What is Cluster Computing?

- A **computer cluster** consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system.
- Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.

1. Scalable Parallel Computer Architectures

- In past decades many high performance computing have emerged.
- Their classification is based on how their processors, memory and interconnect are laid out.
- The most common parallel computing architectures are as following
  - Massively parallel processors (MPP)
  - Symmetric multiprocessors (SMP)
  - Cache-coherent non-uniform memory access (CC-NUMA)
  - Distributed Systems
  - Clusters

Characteristics	MPP	SMP with CC NUMA	Cluster	Distributed
No of Nodes	100-1000	10-100	100 of Less	10-1000
Node complexity	Fine or Medium grained	Medium or coarse grained	Medium grain	Wide range
Internode communication	MPI or DSM	Centralized and Distributed	MPI	Shared files and RPC and MPI
Job scheduling	Single run queue	Single run queue	multiple queues	Multiple queues
SSI support	Partially	Always in SMP and sometimes in NUMA	Desired	No
Node OS copies and Type	Multiple	One monolithic	N OS platforms heterogeneous	N OS heterogeneous
Address Space	Multiple	Single	Multiple or Single	Multiple
Internode security	Not Necessary	Not Necessary	Necessary	Necessary

Table 1.1 Key characteristics of Scalable parallel computing architectures

2. A Cluster Computer and its Architecture

- A **cluster** is a type of parallel or distributed processing system.
- It consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource.

- It is a collection of computers or workstations working together as a single, integrated computing resource connected via high speed interconnects.

