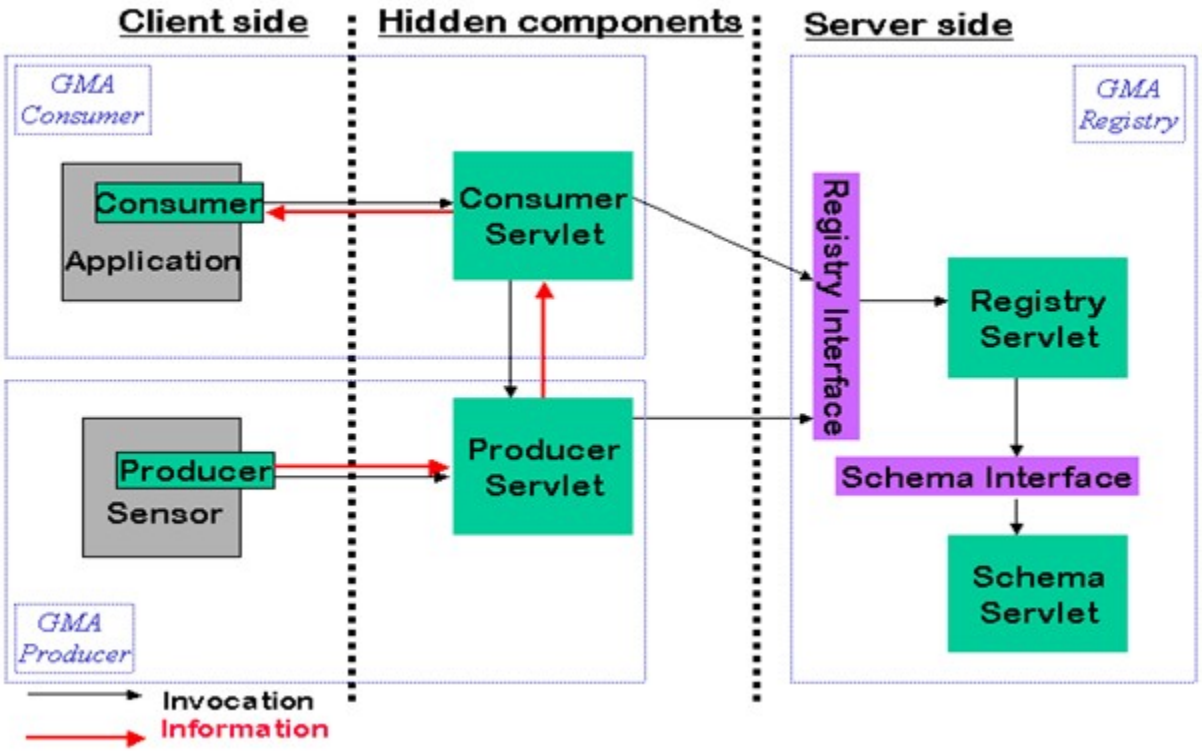
Various LDAP/Web Browsers

The Grid technology Repository is a good place to look for MDS info providers and clients.

Network Weather Service.

R –GMA

- R-GMA is used in the European Data Grid Project
- Based on a Relational Data Model (uses Individual RDBMS and SQL statements to provide the functionality outlined in GMA)
- Uses Java Servlets (tomcat). Moving to Web Services
- Can be used as a replacement to MDS (tools are provided to invoke MDS Info Providers)
- Nagios is used for graphs and notification
- Clients: R-GMA Browser (Java Graphical display Tool), command line tool (Python) and an API for programmatic access.



Other Monitoring Systems

Ganglia

Ganglia, by comparison, was born at Berkeley, in an academic, Grid-computing culture. The HPC-centric admin and engineers who designed it were used to thinking about massive, parallel applications, so even though the designers of other monitoring systems looked at tens of thousands of hosts and saw a problem, it was natural for the Berkeley engineers to see those same hosts as the solution.

Ganglia's metric collection design mimics that of any well-designed parallel application. Every individual host in the grid is an active participant, and together they cooperate, organically distributing the workload while avoiding serialization and single points of failure. The data itself is replicated and dispersed throughout the Grid without incurring a measurable load on any of the nodes. Ganglia's protocols were carefully designed, optimizing at every opportunity to reduce overhead and achieve high performance.

This cooperative design means that every node added to the network only increases Ganglia's polling capacity and that the monitoring system stops scaling only when your network stops growing. Polling is separated from data storage and presentation, both of which may also be redundant. All of this functionality is bought at the cost of a bit more per-host configuration than is employed by other, more traditional monitoring systems.

Grid Mon

GridMon Service Manager provides a proactive, real-time monitoring solution that watches over servers, workstations and network devices to ensure maximum availability and performance.

A live control panel is accessible 24x7 from an office computer or mobile phone. Gridmon Service Manager is built using high availability infrastructure to deliver information 24 hours a day.

GridMon Service Manager consolidates across multiple sites to give an end-to-end view of a corporate IT infrastructure. It is scalable, cross-platform and has the flexibility to support customised monitoring.

UNIT III

GRID SECURITY AND RESOURCE MANAGEMENT:

The **Grid Security Infrastructure** (GSI), formerly called the **Globus Security Infrastructure**, is a specification for secret, tamper-proof, delegable communication between software in a grid computing environment. Secure, authenticatable communication is enabled using asymmetric encryption.

GRID Security :

- To protect application and data from the owner/administrator of the system and
- to protect local programs and data on the system on which another remote user's process may also be getting executed
- Data, Code and resources accepted after proper *authentication*

Integrity of data and code is required to be verified.

Security Requirements

1. AUTHENTICATION

Verifying a principal's claimed identity.

Principal:- a user logged on a remote system or

- a local user logged on the server or

- the server itself

Two - step process: - User Name

- Password

(check: - something you know (common)

- Something you have
- Something you are
- what you do (key-stroke patterns)
- where you are)

GRID: Mutual authentication required for user and serviceprovider.(The resources and data being provided by a server could be provided by an attacker.)

Data origin authentication: To determine whether a program was modified or sent by an attacker to compromise the server.

Data origin authentication**:** does not inform the data wasrecently sent by the principal.

Delegation of Identity: When an application or a process is authorized to assume the identity of a different principal.

A Brief security primer

PRI

A report entitled *Stepping Up Governance on Cyber Security* by the Principles for Responsible Investment (PRI) Association, found only 15% of public companies explicitly outlined how they trained staff and only 17% said they conduct regular audits. The research was conducted with a sample of 100 publicly listed companies.

PRI researchers also found that nearly 60% of those companies polled failed to indicate how the board or a board sub-committee was responsible for cyber security issues and less than a third (31%) documented how they employed internal expertise or external consultants.

X509 Certificates

Most plug-ins of gPlazma support X.509 certificates for authentication and authorization. X.509 certificates are used to identify entities (e.g., persons, hosts) in the Internet. The certificates contain a DN (Distinguished Name) that uniquely describes the entity. To give the certificate credibility it is issued by a CA (Certificate Authority) which checks the identity upon request of the certificate (e.g., by checking the persons id). For the use of X.509 certificates with dCache your users will have to request a certificate

from a CA you trust and you need host certificates for every host of your dCache instance.

Grid Scheduling and Resource management

Grid Scheduling

Resource management

- Members should be trustful and trustworthy.
- Sharing is conditional.
- Should be secure.
- Sharing should be able to change dynamically.
- Discovery and registering of resources.
- Can be peer to peer or client/server.
- Same resource may be used in different ways.
- Well defined architecture and protocols.

Grid is affected by continuous innovations:

- Schemas conversion technologies.
- Allow data to move between different systems technologies.
- Intercommunication between different network domains.
- Services on the net need specific resources requirements
- Grid need to handle resources in more dynamic way
- Grid Services will require to coordination and orchestration of resources at run time
- QoS Aware

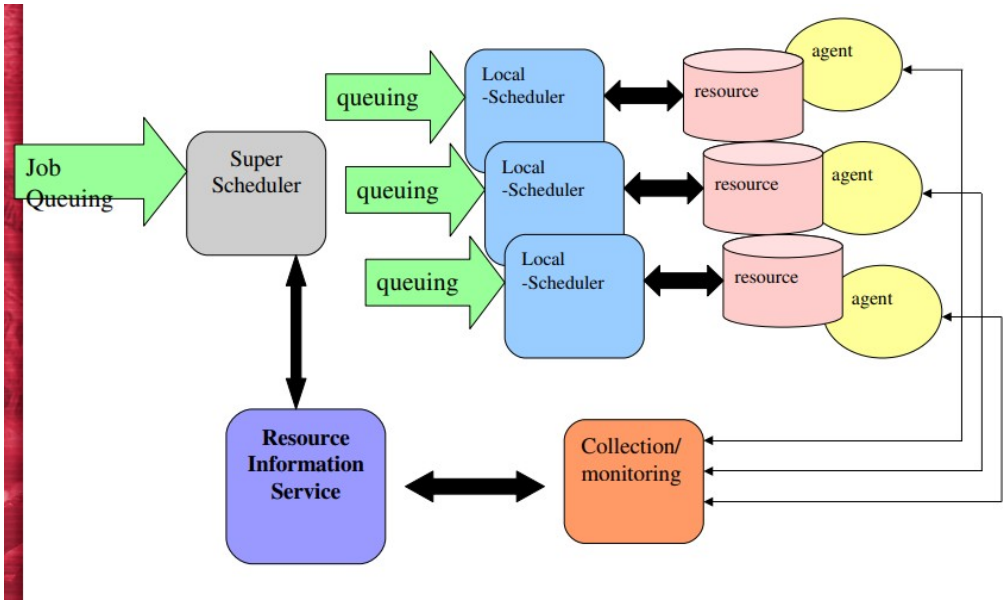


Figure: Resource Management Architecture

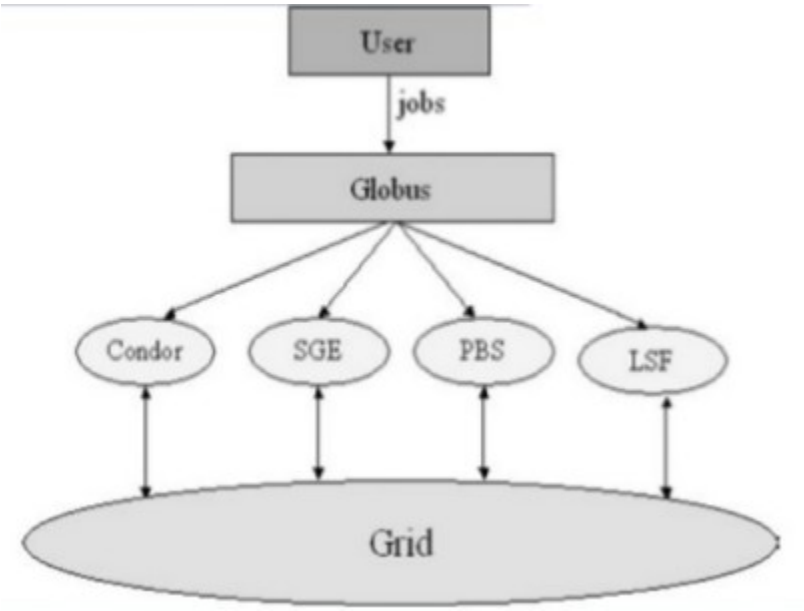
Resource Management Challenges:

