**Cloud Computing**

**Unit – 1**

**UNIT I: Systems modeling, Clustering and virtualization:**

Scalable Computing over the Internet, Technologies for Network based systems, System models for Distributed and Cloud Computing, Software environments for distributed systems and clouds, Performance, Security And Energy Efficiency.

# SCALABLE COMPUTING OVER THE INTERNET

Over the past 60 years, computing technology has undergone a series of platform and environment changes. In this section, we assess evolutionary changes in machine architecture, operating system platform, network connectivity, and application workload. Instead of using a centralized computer to solve computational problems, a parallel and distributed computing system uses multiple computers to solve large-scale problems over the Internet. Thus, distributed computing becomes data-intensive and network-centric.

# The Age of Internet Computing

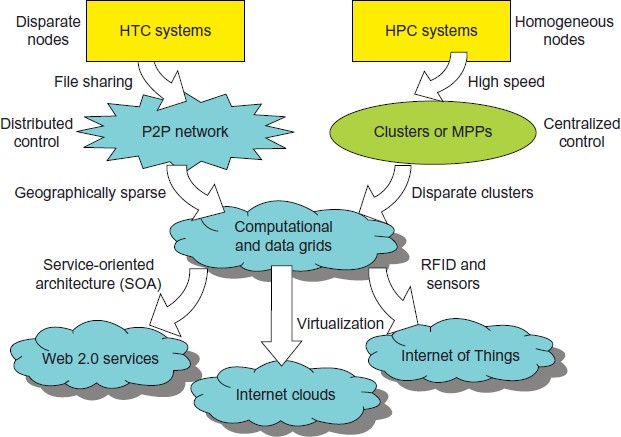
Billions of people use the Internet every day. As a result, supercomputer sites and large data centers must provide high-performance computing services to huge numbers of Internet users concurrently. Because of this high demand, the Linpack Benchmark for high-performance computing (HPC) applications is no longer optimal for measuring system performance.

The emergence of computing clouds instead demands high-throughput computing (HTC) systems built with parallel and distributed computing technologies

# High-Performance Computing

For many years, HPC systems emphasize the raw speed performance. The speed of HPC systems has increased from Gflops in the early 1990s to now Pflops in 2010.

This improvement was driven mainly by the demands from scientific, engineering, and manufacturing communities. For example, the Top 500 most powerful computer systems in the world are measured by floating-point speed in Linpack benchmark results. However, the number of supercomputer users is limited to less than 10% of all computer users. Today, the majority of computer users are using desktop computers or large servers when they conduct Internet searches and market-driven computing tasks.



# High-Throughput Computing

The development of market-oriented high-end computing systems is undergoing a strategic change from an HPC paradigm to an HTC paradigm.

This HTC paradigm pays more attention to high-flux computing. The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously.

The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time. HTC technology needs to not only improve in terms of batch processing speed, but also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers. This book will address both HPC and HTC systems to meet the demands of all computer users.

# Three New Computing Paradigms

With the introduction of SOA, Web 2.0 services become available. Advances in virtualization make it possible to see the growth of Internet clouds as a new computing paradigm. The maturity of radio-frequency identification (RFID), Global Positioning System (GPS), and sensor technologies has triggered the development of the Internet of Things (IoT).

When the Internet was introduced in 1969, Leonard Klienrock of UCLA declared: “As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of computer utilities, which like present electric and telephone utilities, will service individual homes and offices across the country.” Many people have redefined the term “computer” since that time. In 1984, John Gage of Sun Microsystems created the slogan, “The network is the computer.”

In 2008, David Patterson of UC Berkeley said, “The data center is the computer. There are dramatic differences between developing software for millions to use as a service versus distributing software to run on their PCs.” Recently, Rajkumar Buyya of Melbourne University simply said: “The cloud is the computer.”

# Computing Paradigm Distinctions

The high-technology community has argued for many years about the precise definitions of centralized computing, parallel computing, distributed computing, and cloud computing. In general, distributed computing is the opposite of centralized computing.

The field of parallel computing overlaps with distributed computing to a great extent, and cloud computing overlaps with distributed, centralized, and parallel computing.

* Centralized computing This is a computing paradigm by which all computer resources are centralized in one physical system. All resources (processors, memory, and storage) are fully shared and tightly coupled within one integrated OS. Many data centers and supercomputers are centralized systems, but they are used in parallel, distributed, and cloud computing applications.
* Parallel computing In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Some authors refer to this discipline as parallel processing. Interprocessor communication is accomplished through shared memory or via message passing.

A computer system capable of parallel computing is commonly known as a parallel computer. Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming.

* Distributed computing This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network.

Information exchange in a distributed system is accomplished through message passing. A computer program that runs in a distributed system is known as a distributed program. The process of writing distributed programs is referred to as distributed programming.

* Cloud computing An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed. Some authors consider cloud computing to be a form of utility computing or service computing.

# Distributed System Families

Since the mid-1990s, technologies for building P2P networks and networks of clusters have been consolidated into many national projects designed to establish wide area computing infrastructures, known as computational grids or data grids.