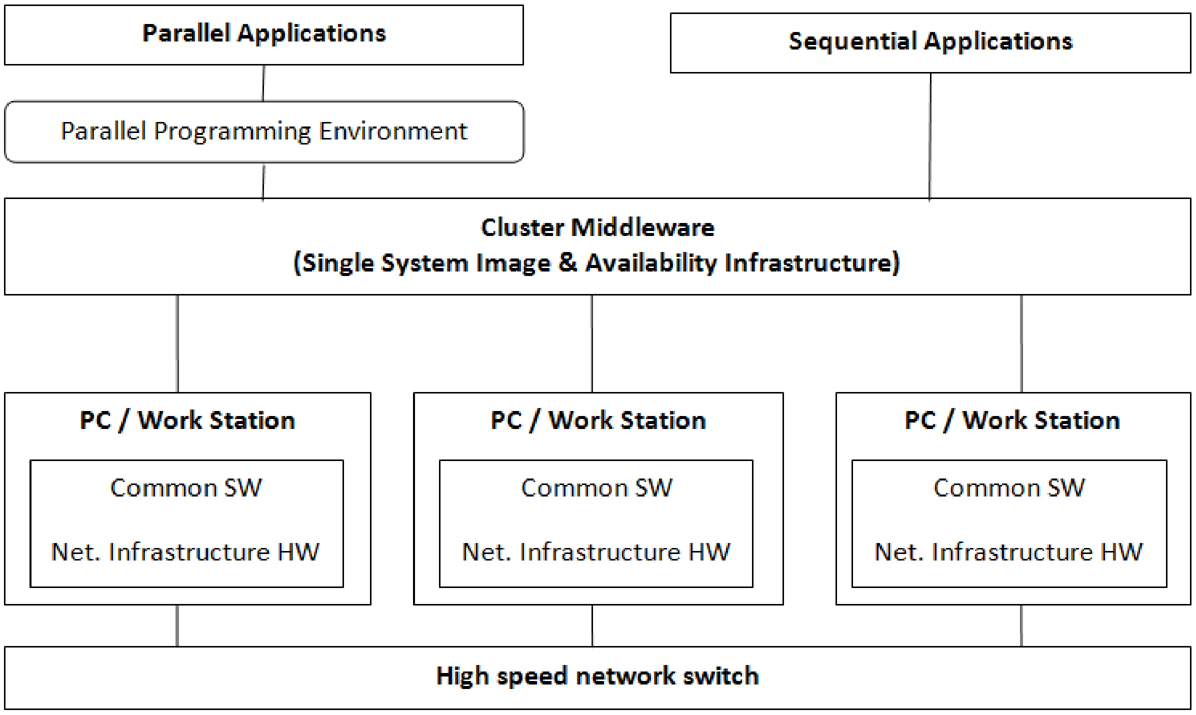


Figure 1.1 Cluster computer architecture

Followings are the **components of a cluster computer:**

- **High performance computers:** These can be PCs, workstations, SMP. These nodes generally have memory, I/O facility and operating system.
- **Modern operating system:** OS should be layered or micro kernel based. Typical OS used in clusters are Linux, Solaris, Windows NT.
- **High speed networks/Switches:** Interconnects as Gigabit Ethernet, ATM etc are suitable and high performance switches are used for connections.
- **Network Interface hardware** is generally various NICs which is an interface to network.
- **Communication Software** includes various fast communication protocols and services as AM, FM which is used for speedy communication between nodes.
- **Cluster Middleware:** its basic aim is to provide SSI and SAI services to the user. It includes
  - Hardware like Hardware DSM, Memory channel etc.
  - Operating system kernel also called the gluing layer like Solaris MC, GLUnix etc.
  - Applications and subsystem like Software DSM, File systems Management tools, RMS software etc
- **Parallel Programming environment and Tools:** These provide portable, efficient and easy to use tools to work in parallel environment and run applications simultaneously. E.g. PVM, MPI etc.
- **Applications:** Sequential as well as parallel applications can be run on a cluster.

3. Cluster Classification

- Clusters are classified into many categories based on various factors as indicated below.

- **Application target**
  - Computational science or mission-critical applications.
  - High performance(HP) clusters
  - High Availability(HA) clusters

- **Node ownership**
  - Owned by individual or dedicated as a cluster node.
  - Dedicated clusters
  - Non-dedicated clusters

- **Node hardware**
  - PC, Workstation, or SMP
  - Clusters of PCs (CoPs) or Piles of PCs (PoPs)
  - Clusters of Workstations (COWs)
  - Clusters of SMPs (CLUMPs)

- **Node operating system**
  - Linux, NT, Solaris, AIX etc.
  - Linux Clusters (Ex: Beowulf)
  - Solaris Clusters (Ex: Berkley NOW)
  - NT Clusters (Ex: HPVM)
  - AIX Clusters (Ex: IBM SP2)
  - Digital VMS Clusters
  - HP-UX Clusters
  - Microsoft wolfpack clusters

- **Node configuration**
  - Node architecture and type of OS it is loaded with.
  - Homogeneous clusters: Simil
  - Heterogeneous clusters

- **Levels of clustering**
  - Based on location of nodes and their count
  - Group clusters: Nodes are connected through SAN (System area networks)
  - Departmental clusters: 10s to 100s nodes
  - Organizational clusters: more than 100s node
  - National meta-computers (WAN/Internet-based)
  - International meta-computers (Internet-based)

4. Commodity Components for Clusters

- Followings are the components which are commonly used to build clusters and nodes.

- **Processors**
  - Computer systems based on these processors have been used to form clusters.
  - For example, Berkeley NOW uses Sun's SPARC family of processors in their cluster nodes.
  - Other popular examples of processors includes x86 variants (AMD x86, Cyrix x86), Digital Alpha IBM PowerPC, Sun SPARC, SGI MIPS, and HP PA.

- **Memory and Cache**
  - Nodes can use various types of memory and they include extended data out(EDO) and fast page.
  - EDO allows the next access to begin while previous data is still being read.
  - Fast page allows multiple adjacent accesses to be made more efficiently.
  - Caches are used to keep recently used blocks of memory for very fast access if the CPU references a word from that block again.

- **Disk and I/O**
  - One way of improving I/O performance is to carry out I/O operations in parallel.
  - It can be done by parallel file systems based on hardware or software RAID.
  - Hardware RAIDs can be expensive, Software RAID can be constructed by using disks associated with each workstation in the cluster.

- **System Bus**
  - The initial PC bus known as ISA bus bas worked with 16 bits wide and clocked in excess of 13 MHz.
  - It was not sufficient to meet the demand of the latest CPUs, disk interfaces and other peripherals.
  - Group of PC manufacturers introduced the VESA local bus, 32-bit bus that matched the system's clock speed.