- Satisfactory endtoend performance through multiple domains
- Availability of computational resources
- Handle of conflicts between common resources demand
- Faulttolerance
- Interdomain compatibility (P2P)

Scheduling paradigms

- A finite set of n jobs
- Each job consists of a chain of operations
- A finite set of m machines
- Each machine can handle at most one operation at a time
- Each operation needs to be processed during an uninterrupted period of a given length on a given machine
- Purpose is to find a schedule, that is, an allocation of the operations to time intervals to machines, that has minimal length

Working principles of scheduling



A review of condor

- CondorG:
- Condor is a highthroughput scheduler.
- CondorG uses Globus Toolkit libraries for:
    - » Security (GSI)
    - » Managing remote jobs on Grid (GRAM)
    - » File staging & remote I/O (GSIFTP)
    - » Grid job management interface & scheduling
- Supports single or highthroughput apps on Grid
- Personal job manager which can exploit Grid resources,

SGE

Grid Engine which is often called Sun Grid Engine (SGE) is a software classic. It is a batch jobs controller like batch command on steroids, not so much a typical scheduler. At one point Sun open source the code, so open source version exists. It is the most powerful (albeit specialized) open source scheduler in existence. This is one of most valuable contributions of Sun to open source community as it provides industrial strength batch scheduler for Unix/Linux.

SGE is an very powerful and flexible batch system, that probably should became standard Linux subsystem replacing or supplementing very basic batch command. It is available in several Linux distributions such as Debian and Ubuntu as installable software package from the main depository. It is available from "other" depositories for CentOS, RHEL and Suse.SGE has many options to help effectively use all of computational resources -- grid consisting of head node and computational nodes, each with certain number of cores (aka slots).

PBS

Portable Batch System (or simply **PBS**) is the name of computer software that performs job scheduling. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources. It is often used in conjunction with UNIX cluster environments.

LSF

Scheduler software exists simply because the amount of jobs that users wish to run on a cluster at any given time is usually greatly in excess of the amount of resources available. This means that the **scheduler** must queue jobs and **work** out how to run them efficiently. A job scheduler, or "batch" scheduler, is a tool that manages how user jobs are queued and run on a set of compute resources. In the case of LOTUS the compute resources are the set of compute nodes that make up the LOTUS hardware. Each user can submit jobs to the scheduler which then decides which jobs to run and where to execute them. The scheduler manages the jobs to ensure that the compute resources are being used efficiently and that users get appropriate access to those resources.

Grid scheduling with QoS

- A broad definition: methods for *differentiating* traffic and services
- To some, introducing an element of predictability and consistency into a highly variable best-effort network

- To others, obtaining higher network throughput while maintaining consistent behavior
- Or, offering resources to high-priority service classes at the expense of lower-priority classes (conservation law)
- Or, matching network resources to application demands
- No longer enough for an ISP to keep traffic flowing, links up, routing stable --- offer QoS to be more competitive
- QoS is becoming more cost-effective with improved implementation of differentiation tools (classification, queue-manipulation, …), more reliable tools for measuring delivered QoS
- Hard to dimension the network as it is becoming increasingly hard (if not impossible) to predict traffic patterns (e.g., 80 local/20 remote rule no longer reliable, now mostly reversed 25/75)
- If you throw more bandwidth, there will be more demand (a “vicious cycle”)
- Real-time: voice, video, emergency control, stock quotes ...
- Non-real-time (or best-effort): telnet, ftp, …

Real-time:

- hard with deterministic or guaranteed QoS: no loss, packet delay less than deadline, difference in delays of 2 packets less than jitter bound, …

- soft with statistical or probabilistic QoS: no more than x% of packets lost or experience delay greater than deadline, …

Best-effort do not have such timing constraints

UNIT IV

EXAMINING THE VALUE PROPOSITION

Defining Cloud Computing

The basic definition of Cloud computing is on-demand computing. Cloud computing is defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications.

The cloud concept is built on layers, each providing a distinct level of functionality. This stratification of the cloud's components has provided a means for the layers of cloud computing to becoming a commodity just like electricity, telephone service, or natural gas. The commodity that cloud computing sells is computing power at a lower cost and expense to the user. Cloud computing is poised to become the next mega-utility service.

From an organization point of view it is cost effective as the user (organization) has to pay only what they use and simultaneously they also get more resources easily when required. This is in contrast to the traditional CAPEX model where the organization invests in getting servers, space, maintenance initially and then gets a return over its investment over a period of time.

Cloud computing shares characteristics with:

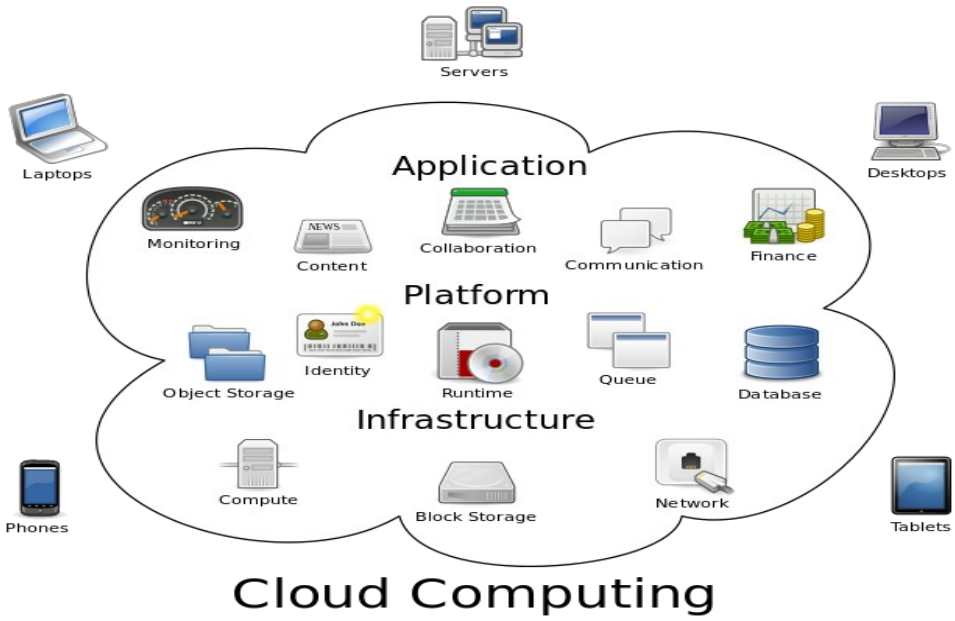
Client–server model — *Client–server computing* refers broadly to any distributed application that distinguishes between service providers (servers) and service requestors (clients).

Grid computing — "A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks."

Mainframe computer — Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as: census; industry and consumer statistics; police and secret intelligence services; enterprise resource planning; and financial transaction processing.

Utility computing — The "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity."

Peer-to-peer — A distributed architecture without the need for central coordination. Participants are both suppliers and consumers of resources (in contrast to the traditional client–server model).



Assessing the Value Proposition

Cloud Foundry is an open application Platform as a Service (PaaS) developed under an open source license. In other words, **Cloud Foundry** is a system to easily deploy, operate and scale stateless applications which are written in any programming language or framework.

Understanding Cloud Architecture

Cloud Computing Model

The cloud computing model is comprised of a **front end** and a **back end**. These two elements are connected through a network, inVirtualization software makes it possible to run multiple operating systems and multiple applications on the same server at the same time.

Cloud computing is a pay-as-you-go on-demand architecture of computer, networking, storage, applications, power, and other applications resources. In cloud computing architecture, all information is accessed via a public network (internet) from a particular geographical location which is called a cloud. This infrastructure is managed by some bodies, organizations which are called Managed Service Provider (MSP), Cloud Service Provider (CSP). A company can store and access all of their information on a cloud platform which can be located on any geographical location.

Cloud computing architecture has many advantages, some of them are as follows: Developing and deploying new apps and services

- Eliminate cost of buying expensive hardware and software
- Self-service portal for agility and speed
- Ability to scale-up and scale-out easily
- Remove the requirements of racking and stacking hardware to improve productivity
- Run your applications on secure data centers worldwide on latest computing hardware which improve performance
- Easy data backup & recovery, and business continuity which improves reliability

Cloud Architectures

Cloud computing architecture are the resources which delivered by server-based applications using public networks. As per (NIST Cloud Computing Program (NCCP), 2018), following five qualities that explain cloud computing:

- on-demand self-service
- broad network access
- resource pooling
- rapid elasticity or expansion
- measured service

Cloud computing architecture may have divided into two parts:

a. Front End

Front-end cloud computing architecture is a visible interface that can be encounter through a web-enabled interface. But we should remember that all computing systems are not using the same user interface.

b. Back End

Backend cloud computing architecture is composed of the physical infrastructure of bare-metal servers and virtual machines (VMs), data storage systems, a security system or mechanism, and services, which are used for the conformance of a deployment model to providing services to their clients. The back end architecture of cloud computing is also responsible to provide security, traffic control, and network protocols who connect the computers all over the globe.