Meeting these goals requires to yield the following design objectives:

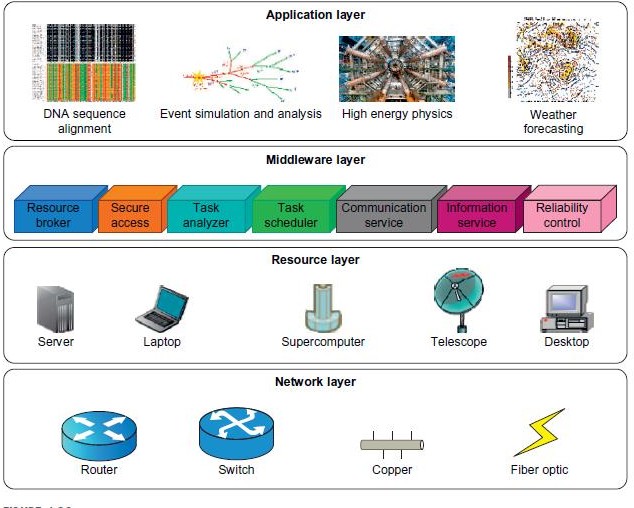
* Efficiency measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, data access, storage, and power efficiency.
* Dependability measures the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.
* Adaptation in the programming model measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.
* Flexibility in application deployment measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

# Degrees of Parallelism

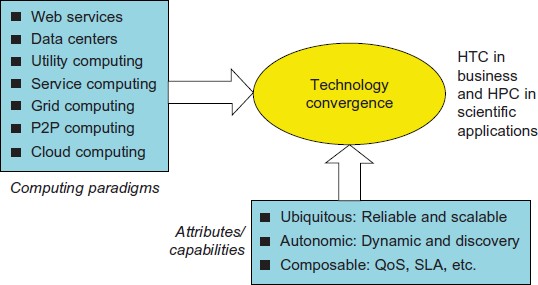
Fifty years ago, when hardware was bulky and expensive, most computers were designed in a bit-serial fashion. In this scenario, bit-level parallelism (BLP) converts bit-serial processing to word-level processing gradually. Over the years, users graduated from 4-bit microprocessors to 8-,16-, 32-, and 64-bit CPUs. This led us to the next wave of improvement, known as instruction-level parallelism (ILP), in which the processor executes multiple instructions simultaneously rather than only one instruction at a time.

For the past 30 years, we have practiced ILP through pipelining, superscalar computing, VLIW (very long instruction word) architectures, and multithreading. ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently. Data-level parallelism (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly.

Ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring task-level parallelism (TLP). A modern processor explores all of the aforementioned parallelism types. In fact, BLP, ILP, and DLP are well supported by advances in hardware and compilers. However, TLP is far from being very successful due to difficulty in programming and compilation of code for efficient execution on multicore CMPs.



Utility computing focuses on a business model in which customers receive computing resources from a paid service provider. All grid/cloud platforms are regarded as utility service providers. However, cloud computing offers a broader concept than utility computing. Distributed cloud applications run on any available servers in some edge networks. Major technological challenges include all aspects of computer science and engineering.



# The Internet of Things

The concept of the IoT was introduced in 1999 at MIT [40]. The IoT refers to the networked interconnection of everyday objects, tools, devices, or computers. One can view the IoT as a wireless network of sensors that interconnect all things in our daily life. These things can be large or small and they vary with respect to time and place. The idea is to tag every object using RFID or a related sensor or electronic technology such as GPS.

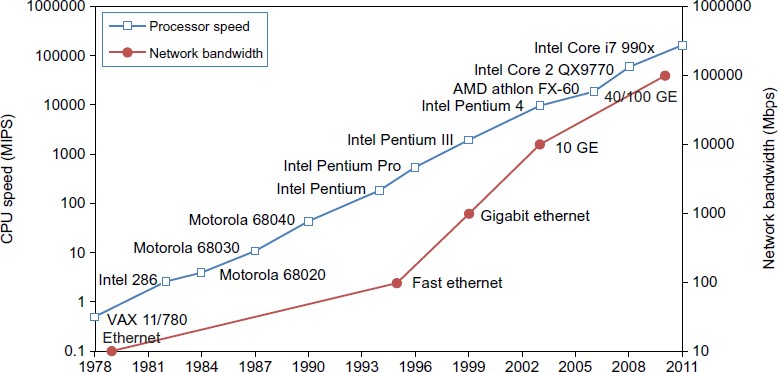
# TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

**Multicore CPUs and Multithreading Technologies**

Consider the growth of component and network technologies over the past 30 years. They are crucial to the development of HPC and HTC systems. In Figure 1.4, processor speed is measured in millions of instructions per second (MIPS) and network bandwidth is measured in megabits per second (Mbps) or gigabits per second (Gbps). The unit GE refers to 1 Gbps Ethernet bandwidth.

# Advances in CPU Processors

Advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels. Processor speed growth is plotted in the upper curve in across generations of microprocessors or CMPs.