- **Cluster Interconnects**
  - Ethernet, Fast Ethernet, and Gigabit Ethernet
  - Asynchronous transfer mode (ATM)
  - Scalable coherent interface (SCI)

- **Operating systems**
  - LINUX
  - Solaris
  - Microsoft window NT

5. Cluster middleware and Single System Image (SSI)

- Single system image (SSI) means collection of interconnected computers designed to appear as a unified resource.
- The SSI is supported by a middleware layer.
- Middleware layer resides between operating system and user-level environment.
- Middleware consists of two sub layers of software infrastructure.
  1. **SSI infrastructure**: It manages operating systems to perform unified access to all system resources.
  2. **System availability infrastructure:** It enables cluster services such as check pointing, automatic failover, recovery from failure, and fault-tolerant support among all nodes of the cluster.

5.1 Single System Image Levels/Layers

- A SSI is defined as the illusion, created by hardware or software, that presents a collection of resources as one, more powerful resource.
- SSI and system availability services can be offered by one or more of the following levels/layers:

- **Hardware** (such as Digital memory channel, hardware DSM and SMP techniques)
  - Digital's memory channel is dedicated clusters interconnect.
  - It provides virtual shared memory among nodes by means of intermodal address space mapping.

- **Operating system kernel** (Under ware or Gluing layers such as Solaris MC or GLUnix)
  - OS must support gang-scheduling and process migration
  - Must provide good operating capability for both sequential as well as parallel applications.
  - A full cluster vide SSI allows all resources and kernel resources to be visible and accessible from all other nodes with in the system.

- **Applications and subsystems – Middleware**
  - Applications such as system management tools and electronic forms
  - Runtime systems such as software DSM and parallel file system
  - Resource management and scheduling software such as LSF and CODINE.

5.2 Benefits of SSI

- Simple view of all system resources and activities, from any node of cluster.
- End user do not worry about to know where application will run.
- Frees operator to know where a resource is located.
- Let user work with familiar interface and commands
- Allows the administrator to manage the entire cluster as a single entity.
- Reduces the risk of operator errors, with improved reliability and higher availability.
- Allows centralized/decentralized system management and control to avaoid the need of skilled administrator.

- Presents multiple, cooperating components of an application to the administrator as a single applications.
- Greatly simplifies system management.
- Helps to track the locations of all resources.

5.3 Middleware Design Goals

- Design goals are mainly focused on complete transparency in resource management, scalable performance, and system availability in supporting user applications.

- **Complete Transparency**
    - The SSI layer must allow the user to use a cluster without the knowledge of underlying system architecture.
    - The operating environment appears familiar and is convenient to use.
    - The cluster is provided with the view of a globalized file system.
    - User can login at any node and the system administrator can install/load software at anyone's node and have be visible across the entire cluster.
    - Note that in distributed systems, one needs to install the same software for each node.
    - So it is states the clear advantage of transparency and simplicity of cluster compare to distributed systems.

- **Scalable Performance**
    - Scalability should happen without the need for new protocols and APIs.
    - To extract maximum performance, the SSI service must support load balancing and parallelism by distributing workload evenly among nodes.
    - For more scalability proper work distribution must be made through work load manager.

- **Enhanced Availability**
    - The middleware services must be highly available at all times.
    - At any point of time failure should be recoverable without affecting a user's application.
    - This can be achieved by employing check pointing and fault tolerance technologies to enable rollback recovery.
    - When any of the node fails, that portion of file system could be migrated to another node transparently.

5.4 Key Services of SSI

- **SSI Support Services:**
    - **Single point of Entry:** The user can connect to the cluster as a single system instead of individual nodes.
    - **Single file hierarchy:** There is hierarchy of files and all the directories come under same root director for every user. The hierarchy of files appears same to every user
    - **Single point of management:** Entire cluster could be maintained and controlled from a single window (like Task Manager in Windows).

- **Single virtual networking:** Any node can be accessed from any other node throughout the cluster, whether there is a physical link between the nodes or not.
- **Single memory space:** It appears as there is a huge single, shared memory. Memories of various nodes are associated to create an illusion of a single memory.
- **Single job management system:** User can submit job from any node and job can be run in parallel, batch or interactive modes.
- **Single user interface:** There should be a single cluster wide GUI interface for the user. The look and feel of the interface should be uniform on every node.