In short, we can say that the front end is the part which actually seen by the customer or user, and the back end part is the computing which acts behind the scene and is responsible to make an available front-end for the users.

Cloud Computing Services

Cloud computing architectures mainly fall into three cloud computing services: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). These services are sometimes to referred to as cloud computing stack, as they build on top of one another.

Infrastructure as a Service (IaaS)

IaaS provides businesses a web architecture such computing servers, data storage space, and network connectivity without purchasing and managing humongous internet infrastructure themselves. This cloud computing service allows organizations to grow and develop on demand, the more you need, the more you can demand.

Platform as a Service (PaaS)

PaaS is the hardest part of cloud computing services to explain. It's a cloud computing service that allows developers to build their apps and services using a platform and environment on the internet. PaaS services can be accessed by using a web browser as these are hosted on the cloud platform.By using a PaaS service, a user can initiate his/her app without any stress and it can be scalable according to requirements.

Software as a Service (SaaS)

SaaS is comparatively matured service which allows the cloud to be leveraged for software design and architecture, reduce the maintenance burden, support, and run the apps on computers which belongs to the vendor. SaaS removes the requirements for businesses to run their apps on their own infrastructure. This capability removes requirements of expensive hardware purchasing, maintenance, installation, support, and software licensing requirements. Gmail and Salesforce are the major examples of SaaS that run on cloud computing architecture.

Cloud Formation or Cloud Deployment Models

There are four types of cloud deployment models/formations : private (on premise), public, hybrid and community.

Public clouds are available to the general public or a large industry group and are owned and provisioned by an organization selling cloud services. A public cloud is what is thought of as the cloud in the usual sense; that is, resources dynamically provisioned over the Internet using web applications from an off-site third-party provider that supplies shared resources and bills on a utility computing basis.

Private clouds exist within your company's firewall and are managed by your organization. They are cloud services you create and control within your enterprise. Private clouds offer many of the same benefits as the public clouds — the major distinction being that your organization is in charge of setting up and maintaining the cloud.

Hybrid clouds are a combination of the public and the private cloud using services that are in both the public and private space. Management responsibilities are divided between the public cloud provider and the business itself. Using a hybrid cloud, organizations can determine the objectives and requirements of the services to be created and obtain them based on the most suitable alternative.

Community clouds are the ones which are created to serve a common function or purpose. Basically it is suitable for a group of organizations which serves a common mission, policies, security, regulatory compliance needs and so on.

Understanding Services and Applications by Type

Infrastructure as a Service

Infrastructure as a Service (IaaS) is a cloud computing service model in which hardware is virtualized in the cloud. In this particular model, the service vendor owns the

equipment: servers, storage, network infrastructure, and so forth. The developer interacts with the IaaS model to create virtual private servers, virtual private storage, virtual private networks, and so on, and then populates these virtual systems with the applications and services it needs to complete its solution.

IaaS Workloads

The fundamental unit of virtualized client in an IaaS deployment is called a workload. A workload simulates the ability of a certain type of real or physical server to do an amount of work. In other words it is the amount of resources allocated to a client to do an amount of work.

In cloud computing, a provisioned server called an instance is reserved by a customer, and the necessary amount of computing resources needed to achieve that type of physical server is allocated to the client's needs.

Consider a transactional eCommerce system, for which a typical stack contains the following
components:

- Web server
- Application server
- File server
- Database
- Transaction engine

This e-Commerce system has several different workloads that are operating: queries against the database, processing of business logic, and serving up clients' Web pages.

Pods, aggregation, and silos

Workloads support a certain number of users, at which point you exceed the load that the instance sizing allows. When you reach the limit of the largest virtual machine instance possible, you must make a copy or clone of the instance to support additional

users. A group of users within a particular instance is called a pod. Pods are managed by a Cloud Control System (CCS). In AWS, the CCS is the AWS Management Console.Sizing limitations for pods need to be accounted for if you are building a large cloud-based application. Pods are aggregated into pools within an IaaS region or site called an availability zone. In very large cloud computing networks, when systems fail, they fail on a pod-by-pod basis, and often on a zone-by-zone basis. A failover system between zones gives IaaS private clouds a very high degree of availability.

When a cloud computing infrastructure isolates user clouds from each other so the management system is incapable of interoperating with other private clouds, it creates an information silo, or simply a silo. Most often, the term silo is applied to PaaS offerings such as Force.com or QuickBase, but silos often are an expression of the manner in which a cloud computing infrastructure is architected. Silos are the cloud computing equivalent of compute islands: They are processing domains that are sealed off from the outside.

When you create a private virtual network within an IaaS framework, the chances are high that you are creating a silo. Silos impose restrictions on interoperability that runs counter to the open nature of build-componentized service-oriented applications. However, that is not always a bad thing. A silo can be its own ecosystem; it can be protected and secured in ways that an open system can't be. Silos just aren't as flexible as open systems and are subject to vendor lock-in.

Defining PAAS

The Platform as a Service model describes a software environment in which a developer can create customized solutions within the context of the development tools that the platform provides. Platforms can be based on specific types of development languages, application frameworks, or other constructs.The one example that is most quoted as a PaaS offering is Google's App Engine platform. Developers program against the App Engine using Google's published APIs. The tools for working within the

development framework, as well as the structure of the file system and data stores, are defined by Google.

The difficulty with PaaS is that it locks the developer (and the customer) into a solution that is dependent upon the platform vendor. An application written in Python against Google's API using the Google App Engine is likely to work only in that environment. There is considerable vendor lock-in associated with a PaaS solution.

Defining SAAS

The most complete cloud computing service model is one in which the computing hardware and software, as well as the solution itself, are provided by a vendor as a complete service offering. It is referred to as the Software as a Service (SaaS) model. SaaS provides the complete infrastructure, software, and solution stack as the service offering.

Examples of SaaS software for end-users are Google Gmail and Calendar, QuickBooks online, Zoho Office Suite etc.However, many other SaaS solutions expose Application Programming Interfaces (API) to developers to allow them to create custom composite applications.These APIs may alter the security model used, the data schema, workflow characteristics, and other fundamental features of the service's expression as experienced by the user. Examples of an SaaS platform with an exposed API are Salesforce.com and Quicken.com. So SaaS does not necessarily mean that the software is static or monolithic.

SaaS Characteristics

All Software as a Service (SaaS) applications share the following characteristics:

1.The software is available over the Internet globally through a browser on demand.

2. The typical license is subscription-based or usage-based and is billed on a recurring basis.In a small number of cases a flat fee may be changed, often coupled with a maintenance fee.

3. The software and the service are monitored and maintained by the vendor, regardless of where all the different software components are running.There may be executable client-side code, but the user isn't responsible for maintaining that code or its interaction with the service.
4. Reduced distribution and maintenance costs and minimal end-user system costs generally make SaaS applications cheaper to use than their shrink-wrapped versions.
5. Such applications feature automated upgrades, updates, and patch management and much faster rollout of changes.
6. SaaS applications often have a much lower barrier to entry than their locally installed competitors, a known recurring cost, and they scale on demand (a property of cloud computing in general).
7. All users have the same version of the software so each user's software is compatible with another's.
8. SaaS supports multiple users and provides a shared data model through a single-instance,multi-tenancy model.

Open SaaS and SOA

A considerable amount of SaaS software is based on open source software. When open source software is used in a SaaS, you may hear it referred to as Open SaaS. The advantages of using open source software are that systems are much cheaper to deploy because you don't have to purchase the operating system or software, there is less vendor lock-in, and applications are more portable.

The componentized nature of SaaS solutions enables many of these solutions to support a feature called mashups. A mashup is an application that can display a Web page that shows data and supports features from two or more sources. Annotating a map such as Google maps is an example of a mashup. Mashups are considered one of the premier examples of Web 2.0, and that is technology's ability to support social network systems.

Gartner Group predicts that approximately 25 percent of all software sold by 2011 will use the SAAS model, offered either by vendors or an intermediary party, sometimes

referred to as an aggregator. An aggregator bundles SaaS applications from different vendors and presents them as part of a unigied platform or solution.

Salesforce.com and CRM SaaS

Perhaps the best-known example of Software as a Service (SaaS) is the Customer Relationship Management software offered by Salesforce.com whose solution offers sales, service, support, marketing, content, analytical analysis, and even collaboration through a platform called Chatter.Salesforce.com extended its SaaS offering to allow developers to create add-on applications, essentially turning the SaaS service into a Platform as a Service (PaaS) offering called the Force.com platform.

Applications built on Force.com are in the form of the Java variant called Apex using an XML syntax for creating user interfaces in HTML, Ajax, and Flex. Nearly a thousand applications now exist for this platform from hundreds of vendors.

UNIT V

USING PLATFORMS

Understanding Abstraction and Virtualization

Load Balancing and Virtualization

One characteristic of cloud computing is virtualized network access to a service. No matter where you access the service, you are directed to the available resources. The technology used to distribute service requests to resources is referred to as load balancing.Load balancing is an optimization technique; it can be used to increase utilization and throughput, lower latency, reduce response time, and avoid system overload.Load balancer uses different types of algorithm to decide where the traffic is routed. It also creates a session so that subsequent traffic related to that session is routed to the same resource.

Advanced load balancing

The more sophisticated load balancers are workload managers. They determine the current utilization of the resources in their pool, the response time, the work queue length, connection latency and capacity, and other factors in order to assign tasks to each resource.An Application Delivery Controller (ADC) is a combination load balancer and application server that is a server placed between a firewall or router and a server farm providing Web services. An Application Delivery Controller is assigned a virtual IP address (VIP) that it maps to a pool of servers based on application specific criteria. An ADC is a combination network and application layer device.

An ADC is considered to be an advanced version of a load balancer as it not only can provide the features described in the previous paragraph, but it conditions content in order to lower the workload of the Web servers.

Understanding Hypervisors

Load balancing virtualizes systems and resources by mapping a logical address to a physical address. Another fundamental technology for abstraction creates virtual systems out of physical systems.