# Multicore CPU and Many-Core GPU Architectures

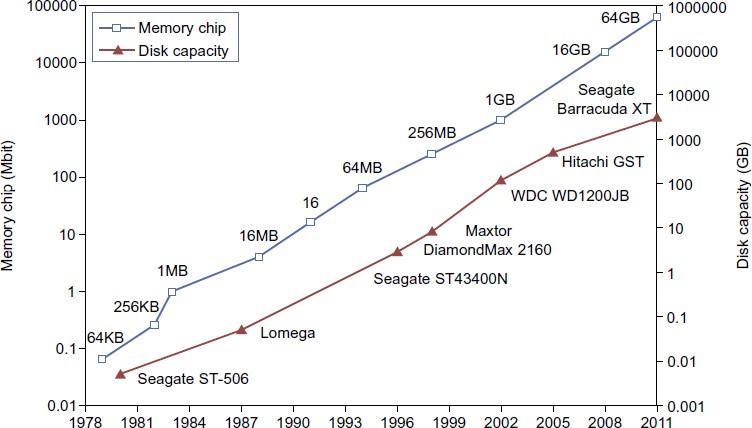
Multicore CPUs may increase from the tens of cores to hundreds or more in the future. But the CPU has reached its limit in terms of exploiting massive DLP due to the aforementioned memory wall problem. This has triggered the development of many-core GPUs with hundreds or more thin cores. Both IA-32 and IA- 64 instruction set architectures are built into commercial CPUs. Now, x-86 processors have been extended to serve HPC and HTC systems in some high-end server processors.

# Multithreading Technology & How GPUs Work Memory, Storage, and Wide-Area Networking Memory Technology

The growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011. This shows that memory chips have experienced a 4x increase in capacity every three years. Memory access time did not improve much in the past. In fact, the memory wall problem is getting worse as the processor gets faster. For hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004. The Seagate Barracuda XT hard drive reached 3 TB in 2011.

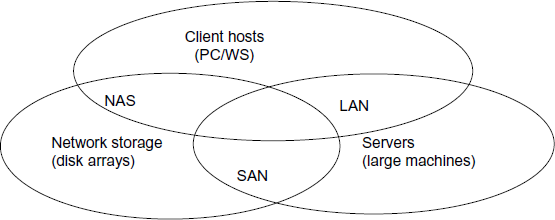
# Disks and Storage Technology

Beyond 2011, disks or disk arrays have exceeded 3 TB in capacity. The lower curve in Figure shows the disk storage growth in 7 orders of magnitude in 33 years. The rapid growth of flash memory and solid-state drives (SSDs) also impacts the future of HPC and HTC systems. The mortality rate of SSD is not bad at all. A typical SSD can handle 300,000 to 1 million write cycles per block. So the SSD can last for several years, even under conditions of heavy write usage. Flash and SSD will demonstrate impressive speedups in many applications.



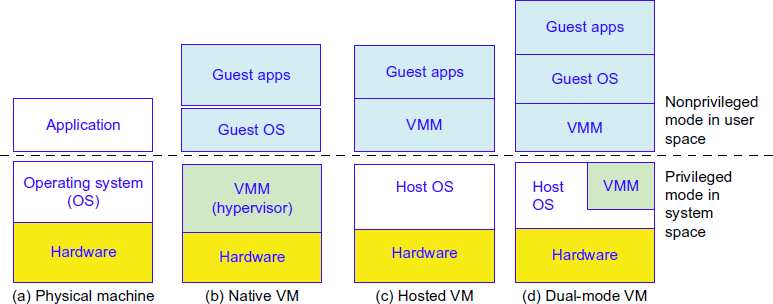
# System-Area Interconnects

The nodes in small clusters are mostly interconnected by an Ethernet switch or a local area network (LAN). As Figure 1.11 shows, a LAN typically is used to connect client hosts to big servers. A storage area network (SAN) connects servers to network storage such as disk arrays. Network attached storage (NAS) connects client hosts directly to the disk arrays. All three types of networks often appear in a large cluster built with commercial network components.



# Virtual Machines and Virtualization Middleware

A conventional computer has a single OS image. This offers a rigid architecture that tightly couples application software to a specific hardware platform. Some software running well on one machine may not be executable on another platform with a different instruction set under a fixed OS. Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.



# Virtual Machines

The host machine is equipped with the physical hardware, as shown at the bottom of the figure. An example is an x-86 architecture desktop running its installed Windows OS, as shown in part (a) of the figure.

The VM can be provisioned for any hardware system. The VM is built with virtual resources managed by a guest OS to run a specific application. Between the VMs and the host platform, one needs to deploy a middleware layer called a virtual machine monitor (VMM). Figure 1.12(b) shows a native VM installed with the use of a VMM called a hypervisor in privileged mode. For example, the hardware has x-86 architecture running the Windows system.

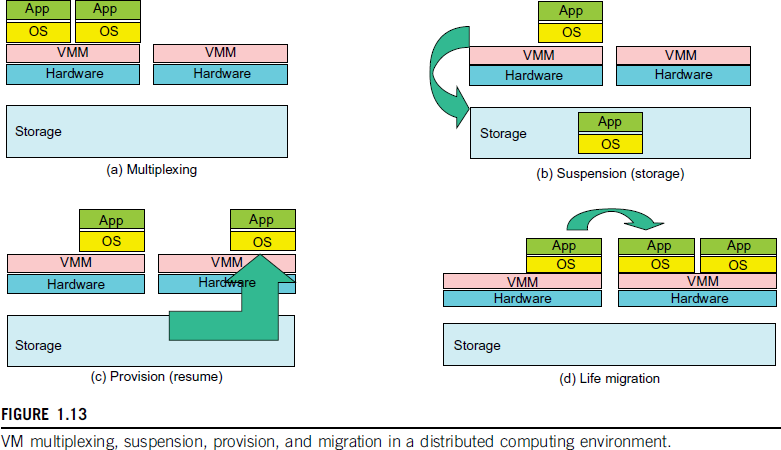
The guest OS could be a Linux system and the hypervisor is the XEN system developed at Cambridge University. This hypervisor approach is also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly. Another architecture is the host VM shown in Figure 1.12(c). Here the VMM runs in nonprivileged mode. The host OS need not be modified. The VM can also be implemented with a dual mode, as shown in Figure 1.12(d).

Part of the VMM runs at the user level and another part runs at the supervisor level. In this case, the host OS may have to be modified to some extent. Multiple VMs can be ported to a given hardware system to support the virtualization process. The VM approach offers hardware independence of the OS and applications. The user application running on its dedicated OS could be bundled together as a virtual

appliance that can be ported to any hardware platform. The VM could run on an OS different from that of the host computer.

# VM Primitive Operations

The VMM provides the VM abstraction to the guest OS. With full virtualization, the VMM exports a VM abstraction identical to the physical machine so that a standard OS such as Windows 2000 or Linux can run just as it would on the physical hardware. Low-level VMM operations are indicated by Mendel Rosenblum [41] and illustrated in Figure 1.13.



First, the VMs can be multiplexed between hardware machines, as shown in Figure (a). Second, a VM can be suspended and stored in stable storage, as shown in Figure (b).

Third, a suspended VM can be resumed or provisioned to a new hardware platform, as shown in Figure (c).

Finally, a VM can be migrated from one hardware platform to another, as shown in Figure (d).

# Virtual Infrastructures

Physical resources for compute, storage, and networking at the bottom of Figure are mapped to the needy applications embedded in various VMs at the top. Hardware and software are then separated. Virtual infrastructure is what connects resources to distributed applications. It is a dynamic mapping of system resources to specific applications. The result is decreased costs and increased efficiency and responsiveness. Virtualization for server consolidation and containment is a good example of this.

# Data Center Virtualization for Cloud Computing Data Center Growth and Cost Breakdown

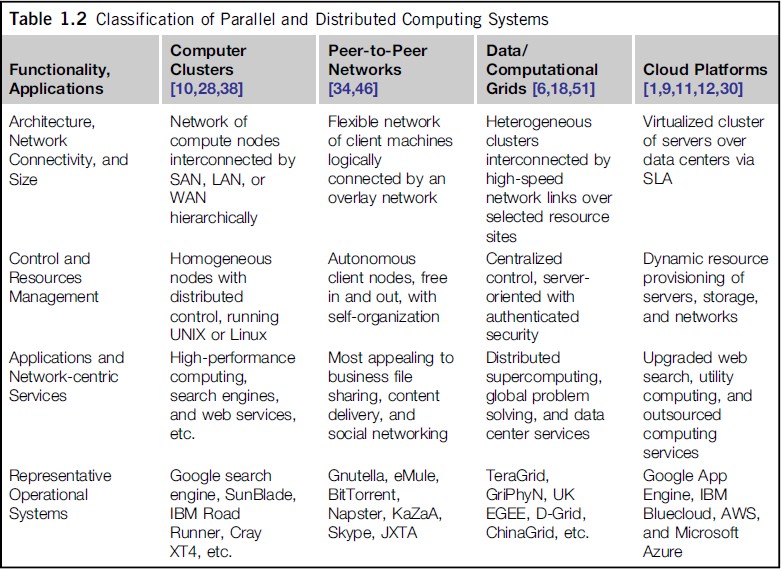
A large data center may be built with thousands of servers. Smaller data centers are typically built with hundreds of servers. The cost to build and maintain data center servers has increased over the years. According to a 2009 IDC report, typically only 30 percent of data center costs goes toward purchasing IT equipment (such as servers and disks), 33 percent is attributed to the chiller, 18 percent to the uninterruptible power supply (UPS), 9 percent to computer room air conditioning (CRAC), and the remaining 7 percent to power distribution, lighting, and transformer costs. Thus, about 60 percent of the cost to run a data center is allocated to management and maintenance. The server purchase cost did not

increase much with time. The cost of electricity and cooling did increase from 5 percent to 14 percent in 15 years.

# SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner. With today’s networking technology, a few LAN switches can easily connect hundreds of machines as a working cluster. A WAN can connect many local clusters to form a very large cluster of clusters. In this sense, one can build a massive system with millions of computers connected to edge networks.

Massive systems are considered highly scalable, and can reach web-scale connectivity, either physically or logically. In Table 1.2, massive systems are classified into four groups: clusters, P2P networks, computing grids, and Internet clouds over huge data centers. In terms of node number, these four system classes may involve hundreds, thousands, or even millions of computers as participating nodes.