Given a computer system with a certain set of resources, you can set aside portions of those resources to create a virtual machine. From the standpoint of applications or users, a virtual machine has all the attributes and characteristics of a physical system but is strictly software that emulates a physical machine. A system virtual machine (or a hardware virtual machine) has its own address space in memory, its own processor resource allocation, and its own device I/O using its own virtual device drivers. Some virtual machines are designed to run only a single application or process and are referred to as process virtual machines.

Virtual machines provide the capability of running multiple machine instances, each with their own operating system. From the standpoint of cloud computing, these features enable VMMs to manage application provisioning, provide for machine instance cloning

and replication, allow for graceful system failover, and provide several other desirable features. The downside of virtual machine technologies is that having resources indirectly addressed means there is some level of overhead.

Virtual machine types

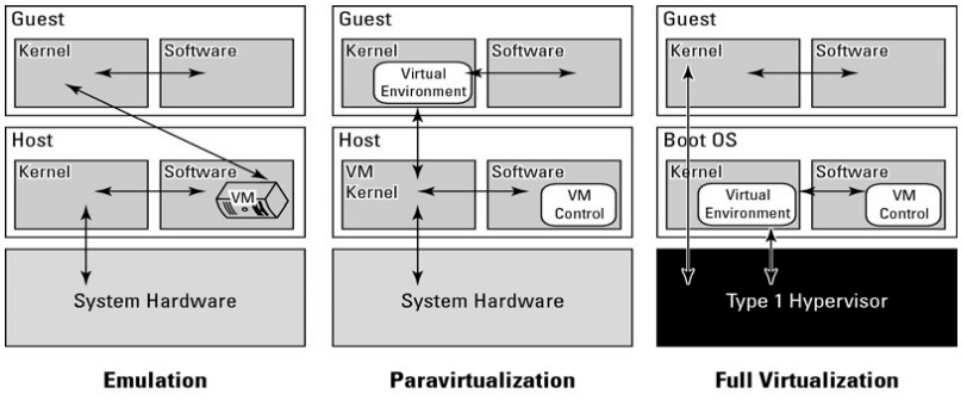
A low-level program is required to provide system resource access to virtual machines, and this program is referred to as the hypervisor or Virtual Machine Monitor (VMM). A hypervisor running on bare metal is a Type 1 VM or native VM. Examples of Type 1 Virtual Machine Monitors are LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogix VLX, VMware ESX and ESXi, and Wind River VxWorks, among others. The operating system loaded into a virtual machine is referred to as the guest operating system, and there is no constraint on running the same guest on multiple VMs on a physical system. Type 1 VMs have no host operating system because they are installed on a bare system.

Some hypervisors are installed over an operating system and are referred to as Type 2 or hosted VM. Examples of Type 2 Virtual Machine Monitors are Containers, KVM, Microsoft Hyper V, Parallels Desktop for Mac, Wind River Simics, VMWare Fusion, Virtual Server 2005 R2, Xen, Windows Virtual PC, and VMware Workstation 6.0 and Server, among others.

On a Type 2 VM, a software interface is created that emulates the devices with which a system would normally interact. This abstraction is meant to place many I/O operations outside the virtual environment, which makes it both programmatically easier and more efficient to execute device I/O than it would be inside a virtual environment. This type of virtualization is sometimes referred to as paravirtualization, and it is found in hypervisors such as Microsoft's Hyper-V and Xen. It is the host operating system that is performing the I/O through a para-API.

There is a difference between **emulation**, **paravirtualization,** and **full virtualization.** In emulation, the virtual machine simulates hardware, so it can be independent of the

underlying system hardware. A guest operating system using emulation does not need to be modified in any way. Paravirtualization requires that the host operating system provide a virtual machine interface for the guest operating system and that the guest access hardware through that host VM. An operating system running as a guest on a paravirtualization system must be ported to work with the host interface. Finally, in a full virtualization scheme, the VM is installed as a Type 1 Hypervisor directly onto the hardware. All operating systems in full virtualization communicate directly with the VM hypervisor, so guest operating systems do not require any modification. Guest operating systems in full virtualization systems are generally faster than other virtualization schemes.The Virtual Machine Interface (VMI) open standard that VMware has proposed is an example of a para virtualization API.



Process or application virtual machines

Most folks run the Java Virtual Machine or Microsoft's .NET Framework VM (called the Common Language Runtime or CLR) on their computers. A process virtual machine instantiates when a command begins a process, the VM is created by an interpreter, the VM then executes the process, and finally the VM exits the system and is destroyed. During the time the VM exists, it runs as a high-level abstraction.

Operating system virtualization

Some operating systems such as Sun Solaris and IBM AIX 6.1 support a feature known as operating system virtualization. This type of virtualization creates virtual servers at

the operating system or kernel level. Each virtual server is running in its own virtual environment (VE) as a virtual private server (VPS). Different operating systems use different names to describe these machine instances, each of which can support its own guest OS. However, unlike true virtual machines, VPS must all be running the same OS and the same version of that OS. Sun Solaris 10 uses VPS to create what is called Solaris Zones. With IBM AIX, the VPS is called a System Workload Partition (WPAR). This type of virtualization allows for a dense collection of virtual machines with relatively low overhead.

Understanding Machine Imaging

As it is apparent by now abstraction in cloud computing can be achieved through redirection and virtualization. A third mechanism is commonly used to provide system portability, instantiate applications, and provision and deploy systems in the cloud. This third mechanism is through storing the state of a systems using a system image.

A system image makes a copy or a clone of the entire computer system inside a single container such as a file. The system imaging program is used to make this image and can be used later to restore a system image. Some imaging programs can take snapshots of systems, and most allow you to view the files contained in the image and do partial restores.

A prominent example of a system image and how it can be used in cloud computing architectures is the Amazon Machine Image (AMI) used by Amazon Web Services to store copies of a virtual machine. An AMI is a file system image that contains an operating system, all appropriate device drivers, and any applications and state information that the working virtual machine would have.

Porting Applications

Cloud computing applications have the ability to run on virtual systems and for these systems to be moved as needed to respond to demand. Systems (VMs running

applications), storage, and network assets can all be virtualized and have sufficient flexibility to give acceptable distributed WAN application performance.

Developers who write software to run in the cloud will undoubtedly want the ability to port their applications from one cloud vendor to another, but that is a much more difficult proposition. Cloud computing is a relatively new area of technology, and the major vendors have technologies that don't interoperate with one another.

Simple Cloud API

If you build an application on a platform such as Microsoft Azure, porting that application to Amazon Web Services or GoogleApps may be difficult, if not impossible. In an effort to create an interoperability standard, Zend Technologies has started an open source initiative to create a common application program interface that will allow applications to be portable.

The initiative is called the Simple API for Cloud Application Services, and the effort has drawn interest from several major cloud computing companies. Among the founding supporters are IBM, Microsoft, Nivanix, Rackspace, and GoGrid.

AppZero Virtual Application Appliance

Applications that run in datacenters are captive to the operating systems and hardware platforms that they run on. So moving an application from one platform to another isn't nearly as simple as moving a machine image from one system to another. The situation is further complicated by the fact that applications are tightly coupled with the operating systems on which they run. An application running on Windows, for example, isn't isolated from other applications. When the application loads, it often loads or uses different Dynamic Link Libraries (DLL), and it is through the sharing or modification of DLLs that Windows applications get themselves in trouble. Further modifications include modifying the registry during installation. These factors make it difficult to port applications from one platform to another without lots of careful work. If you are a Platform as a Service (PaaS) application developer, you are packaging a complete

software stack that includes not only your application, but the operating system and application logic and rules as well.0Vendor lock-in for you application is assured.

The ability to run an application from whatever platform you want is not one of the characteristics of cloud computing, but you can imagine that it is a very attractive proposition. While the Simple Cloud API is useful for applications written in PHP, other methods may be needed to make applications easily portable. The AppZero solution creates a virtual application appliance as an architectural layer between the Windows or the UNIX operating system and applications. The virtualization layer serves as the mediator for file I/O, memory I/O, and application calls and response to DLLs, which has the effect of sandboxing the application. The running application in AppZero changes none of the registry entries or any of the files on the Windows Server.
VAA creates a container that encapsulates the application and all the application's dependencies within a set of files; it is essentially an Application Image for a specific OS. Dependencies include DLL, service settings, necessary configuration files, registry entries, and machine and network settings. This container forms an installable server-side application stack that can be run after installation, but has no impact on the underlying operating system. VAAs are created using the AppZero Creator wizard, managed with the AppZero Admin tool, and may be installed using the AppZero Director, which creates a VAA runtime application. If desired, an application called AppZero Dissolve removes the VAA virtualization layer from the encapsulated application and installs that application directly into the operating system.

Capacity Planning

A capacity planner seeks to meet the future demands on a system by providing the additional capacity to fulfill those demands. Many people equate capacity planning with system optimization (or performance tuning, if you like), but they are not the same. System optimization aims to get more production from the system components you have. Capacity planning measures the maximum amount of work that can be done using the current technology and then adds resources to do more work as needed.

If system optimization occurs during capacity planning, that is all to the good; but capacity planning efforts focus on meeting demand. If that means the capacity planner must accept the inherent inefficiencies in any system, so be it.

Capacity planning is an iterative process with the following steps:
1.Determine the characteristics of the present system.
2. Measure the workload for the different resources in the system: CPU, RAM, disk, network, and so forth.
3. Load the system until it is overloaded, determine when it breaks, and specify what is required to maintain acceptable performance. Knowing when systems fail under load and what factor(s) is responsible for the failure is the critical step in capacity planning.
4.Predict the future based on historical trends and other factors.
5.Deploy or tear down resources to meet your predictions.
6. Iterate Steps 1 through 5 repeatedly.

Performance Monitoring

Performance should be monitored to understand the capacity. Capacity planning must measure system-level statistics, determining what each system is capable of, and how resources of a system affect system-level performance.