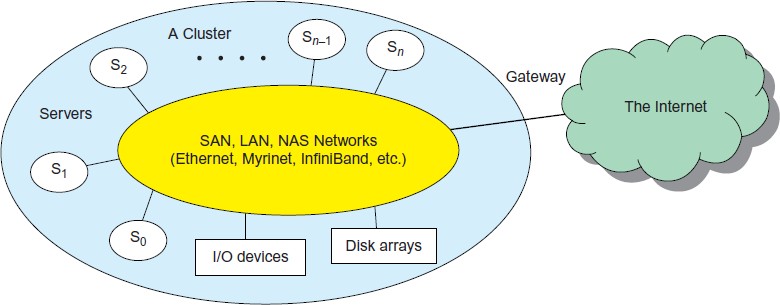
# Clusters of Cooperative Computers

A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource. In the past, clustered computer systems have demonstrated impressive results in handling heavy workloads with large data sets.

# Cluster Architecture

Figure 1.15 shows the architecture of a typical server cluster built around a low-latency, high bandwidth interconnection network. This network can be as simple as a SAN (e.g., Myrinet) or a LAN (e.g., Ethernet). To build a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to

the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.



# Single-System Image

Greg Pfister [38] has indicated that an ideal cluster should merge multiple system images into a single- system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.

# Hardware, Software, and Middleware Support

Clusters exploring massive parallelism are commonly known as MPPs. Almost all HPC clusters in the Top 500 list are also MPPs. The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM or MPI, and a network interface card in each computer node. Most clusters run under the Linux OS. The computer nodes are interconnected by a high-bandwidth network (such as Gigabit Ethernet, Myrinet, InfiniBand, etc.). Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources.

# Grid Computing Infrastructures

Internet services such as the Telnet command enables a local computer to connect to a remote computer. A web service such as HTTP enables remote access of remote web pages. Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

# Computational Grids

Like an electric utility power grid, a computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.

# Grid Families

Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, and others. New grid

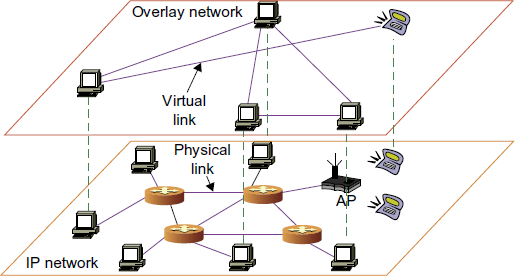
service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades.

# Peer-to-Peer Network Families

An example of a well-established distributed system is the client-server architecture. In this scenario, client machines (PCs and workstations) are connected to a central server for compute, e-mail, file access, and database applications. The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented. In this section, P2P systems are introduced at the physical level and overlay networks at the logical level.

# Overlay Networks

Data items or files are distributed in the participating peers. Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level.



Cloud Computing over the Internet

“A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications.” Based on this definition, a cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines. The cloud supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures. Finally, the cloud system should be able to monitor resource use in real time to enable rebalancing of allocations when needed.

Internet Clouds

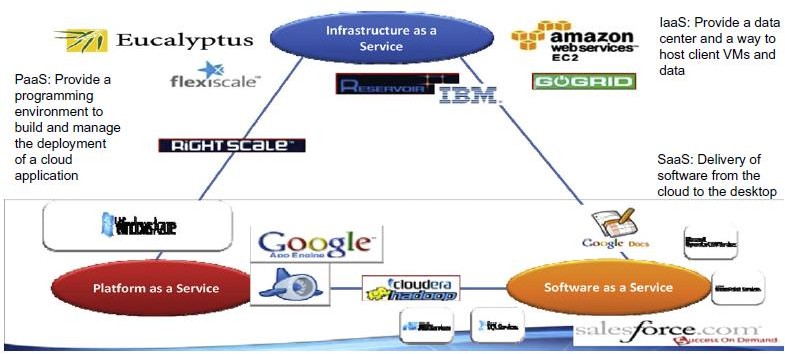
Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically. The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers. Cloud computing leverages its low cost and simplicity to benefit both users and providers.

The Cloud Landscape

Traditionally, a distributed computing system tends to be owned and operated by an autonomous administrative domain (e.g., a research laboratory or company) for on-premises computing needs. However, these traditional systems have encountered several performance bottlenecks: constant system

maintenance, poor utilization, and increasing costs associated with hardware/software upgrades. Cloud computing as an on-demand computing paradigm resolves or relieves us from these problems.

* Infrastructure as a Service (IaaS) This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.
* Platform as a Service (PaaS) This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java. The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.
* Software as a Service (SaaS) This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.



# SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS

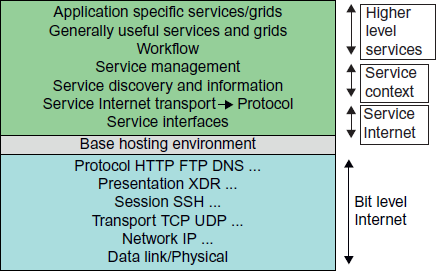
**Service-Oriented Architecture (SOA)**

In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide the base networking abstractions. On top of this we have a base software environment, which would be .NET or Apache Axis for web services, the Java Virtual Machine for Java, and a broker network for CORBA. On top of this base environment one would build a higher level environment reflecting the special features of the distributed computing environment.

# Layered Architecture for Web Services and Grids

The entity interfaces correspond to the Web Services Description Language (WSDL), Java method, and CORBA interface definition language (IDL) specifications in these example distributed systems. These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP in the three examples. These communication systems support features including particular message patterns (such as Remote Procedure Call or RPC), fault recovery, and specialized routing. Often, these communication systems are built on message-oriented middleware (enterprise bus) infrastructure such as Web- Sphere MQ or Java Message Service (JMS) which provide rich functionality and support virtualization of routing, senders, and recipients.

In the case of fault tolerance, the features in the Web Services Reliable Messaging (WSRM) framework mimic the OSI layer capability (as in TCP fault tolerance) modified to match the different abstractions (such as messages versus packets, virtualized addressing) at the entity levels. Security is a critical capability that either uses or reimplements the capabilities seen in concepts such as Internet Protocol Security (IPsec) and secure sockets in the OSI layers. Entity communication is supported by higher level services for registries, metadata, and management.



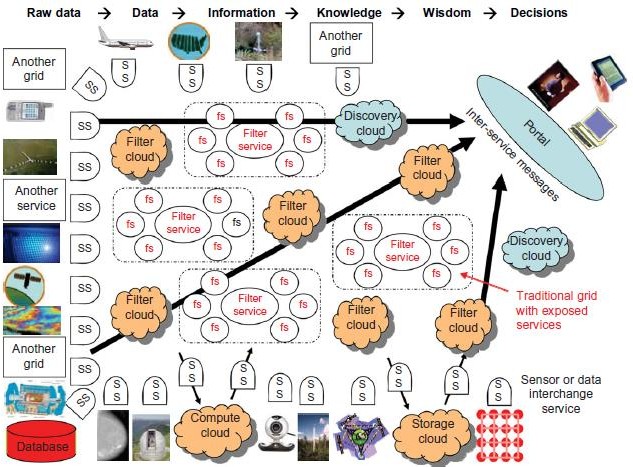
# Web Services and Tools

Loose coupling and support of heterogeneous implementations make services more attractive than distributed objects. The above picture corresponds to two choices of service architecture: web services or REST systems. Both web services and REST systems have very distinct approaches to building reliable interoperable systems. In web services, one aims to fully specify all aspects of the service and its environment. This specification is carried with communicated messages using Simple Object Access Protocol (SOAP). The hosting environment then becomes a universal distributed operating system with fully distributed capability carried by SOAP messages. This approach has mixed success as it has been hard to agree on key parts of the protocol and even harder to efficiently implement the protocol by software such as Apache Axis.

# The Evolution of SOA

As shown in Figure 1.21, service-oriented architecture (SOA) has evolved over the years. SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds), and systems of systems in general. A large number of sensors provide data-collection services, denoted in the

figure as SS (sensor service). A sensor can be a ZigBee device, a Bluetooth device, a WiFi access point, a personal computer, a GPA, or a wireless phone, among other things. Raw data is collected by sensor services. All the SS devices interact with large or small computers, many forms of grids, databases, the compute cloud, the storage cloud, the filter cloud, the discovery cloud, and so on. Filter services ( fs in the figure) are used to eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.



**Introduction to Grid Architecture and standards**

## Grid Architecture

## Grid architecture is a discipline with roots in system architecture, network theory, control engineering, and software architecture, all of which we apply to the electric power grid. An architectural description is a structural representation of a system that helps people think about the overall shape of the system, its attributes, and how the parts interact.

## Grid architecture serves many purposes, including the following:

## managing grid modernization complexity and assisting stakeholders in communicating about the grid

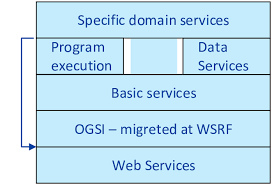
## identifying and removing structural barriers and defining essential limits to grid behavior

## creating new structure to enable new capabilities or strengthen grid properties such as resilience

## identifying gaps in theory, technology, and organization

## providing a framework for grid modernization activities.

## Most of all, grid architecture provides insight to stakeholders so they can make informed decisions about grid modernization.



Open Grid Services Architecture (OGSA) is a set of standards that extends Web services and service-oriented architecture to the grid computing environment. OGSA definitions and criteria describe how information is shared and distributed among the components of large, heterogeneous grid systems; they apply to hardware, platforms and software.

Grid computing has intrigued the IT world for years. The notion of harnessing the processing power of multiple computers whether within an organization, supplied by volunteers or provided as a broadband, metered computing utility is attractive and compelling, but implementing it has proved somewhat difficult. One of the more recent developments advancing this cause, OGSA extends the idea of Web services to the universe of grid computing and thereby extends and refines the concept of service-oriented architecture.

To create workable grid services, developers have had to address several important issues: how to establish identity and negotiate authentication, how to express and negotiate policy, how to find out what services are available, how to negotiate and monitor service-level agreements, how to organize and manage collections of services to deliver reliable and scalable services, and how to integrate data resources into computations.

OGSA is based primarily on the technologies of Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP), an XML-based protocol for passing messages between systems over the Internet. It is service- oriented because it works as a series of loosely coupled, interacting services that use industry-accepted Web services standards.

For a Web service to be considered a grid service, clients must be able to easily discover, update, modify and delete information about the service and its functionality and relevant data; define how the service evolves; and ensure ongoing compatibility with other services.