Below is a list of some LAMP performance tools:
Alertra-Websitemonitoring service
Cacti - Open source RRDTool graphing module
Collectd - System statistics collection daemon
Dstat - System statistics utility; replaces vmstat, iostat, netstat, and ifstat
Ganglia - Open source distributed monitoring system
RRDTool - Graphing and performance metrics storage utility
ZenOSS - Operations monitor, both open source and commercial versions

Load testing

Examining your server under load for system metrics isn't going to give you enough information to do meaningful capacity planning. You need to know what happens to a system when the load increases. That is the aim of Load Testing. Load testing is also referred to as performance testing, reliability testing, stress testing, and volume testing. If you have one production system running in the cloud, then overloading that system until it breaks isn't going to make you popular with your colleagues. However, cloud computing offers virtual solutions. One possibility is to create a clone of your single system and then use a tool to feed that clone a recorded set of transactions that represent your application workload.

You may want to consider these load generation tools as well:

- HP LodeRunner
- IBM Rational Performance Tester
- Jmeter
- OpenSTA
- Micro Focus (Borland) SilkPerfomer

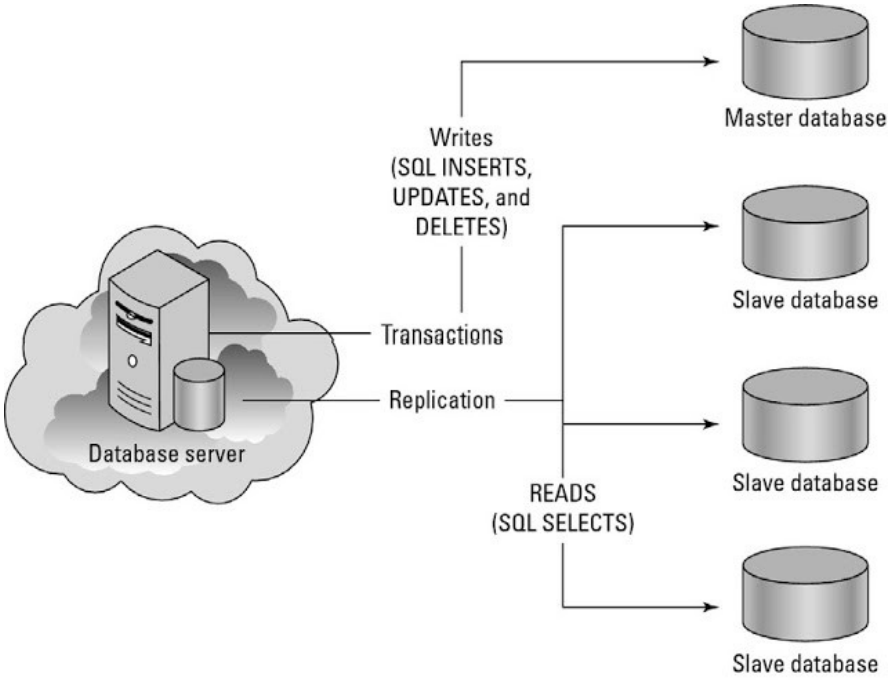
Resource Ceiling

During load testing over a certain load for a particular server, the CPU, RAM, and Disk I/O utilization rates rise but do not reach their resource ceiling. In this instance, the Network I/O reaches its maximum 100-percent utilization at about 50 percent of the tested load, and this factor is the current system resource ceiling.Network I/O is often a bottleneck in Web servers, and this is why, architecturally, Web sites prefer to scale out using many low-powered servers instead of scaling up with fewer but more powerful servers.

Let's consider a slightly more complicated case that you might encounter in MySQL database systems. Database servers are known to exhibit resource ceilings for either their file caches or their Disk I/O performance. To build high-performance applications, many developers replicate their master MySQL database and create a number of slave

MySQL databases. All READ operations are performed on the slave MySQL databases, and all WRITE operations are performed on the master MySQL database. A master/slave database system has two competing processes and the same server that are sharing a common resource: READs and WRITEs to the master/slave databases, and replication traffic between the master database and all the slave databases.

These types of servers reach failure when the amount of WRITE traffic in the form of INSERT, UPDATE, and DELETE operations to the master database overtakes the ability of the system to replicate data to the slave databases that are servicing SELECT (query/READ) operations.



Network Capacity

There are three aspects to assessing network capacity:

• Network traffic to and from the network interface at the server, be it a physical or virtual interface or server.

•Network traffic from the cloud to the network interface.

•Network traffic from the cloud through your ISP to your local network interface (your computer)

To measure network traffic at a server's network interface, you need to employ what is commonly known as a network monitor, which is a form of packet analyzer. Microsoft includes a utility called the Microsoft Network Monitor as part of its server utilities, and there are many third-party products in this area.

Alternative names for network analyzer include packet analyzer, network traffic monitor, protocol analyzer, packet sniffer, and Ethernet sniffer, and for wireless networks, wireless or wifi sniffer, network detector, and so on.

Scaling

In capacity planning, after you have made the decision that you need more resources, you are faced with the fundamental choice of scaling your systems. You can either scale vertically (scale u00p) or scale horizontally (scale out), and each method is broadly suitable for different types of applications.

To scale vertically, you add resources to a system to make it more powerful. For example, during scaling up, you might replace a node in a cloud-based system that has a dual-processor machine instance equivalence with a quad-processor machine instance equivalence. You also can scale up when you add more memory, more network throughput, and other resources to a single node. Scaling out indefinitely eventually leads you to an architecture with a single powerful supercomputer.

Vertical scaling allows you to use a virtual system to run more virtual machines (operating system instance), run more daemons on the same machine instance, or take advantage of more RAM (memory) and faster compute times. Applications that benefit from being scaled up vertically include those applications that are processor-limited such as rendering or memory-limited such as certain database operations—queries against an in-memory index, for example.

Horizontal scaling or scale out adds capacity to a system by adding more individual nodes. In a system where you have a dual-processor machine instance, you would scale out by adding more dual-processor machines instances or some other type of commodity system. Scaling out indefinitely leads you to an architecture with a large number of servers (a server farm), which is the model that many cloud and grid computer networks use.

Horizontal scaling allows you to run distributed applications more efficiently and is effective in using hardware more efficiently because it is both easier to pool resources and to partition them. Although your intuition might lead you to believe otherwise, the world's most powerful computers are currently built using clusters of computers aggregated using high speed interconnect technologies such as InfiniBand or Myrinet. Scale out is most effective when you have an I/O resource ceiling and you can eliminate the communications bottleneck by adding more channels. Web server connections are a classic example of this situation.

Exploring Platform as a Service

With Platform as a Service systems, you are given a toolkit to work with, a virtual machine to run your software on, and it is up to you to design the software and its user-facing interface in a way that is appropriate to your needs. So PaaS systems range from full-blown developer platforms like Windows Azure Platform to systems like Drupal, Squarespace, Wolf, and others where the tools are modules that are very well developed and require almost no coding.Many Content Management Systems (CMS) are essentially PaaS services where you get standard parts and can build Web sites and other software like Tinker Toys.

The services provided by PAAS model is:

• Application development: A PaaS platform either provides the means to use programs you create in a supported language or offers a visual development environment that writes the code for you.

• Collaboration: Many PaaS systems are set up to allow multiple individuals to work on the same projects.
• Data management: Tools are provided for accessing and using data in a data store.
• Instrumentation, performance, and testing: Tools are available for measuring your applications and optimizing their performance.
• Storage: Data can be stored in either the PaaS vendor's service or accessed from a third-party storage service.
• Transaction management: Many PaaS systems provide services such as transaction managers or brokerage service for maintaining transaction integrity.

PaaS systems exist to allow you to create software that can be hosted as SaaS systems or to allow for the modification of existing SaaS applications. A good PaaS system has certain desirable characteristics that are important in developing robust, scalable, and hopefully portable applications. On this list would be the following attributes:

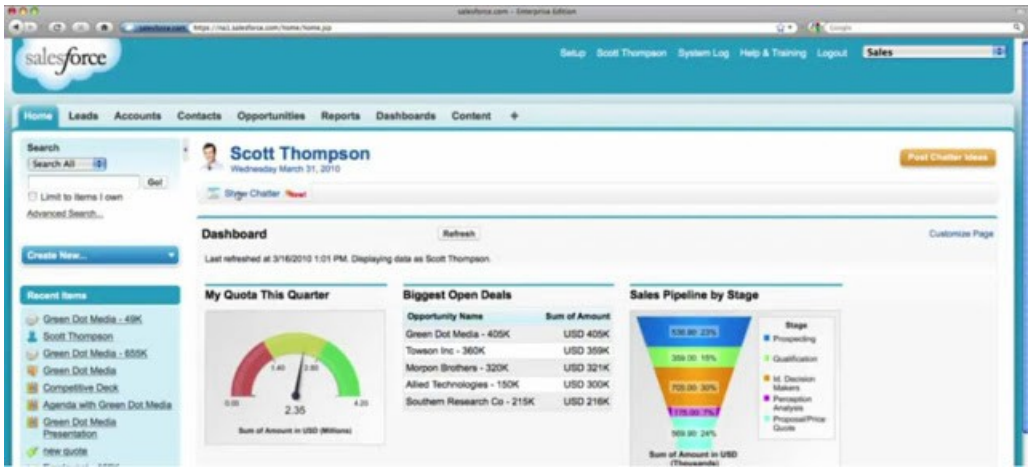
- Separate of data management from the user interface.
- Reliance on cloud computing standards.
- An integrated development environment(IDE).
- Multi-tenant architecture support, security, and scalability
- Performance monitoring, testing and optimization tools

Salesforce.com versus Force.com: SaaS versus PaaS

Force.com. Salesforce.com is a Web application suite that is an SaaS. Force.com is Salesforce.com's PaaS platform for building your own services. The Salesforce.com team created hosted software based on a cloud computing model: pay as you go, simple to use, and multifunctional. The Salesforce.com platform looks like a typical Web site such as Amazon.com, with a multi-tabbed interface—each tab being an individual application.