**GRID STANDARDS**

**A. OGSA (Open Grid Service Architecture)**

The aim of OGSA is to standardize grid computing and to define a basic framework of a grid application structure. Some of the key concepts are first presented by Ian Foster who still leads the OGSA working group. This Architecture combines different aspects from grid computing with advantages from Web Services. The Global Grid Forum published two main guidelines, the Open Grid Service Architecture and the Open Grid Service Infrastructure. OGSA represents service oriented architecture. It defines mechanisms for creating, managing and exchanging information among grid services. The architecture is based on Grid Services.

A Grid Service is defined by a Web Service that provides a set of well-defined interfaces and that follows specific conventions. This was a logical choice, as similar problems arise in both worlds, such as distributed service discovery and use. OGSA was first presented in 1998, but it lasted until June 2004 that OGSA 1.0 was released. Momentary the working group is building version 2.0 which is expected to be finished in summer 2005.

OGSA main goals are : -

* Resources must be handled in distributed and heterogeneous environments
* Support of QoS-orientated (Quality of Service) Service Level Agreement
* Partly autonomic management
* Definition of open, published interfaces and protocols that provide interoperability of diverse resources
* Integration of existing and established standards

**B. OGSA Services**

The OGSA specifies services which occur within a wide variety of grid systems. They can be divided into 4 broad groups: core services, data services, program execution services, and resource management services.

Core Services:

Service Communication This category includes different services which handle communication among services. Including distributed logging, messaging, and events.

Service Management Includes several subservices: installation, deployment, and provisioning; fault management; problem determination; and metering and accounting.

Service Interaction This category includes services to provide interaction mechanisms between services in the grid. Including: Virtual Organisations; service group and discovery services; transactions; service domain, composition, orchestration; and workflow.

Security This service handles right management within virtual organisations. Several aspects must be addressed, as: authentication, authorization, confidentiality, message integrity, policy expression and exchange, delegation, single sign-on, firewall-traversal, credential lifespan and renewal, privacy, secure logging, assurance, manageability, and security at the OGSI layer.The VOs are seen as bridge between these two.

**Grid Computing Standards 3 Data Services**: The wide range of different data types, usability and transparency involve a large variety of different interfaces:

* Interfaces for caching
* Interfaces for data replication
* Interfaces for data access
* Interfaces for data transformation and filtering
* Interfaces for file and DBMS services
* Interfaces for grid storage services

Program Execution: Main goal of this category is to enable applications to have coordinated access to underlying VO resources, regardless of their physical location or access mechanisms. For this purpose a variety of factory services is used. Execution Services support task planning, observation and management of workflows.

Resource Management: Resources need to be reserved and scheduled, orchestrated, and controlled. This group also maintains administration and deployment services for software deployment, change and identity management. Despite Web Services where are good choice there are still several problems which must be solved. OGSA’s most important requirement is an underlying middleware that is ‘state full’. Web Services can be either state full or stateless, but there was no standard way to make them state full. This is why another standard was created: OGSI.

C. OGSI (Open Grid Service Infrastructure)

OGSA defines a Grid Application and what a Grid Service should be able to do. OGSI specifies a Grid Services in detail. As mentioned, Grid Services are Web Services with special additions:

Lifecycle Management

Service Data:

o State informations

o Service metadata

Notifications

Service Groups

PortType Extensions

GSH (Grid Service Handle) & GSR (Grid Service Reference)

The OGSI has a “focus on technical details, providing a full specification of the behaviour and Web Service Definition Language (WSDL) interfaces that define a Grid Service” [17]. In figure 1 you can see how all different parts fit together.

D. WSRF (Web Service Resource Framework) WSRF is a derivative of OGSI.

A first implementation can be found in GT4. The framework combines 6 different WS specifications “that define what is termed the WS-Resource approach to modeling and managing state in a Web services context”. Grid Computing Standards 4 WSRF describes how resources can be handled by Web Services. The reason why WSRF was necessary is that OGSI was never fully accepted by the WS-Community out of several reasons: - OGSI does not work well with existing WS and XML tooling - OGSI is too object oriented, WS are not, WS are stateless and OGSI concepts don’t match with former WS specifications, - OGSI has too much stuff for one specification. The big difference is that the OGSA layer can now work directly on WS and is not dependent on “special” WS. WSRF specifications

WS-ResourceLifetime: mechanisms for WSRessource destruction

WS-ResourceProperties: manipulation and definition of WS properties

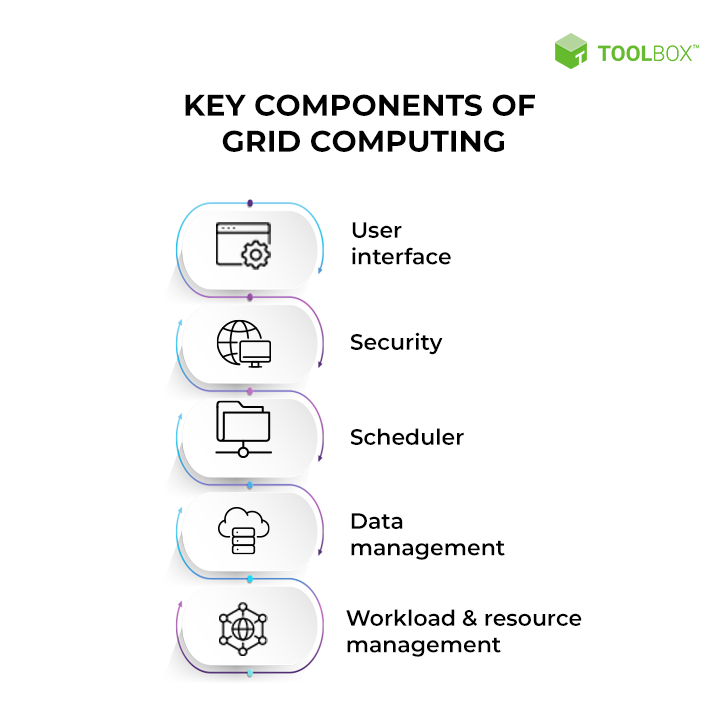
WS-Notification: event management

WS-RenewableReference: defines updating proceeding

WS-ServiceGroup: interface for by-reference collections of WSs

WS-BaseFaults: standardization of possible failures

**Elements of Grid:**



**1. User interface**

Today, users are well-versed with web portals. They provide a single interface that allows users to view a wide variety of information. Similarly, a grid portal offers an interface that enables users to launch applications with resources provided by the grid.

The interface has a portal style to help users query and execute various functions on the grid effectively. A grid user views a single, large virtual computer offering computing resources, similar to an internet user who views a unified instance of content on the web.

**2. Security**

Security is one of the major concerns for grid computing environments. Security mechanisms can include authentication, authorization, data encryption, and others. Grid security infrastructure (GSI) is an important ingredient here. It outlines specifications that establish secret and tamper-proof communication between software entities operating in a grid network.

It includes OpenSSL implementation and provides[a single sign-on mechanism](https://www.spiceworks.com/it-security/vulnerability-management/articles/what-is-single-sign-on/) for users to perform actions within the grid. It offers robust security by providing authentication and authorization mechanisms for system protection.

**3. Scheduler**

On identifying the resources, the next step is to schedule the tasks to run on them. A scheduler may not be needed if standalone tasks are to be executed that do not showcase interdependencies. However, if you want to run specific tasks concurrently that require inter-process communication, the job scheduler would suffice to coordinate the execution of different subtasks.

Moreover, schedulers of different levels operate in a grid environment. For example, a cluster may represent an independent resource with its own scheduler to manage the nodes it contains. Hence, a high-level scheduler may sometimes be required to accomplish the task done on the cluster, while the cluster employs its own separate scheduler to handle work on its individual nodes.

**4. Data management**

Data management is crucial for grid environments. A secure and reliable mechanism to move or make any data or application module accessible to various nodes within the grid is necessary. Consider the Globus toolkit — an open-source toolkit for grid computing.

It offers a data management component called grid access to secondary storage (GASS). It includes GridFTP built on the standard FTP protocol and utilizes GSI for[user authentication](https://www.spiceworks.com/it-security/identity-access-management/articles/what-is-two-factor-authentication) and authorization. After authentication, the user can move files using the GridFTP facility without going through the login process at every node.

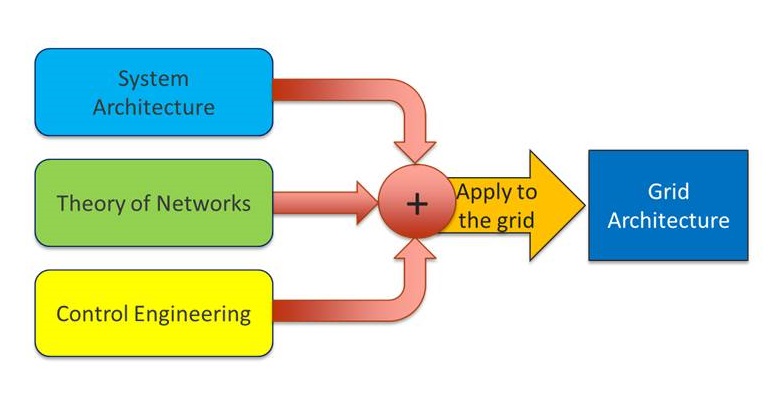
**5. Workload & resource management**

The workload & resource component enables the actual launch of a job on a particular resource, checks its status, and retrieves the results when the job is complete. Say a user wants to execute an application on the grid. In that case, the application should be aware of the available resources on the grid to take up the workload. So, it interacts with the workload manager to determine the resource availability and updates the status accordingly. This helps in efficient workload and resource management for various nodes on the grid.

**Overview of** [**Grid Architecture**](https://www.pnnl.gov/grid-architecture)

It is the application of system architecture, network theory, and control theory to the electric power grid. A grid architecture is the highest level description of the complete grid, and is a key tool to help understand and define the many complex interactions that exist in present and future grids.

– It is the way where grid has been designed.  
– It is described in terms of layers which has a specific function.  
– The lowest layer is the network above the network layer lies the resource layer.  
– The middle layer provides tools that enable various elements.  
– The highest is the application layer which includes all kinds of applications



## Prime Uses of Grid Architecture

* Help manage complexity (and therefore risk)
* Assist communication among stakeholders around a shared vision of the future grid
* Identify and remove barriers and define essential limits
* Identify gaps in theory, technology, organization, regulation
* Provide a framework for complex grid-related development activities