Some of the applications included in the site are:

• Accounts and Contact
• Analytics and Forecasting
• Approvals and Workflow
• Chatter (Instant Messaging/Collaboration)
• Content Library
• E-mail and Productivity
• Jigsaw Business Data
• Marketing and Leads
• Opportunities and Quotes
• Partner Relationship
• Sales
• Service and Support



Because Salesforce.com is browser-based, it is platform-independent. However, the company has extended its audience to mobile devices, such as the Android, Blackberry, iPhone, and Windows Mobile Devices. It also has a server product that supports Salesforce.com applications in-house called the Resin Application Server.

The PAAS platform Force.com uses a Java-based programming language called Apex for its application building, and it has an interface builder called Visualforce that allows a

developer to create interfaces using HTML, Flex, and AJAX. Visualforce uses an XML-type language in its visual interface builder.

Application development

A PaaS provides the tools needed to construct different types of applications that can work together in the same environment.

These are among the common application types:

- Composite business applications
- Data portals
- Mashups of multiple data sources

A mashup is a Web page that displays data from two or more data sources. The various landmarks and overlays you find in Google Earth, or annotated maps, are examples of mashups.

These applications must be able to share data and run in a multi-tenant environment. To make applications work together more easily, a common development language such as Java or Python is usually offered. The more commonly used the language is, the more developers and developer services are going to be available to help users of platform applications. The use of application frameworks such as Ruby on Rails is useful in making application building easier and more powerful.

All PaaS application development must take into account lifecycle management. As an application ages, it must be upgraded, migrated, grown, and eventually phased out or ported. Many PaaS vendors offer systems that are integrated lifecycle development platforms. That is, the vendor provides a full software development stack for the programmer to use, and it isn't expected that the developer will need to go outside of the service to create his application.

An integrated lifecycle platform includes the following:

• The virtual machine and operating system (often offered by an IaaS)
• Data design and storage
• A development environment with defined Application Programming Interfaces
• Middleware
• Testing and optimization tools
• Additional tools and services

Google AppEngine, Microsoft Windows Azure Platform, EccentexAppBase, LongJump, and Wolf are examples of integrated lifecycle platforms.Some PaaS services allow developers to modify existing software. These services are referred to as anchored lifecycle platforms. Examples of an anchored lifecycle platform are QuickBooks.com and Salesforce.com. The applications in these two services are fixed, but developers can customize which applications the users see, how those applications are branded, and a number of features associated with the different applications.

Using PaaS Application Frameworks

Application frameworks provide a means for creating SaaS hosted applications using a unified development environment or an integrated development environment (IDE).Many Web sites are based on the notion of information management and organization; they are referred to as content management systems (CMS). A database is a content management system, but the notion of a Web site as a CMS adds a number of special features to the concept that includes rich user interaction, multiple data sources, and extensive customization and extensibility.

Some examples of Paas Application frameworks are:

Drupal

The Drupal CMS is an example of this type of PaaS. It is extensively used and has broad industry impact, and it is a full-strength developer tool.

Note: The portability of the applications you create in a PaaS is an extremely valuable feature. If your service goes out of business, being able to port an application by simply redeploying that application to another IaaS can be a lifesaver.

Squarespace

Squarespace (http://www.squarespace.com/), is an example of a next-generation Web site builder and deployment tool that has elements of a PaaS development environment.

Eccentex

Eccentex is a Culver City, California, company founded in 2005 that has a PaaS development platform for Web applications based on SOA component architecture to create what it calls Cloudware applications using its AppBase architecture.

LongJump

LongJump (http://www.longjump.com/) is a Sunnyvale, California, company hosting service created in 2003 with a PaaS application development suite. Its development environment is based on Java and uses REST/SOAP APIs. LongJump's PaaS is based on standard Java/JavaScript, SOAP, and REST.

WaveMaker

WaveMaker is a visual rapid application development environment for creating Java-based Web and cloud Ajax applications. The software is open-source and offered under the Apache license. WaveMaker is a WYSIWYG (What You See is What You Get) drag-and-drop environment that runs inside a browser. The metaphor used to build applications is described as the Model-View-Controller system of application architecture.

Wolf Frameworks

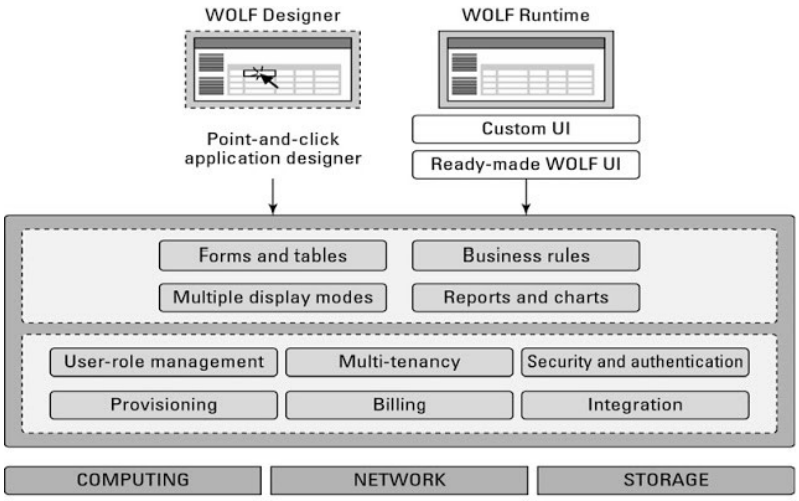
Many application frameworks like Google AppEngine and the Windows Azure Platform are tied to the platform on which they run. You can't build an AppEngine application and port it to Windows Azure without completely rewriting the application. There isn't any particular necessity to build an application framework in this way, but it suits the purpose of these particular vendors: for Google to have a universe of Google applications that build on the Google infrastructure, and for Microsoft to provide another platform on which to extend .NET Framework applications for their developers. If you are building an application on top of an IaaS vendor such as AWS, GoGrid, or

RackSpace, what you really want are application development frameworks that are open, standards-based, and portable. **Wolf Frameworks** is an example of a PaaS vendor offering a platform on which you can build an SaaS solution that is open and cross-platform.

Wolf Frameworks is based on the three core Windows SOA standard technologies of cloud computing:

• AJAX, asynchronous Java
• XML
• .NET Framework

Wolf Frameworks uses a C# engine and supports both Microsoft SQL Server and MySQL database. Applications that you build in Wolf are 100-percent browser-based and support mashable and multisource overlaid content.



Using Google Web Services

Google App Engine (GAE) is a Platform as a Service (PaaS) cloud-based Web hosting service on Google's infrastructure.This service allows developers to build and deploy Web applications and have Google manage all the infrastructure needs, such as monitoring, failover, clustering, machine instance management, and so forth. For an application to run on GAE, it must comply with Google's platform standards, which

narrows the range of applications that can be run and severely limits those applications' portability.

GAE supports the following major features:

- Dynamic Web services based on common standards
- Automatic scaling and load balancing
- Authentication using Google's Accounts API
- Persistent storage, with query access sorting and transaction management features
- Task queues and task scheduling
- A client-side development environment for simulating GAE on your local system
- One of either two runtime environments: Java or Python

When you deploy an application on GAE, the application can be accessed using your own domain name or using the Google Apps for Business URL.

Google App Engine currently supports applications written in Java and in Python, although there are plans to extend support to more languages in the future. The service is meant to be language-agnostic. A number of Java Virtual Machine languages are compliant with GAE, as are several Python Web frameworks that support the Web Server Gateway Interface (WSGI) and CGI. Google has its own Webapp framework designed for use with GAE. The AppScale open-source framework also may be used for running applications on GAE.

To encourage developers to write applications using GAE, Google allows for free application development and deployment up to a certain level of resource consumption. Resource limits are described on Google's quota page at http://code.google.com/appengine/docs/quotas.html, and the quota changes from time to time.

When you enable billing for an application deployed to GAE, you pay for consumption of CPU, network I/O, and other usage above the level of the free quotas that GAE allows.Applications running in GAE are isolated from the underlying operating system, which Google describes as running in a sandbox. This allows GAE to optimize the system so Web requests can be matched to the current traffic load. It also allows applications to be more secure because applications can connect only to computers using the specified URLs for the e-mail and fetch services using HTTP or HTTPS over the standard well-known ports. URL fetch uses the same infrastructure that retrieves Web pages on Google. The mail service also supports Gmail's messaging system.

Applications also are limited in that they can only read files; they cannot write to the file system directly. To access data, an application must use data stored in the memcache (memory cache), the datastore, or some other persistent service. Memcache is a fast in-memory key-value cache that can be used between application instances. For persistent data storage of transactional data, the datastore is used. Additionally, an application responds only to a specific HTTP request—in real-time, part of a queue, or scheduled—and any request is terminated if the response requires more than 30 seconds to complete.

The App Engine relies on the Google Accounts API for user authentication, the same system used when you log into a Google account. This provides access to e-mail and display names within your app, and it eliminates the need for an application to develop its own authentication system. Applications can use the User API to determine whether a user belongs to a specific group and even whether that person is an administrator for your application

Using Amazon Web Services

Working with the Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2) is a virtual server platform that allows users to create and run virtual machines on Amazon's server farm. With EC2, you can launch and run server instances called Amazon Machine Images (AMIs) running different

operating systems such as Red Hat Linux and Windows on servers that have different performance profiles. You can add or subtract virtual servers elastically as needed; cluster, replicate, and load balance servers; and locate your different servers in different data centers or “zones” throughout the world to provide fault tolerance. The term elastic refers to the ability to size your capacity quickly as needed.

The difference between an instance and a machine image is that an instance is the emulation of a hardware platform such as X86, IA64, and so on running on the Xen hypervisor. A machine image is the software and operating system running on top of the instance. A machine image may be thought of as the contents of a boot drive, something that you could package up with a program such as Ghost, Acronis, or TrueImage to create a single file containing the exact contents of a volume. A machine image should be composed of a hardened operating system with as few features and capabilities as possible and locked down as much as possible.

The pricing of these different AMI types depends on the operating system used, which data center the AMI is located in (you can select its location), and the amount of time that the AMI runs. Rates are quoted based on an hourly rate. Additional charges are applied for:

- the amount of data transferred
- whether Elastic IP Addresses are assigned
- your virtual private server's use of Amazon Elastic Block Storage (EBS)
- whether you use Elastic Load Balancing for two or more servers
- other features

System images and software

You can choose to use a template AMI system image with the operating system of your choice or create your own system image that contains your custom applications, code libraries, settings, and data. Security can be set through passwords, Kerberos tickets, or certificates.

These operating systems are offered:

- Red Hat Enterprise Linux
- OpenSuse Linux
- Ubuntu Linux
- Sun OpenSolaris
- Fedora
- Gentoo Linux
- Oracle Enterprise Linux
- Windows Server 2003/2008 32-bit and 64-bit up to Data Center Edition
- Debian

Most of the system image templates that Amazon AWS offers are based on Red Hat Linux, Windows Server, Oracle Enterprise Linux, and OpenSolaris. When you create a virtual private server, you can use the Elastic IP Address feature to create what amounts to a static IP v4 address to your server. This address can be mapped to any of your AMIs and is associated with your AWS account. You retain this IP address until you specifically release it from your AWS account. Should a machine instance fail, you can map your Elastic IP Address to fail over to a different AMI. You don't need to wait until a DNS server updates the IP record assignment, and you can use a form to configure the reverse DNS record of the Elastic IP address change.

Working with Amazon Storage Systems

When you create an Amazon Machine Instance you provision it with a certain amount of storage. That storage is temporal, it only exists for as long as your instance is running. All of the data contained in that storage is lost when the instance is suspended or terminated, as the storage is reassigned to the pool for other AWS users to use. For this and other reasons you need to have access to persistent storage. The Amazon Simple Storage System provides block storage, but is set up in a way that is somewhat unique from other storage systems you may have worked with in the past.

Amazon Simple Storage System (S3)

Amazon S3's cloud-based storage system allows you to store data objects ranging in size from 1 byte up to 5GB in a flat namespace. In S3, storage containers are referred to as buckets, and buckets serve the function of a directory, although there is no object hierarchy to a bucket, and you save objects and not files to it. It is important that you do not associate the concept of a file system with S3, because files are not supported; only objects are stored. Additionally, you do not “mount” a bucket as you do a file system.

The S3 system allows you to assign a name to a bucket, but that name must be unique in the S3 namespace across all AWS customers. Access to an S3 bucket is through the S3 Web API (either with SOAP or REST) and is slow relative to a real-world disk storage system. S3's performance limits its use to non-operational functions such as data archiving and retrieval or disk backup. The REST API is preferred to the SOAP API, because it is easier to work with large binary objects with REST.

The S3 service is used by many people as the third level backup component in a 3-2-1 backup strategy. That is, you have your original data (1), a copy of your data (2), and an off-site copy of your data (3); the latter of these may be S3.

Note: Keep in mind that while Amazon S3 is highly reliable, it is not highly available. You can definitely get your data back from S3 at some point with guaranteed 100% fidelity, but the service is not always connected and experiences service outages.

Amazon Elastic Block Store (EBS)

The third of Amazon's data storage systems are devoted to Amazon Elastic Block Storage (EBS), which is a persistent storage service with a high operational performance. Advantages of EBS are that it can store file system information and its performance is higher and much more reliable than Amazon S3. That makes EBS valuable as an operational data storage medium for AWS. The cost of creating an EBS volume is also greater than creating a similarly sized S3 bucket.

CloudFront

Amazon CloudFront is referred to as a content delivery network (CDN), and sometimes called edge computing. In edge computing, content is pushed out geographically so the data is more readily available to network clients and has a lower latency when requested. You enable CloudFront through a selection in the AWS Management Console.

You can think of a CDN as a distributed caching system. CloudFront servers are located throughout the world—in Europe, Asia, and the United States. As such, CloudFront represents yet another level of Amazon cloud storage. A user requesting data from a CloudFront site is referred to the nearest geographical location. CloudFront supports “geo-caching” data by performing static data transfers and streaming content from one CloudFront location to another.

Understanding Amazon Database Services

Amazon offers two different types of database services: Amazon SimpleDB, which is non-relational, and Amazon Relational Database Service (Amazon RDS).

Amazon SimpleDB

Amazon SimpleDB is an attempt to create a high performance data store with many database features but without the overhead. To create a high performance “simple” database, the data store created is flat; that is, it is non-relational and joins are not supported. Data stored in SimpleDB domains doesn't require maintenances of a schema and is therefore easily scalable and highly available because replication is built into the system. Data is stored as collections of items with attribute-value pairs, and the system is akin to using the database function within a spreadsheet.

Amazon Relational Database Service (RDS)

Amazon Relational Database Service is a variant of the MySQL5.1 database system, but one that is somewhat simplified. The purpose of RDS is to allow database applications that already exist to be ported to RDS and placed in an environment that is

relatively automated and easy to use. RDS automatically performs functions such as backups and is deployable throughout AWS zones using the AWS infrastructure.

In RDS, you start by launching a database instance in the AWS Management Console and assigning the DB Instance class and size of the data store. Any database tool that works with MySQL 5.1 will work with RDS. Additionally, you can monitor your database usage as part of Amazon CloudWatch.

Choosing a database for AWS

In choosing a database solution for your AWS solutions, consider the following factors in making your selection:

Choose SimpleDB when index and query functions do not require relational database support.

Use SimpleDB for the lowest administrative overhead.

Select SimpleDB if you want a solution that autoscales on demand.

Choose SimpleDB for a solution that has a very high availability.

Use RDS when you have an existing MySQL database that could be ported and you want to minimize the amount of infrastructure and administrative management required.

Use RDS when your database queries require relation between data objects.

Using Microsoft Cloud Services

Microsoft's approach is to view cloud applications as software plus service. In this model, the cloud is another platform and applications can run locally and access cloud services or run entirely in the cloud and be accessed by browsers using standard Service Oriented Architecture (SOA) protocols.

Microsoft calls their cloud operating system the Windows Azure Platform. You can think of Azure as a combination of virtualized infrastructure to which the .NET Framework has been added as a set of .NET Services. The Windows Azure service itself is a hosted environment of virtual machines enabled by a fabric called Windows Azure AppFabric.

You can host your application on Azure and provision it with storage, growing it as you need it. Windows Azure service is an Infrastructure as a Service offering.

A number of services interoperate with Windows Azure, including SQL Azure (a version of SQL Server), SharePoint Services, Azure Dynamic CRM, and many of Windows Live Services comprising what is the Windows Azure Platform, which is a Platform as a Service cloud computing model.

Windows Live Services is a collection of applications and services that run on the Web. Some of these applications called Windows Live Essentials are add-ons to Windows and downloadable as applications. Other Windows Live Services are standalone Web applications viewable in a browser.

Exploring Microsoft Cloud Services

Microsoft Live is only one part of the Microsoft cloud strategy. The second part of the strategy is the extension of the .NET Framework and related development tools to the cloud. To enable .NET developers to extend their applications into the cloud, or to build .NET style applications that run completely in the cloud, Microsoft has created a set of .NET services, which it now refers to as the Windows Azure Platform.

Azure is a virtualized infrastructure to which a set of additional enterprise services has been layered on top, including:

- A virtualization service called Azure AppFabric that creates an application hosting environment. AppFabric (formerly .NET Services) is a cloud-enabled version of the .NET Framework.
- A high capacity non-relational storage facility called Storage.
- A set of virtual machine instances called Compute.
- A cloud-enabled version of SQL Server called SQL Azure Database.
- A database marketplace based on SQL Azure Database code-named “Dallas.”
- An xRM (Anything Relations Management) service called Dynamics CRM based on MicrosoftDynamics.

- A document and collaboration service based on SharePoint called SharePoint Services.
- Windows Live Services, a collection of services that runs on Windows Live, which can be used in applications that run in the Azure cloud.

Defining the Windows Azure Platform

Azure is Microsoft's Infrastructure as a Service (IaaS) Web hosting service. Compared to Amazon's and Google's cloud services, Azure (the service) is a competitor to AWS. Windows Azure Platform is a competitor to Google's App Engine.

The software plus services approach

Microsoft has a very different vision for cloud services than either Amazon or Google does. In Amazon's case, AWS is a pure infrastructure play. AWS essentially rents you a (virtual) computer on which to run your application. An Amazon Machine Image can be provisioned with an operating system, an enterprise application, or application stack, but that provisioning is not a prerequisite. An AMI is your machine, and you can configure it as you choose. AWS is a deployment enabler.

Google's approach with its Google App Engine (GAE) is to offer a cloud-based development platform on which you can add your program, provided that the program speaks the Google App Engine API and uses objects and properties from the App Engine framework. Google makes it possible to program in a number of languages, but you must write your applications to conform to Google's infrastructure. Google Apps lets you create a saleable cloud-based application, but that application can only work within the Google infrastructure, and the application is not easily ported to other environments.

Microsoft sees the cloud as being a complimentary platform to its other platforms. The company envisages a scenario where a Microsoft developer with an investment in an application wants to extend that application's availability to the cloud. Perhaps the application runs on a server, desktop, or mobile device running some form of Windows. Microsoft calls this approach **software plus services**.

The Windows Azure Platform allows a developer to modify his application so it can run in the cloud on virtual machines hosted in Microsoft datacenters. Windows Azure serves as a cloud operating system, and the suitably modified application can be hosted on Azure as a runtime application where it can make use of the various Azure Services. Additionally, local applications running on a server, desktop, or mobile device can access Windows Azure Services through the Windows Services Platform API.

The Azure Platform

With Azure's architecture (shown in Figure), an application can run locally, run in the cloud, or some combination of both. Applications on Azure can be run as applications, as background processes or services, or as both.

The Azure Windows Services Platform API uses the industry standard REST, HTTP, and XML protocols that are part of any Service Oriented Architecture cloud infrastructure to allow applications to talk to Azure. Developers can install a client-side managed class library that contains functions that can make calls to the Azure Windows Services Platform API as part of their applications. These API functions have been added to Microsoft Visual Studio as part of Microsoft's Integrated Development Environment (IDE).

The Azure Service Platform hosts runtime versions of .NET Framework applications written in any of the languages in common use, such as Visual Basic, C++, C#, Java, and any application that has been compiled for .NET's Common Language Runtime (CLR). Azure also can deploy Web-based applications built with ASP.NET, the Windows Communication Foundation (WCF), and PHP, and it supports Microsoft's automated deployment technologies. Microsoft also has released SDKs for both Java and Ruby to allow applications written in those languages to place calls to the Azure Service Platform API to the AppFabric Service.

The Windows Azure service

Windows Azure is a virtualized Windows infrastructure run by Microsoft on a set of datacenters around the world.

Six main elements are part of Windows Azure:

Application: This is the runtime of the application that is running in the cloud.

Compute: This is the load-balanced Windows server computation and policy engine that allows you to create and manage virtual machines that serve either in a Web role and a Worker role.

A Web role is a virtual machine instance running Microsoft IIS Web server that can accept and respond to HTTP or HTTPS requests. A Worker role can accept and respond to requests, but doesn't run IIS in that virtual machine. Worker roles can communicate with Azure Storage or through direct connections to clients.

Storage: This is a non-relational storage system for large-scale storage.

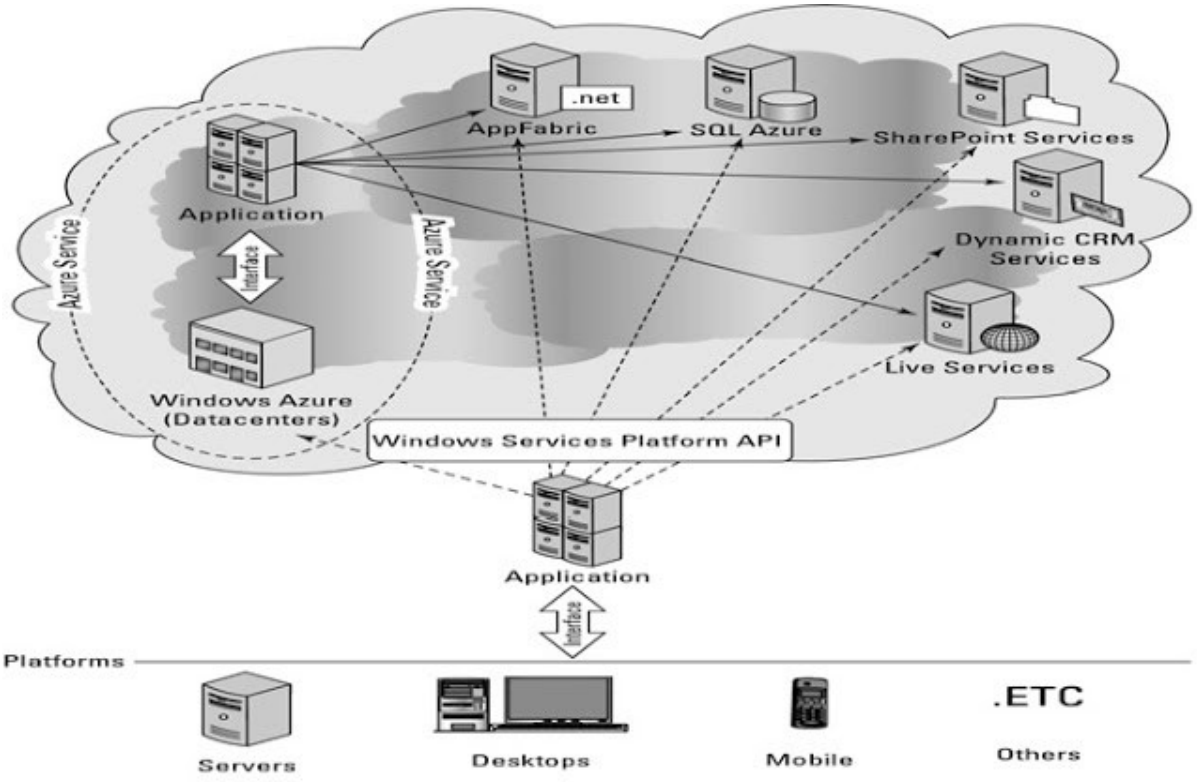
Azure Storage Service lets you create drives, manage queues, and store BLOBs (Binary Large Objects). You manipulate content in Azure Storage using the REST API, which is based on standard HTTP requests and is therefore platform-independent. Stored data can be read using GETs, written with PUTs, modified with POSTs, and removed with DELETE requests. Azure Storage plays the same role in Azure that Amazon Simple Storage Service (S3) plays in Amazon Web Services. For relational database services, SQL Azure may be used.

Fabric: This is the Windows Azure Hypervisor, which is a version of Hyper-V that runs on Windows Server 2008.

Config: This is a management service.

Virtual machines: These are instances of Windows that run the applications and services that are part of a particular deployment.

The Windows Azure Platform extends applications running on other platforms to the cloud using Microsoft infrastructure and a set of enterprise services.

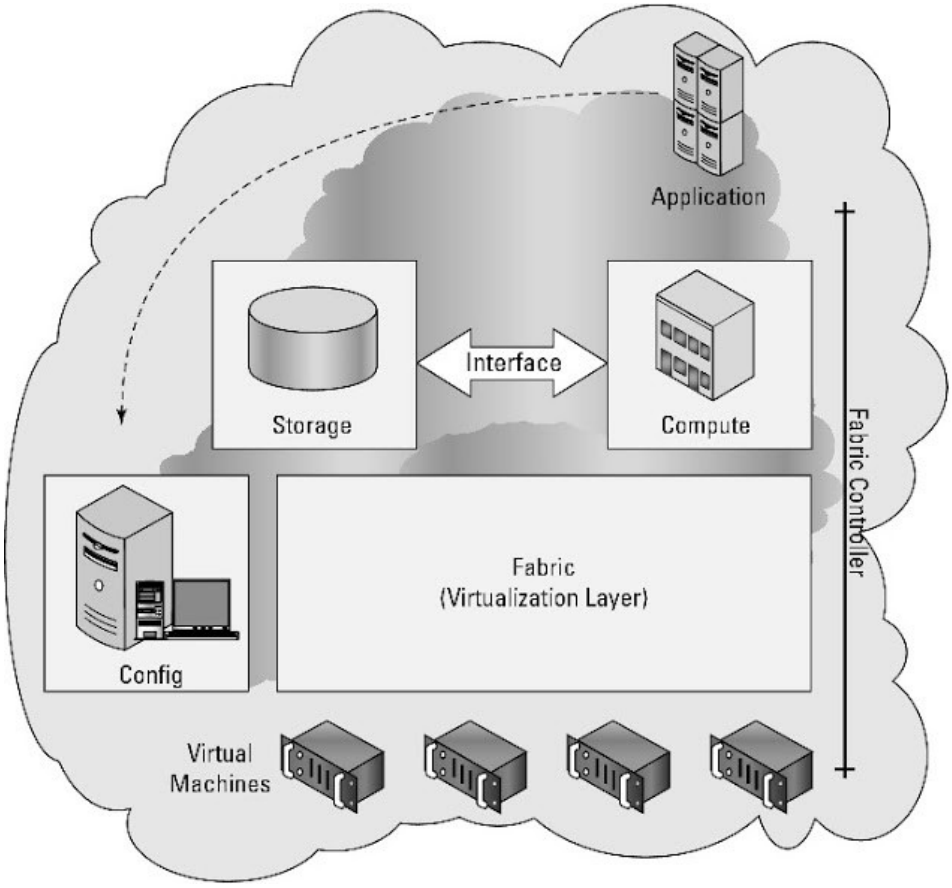


Windows Azure is a virtualized infrastructure that provides configurable virtual machines, independent storage, and a configuration interface. The portion of the Azure environment that creates and manages a virtual resource pool is called the Fabric Controller. Applications that run on Azure are memory-managed, load-balanced, replicated, and backed up through snapshots automatically by the Fabric Controller.

Windows Azure AppFabric

Windows Azure AppFabric provides a comprehensive cloud middleware platform for developing, deploying and managing applications on the Windows Azure Platform. It delivers additional developer productivity, adding in higher-level Platform-as-a-Service (PaaS) capabilities on top of the familiar Windows Azure application model. It also enables bridging your existing applications to the cloud through secure connectivity across network and geographic boundaries, and by providing a consistent development model for both Windows Azure and Windows Server. Finally, it makes development more productive by providing a higher abstraction for building end-to-end applications,

and simplifies management and maintenance of the application as it takes advantage of advances in the underlying hardware and software infrastructure.



Middleware Services: platform capabilities as services, which raise the level of abstraction and reduce complexity of cloud development.

Composite Applications: a set of new innovative frameworks, tools and composition engine to easily assemble, deploy, and manage a composite application as a single logical entity

Scale-out application infrastructure: optimized for cloud-scale services and mid-tier components.

Azure Content Delivery Network

The Windows Azure Content Delivery Network (CDN) is a worldwide content caching and delivery system for Windows Azure blob content. Currently, more than 18 Microsoft datacenters are hosting this service in Australia, Asia, Europe, South America, and the United States, referred to as endpoints. CDN is an edge network service that lowers latency and maximizes bandwidth by delivering content to users who are nearby.

SQL Azure

SQL Azure is a cloud-based relational database service that is based on Microsoft SQL Server. Initially, this service was called SQL Server Data Service. An application that uses SQL Azure Database can run locally on a server, PC, or mobile device, in a datacenter, or on Windows Azure. Data stored in an SQL Azure database is accessed using the Tabular Data Stream (TDS) protocol, the same protocol used for a local SQL Server database. SQL Azure Database supports Transact-SQL statements.

Note: Because SQL Azure is managed in the cloud, there are no administrative controls over the SQL engine. You can't shut the system down, nor can you directly interact with the SQL Servers.

Windows Live Essentials

Windows Live Essentials applications are a collection of client-side applications that must be downloaded and installed on a desktop. Some of these applications were once part of Windows and have been unbundled from the operating system; others are entirely new. Live Essentials rely on cloud-based services for their data storage and retrieval, and in some cases for their processing.

Windows Live Essentials currently includes the following:

Family Safety
Windows Live Messenger
Photo Gallery

Mail

Movie Maker