- **Availability Support Services:**
  - **Single I/O space:** Any node can perform I/O operations on local or remote peripheral devices.
  - **Single process space:** Every process is unique throughout the cluster and has a unique cluster wide process ID. Process can create child process on same or different nodes. Cluster should support global process management.
  - **Check pointing and process migration:** In check pointing, intermediate process results are saved so that process can resume from last checkpoint in case of system failure.
  - Process migration is the moving and running of process on some other node when there is a system crash or when the load on a particular node increases.

6. Resource Management and Scheduling (RMS)

- It is the act of distributing applications among computers to maximize their throughput.
- The software that performs RMS consists of two components.
  - **Resource manager:** It is concerned with problems, such as locating and allocating computational resources, as well as process creation and migration.
  - **Resource scheduler:** It is concerned with tasks such as queuing applications, as well as resource location and assignment.
- RMS is necessary in many aspects such as,
  - Load balancing
  - Utilizing spare CPU cycles
  - Providing fault tolerant systems
  - Managed access to powerful systems
- The basic RMS architecture is client-server system.
- In this each computer sharing computational resources runs a server daemon.
- These daemons maintain up-to-date tables, which store information about the RMS environment in which it resides.
- User interacts with RMS via a client program, it may be a web browser or customized X-windows interface.
- Application can be run either in interactive or batch mode.
- RMS environments also provide middleware services to users that should enable heterogeneous environment of workstations.

6.1 Services provided by RMS environment

- **Process migration**
    - It is used when a process can be suspended, moved and restarted on another computer within the RMS environment.
    - It generally occurs because of two reasons,
        1. If computational resource has become too heavily loaded and there are other free resources, which can be utilized.
        2. In conjunction with the process of minimizing the impact of users mentioned below.

- **Check pointing**
    - It is like snapshot of an executing program's state is saved.
    - That can be used to restart the program from the same point at a later time if necessary.

- **Scavenging Idle Cycles**
    - It is observed that between 70% and 90% most of the time workstations are idle.
    - RMS system can be set up to utilize idle CPU cycles.

- **Fault tolerance**
    - Fault tolerant support can mean that a failed job can be restarted or rerun.
    - Thus, it guaranteeing that the job will be completed.

- **Minimization of Impact on Users**
    - It can be done by either reducing a job's local scheduling priority or suspending the job.
    - Suspended jobs can be restarted later or migrated to other resources in the systems.

- **Load balancing**
    - Job distribution will allow for the efficient and effective usage of all the resources.
    - Process migration can also be part of the load balancing strategy.
    - It may be beneficial to move processes from overloaded system to lightly loaded ones.

- **Multiple applications queues**
    - Job queues can be set up to help and manage the resources at a particular organization.
    - Each queue can be configured with certain attributes.
    - For example, certain users have priority of short jobs run before long jobs.
    - Job queues can also be set up to manage the usage of specialized resources.

7. Cluster Applications

- GCAs (Grand challenge applications) are fundamental problems in science and engineering with broad economic and scientific impact.
- GCAs includes,
    - Massive crystallographic and microtomographic structural problems
    - Protein dynamics
    - Bio-catalysis

- Relativistic quantum chemistry of actinides
- Virtual materials design and processing
- Global climate modeling
- Discrete even simulation

8. Representative Cluster Systems

8.1 The Beowulf Project

- Investigate the potential of PC clusters for performing computational tasks
- Refer to a Pile-of-PCs (PoPC) to describe a loose ensemble or cluster of PCs
- Emphasize the use of mass-market commodity components, dedicated processors, and the use of a private communication network
- Achieve the best overall system cost/performance ratio for the cluster

- **System Software**
  - The collection of software tools
  - Resource management & support distributed applications

- **Communication**
  - Through TCP/IP over Ethernet internal to cluster
  - Employ multiple Ethernet networks in parallel to satisfy the internal data transfer bandwidth required

- All these can be achieved by ‘channel binding’ techniques
- Extend the Linux kernel to allow a loose ensemble of nodes to participate in a number of global namespaces.

- Beowulf implements **Two Global Process ID (GPID) schemes**
  1. Independent of external libraries
  2. GPID-PVM compatible with PVM Task ID format & uses PVM as its signal transport
- Beowulf project is developing a mechanism that allows unmodified versions of standard UNIX utilities to work across a cluster.

8.2 Berkley NOW

- Demonstrate building of a large-scale parallel computer system using mass produced commercial workstations & the latest commodity switch-based network components.
- **Inter-process communication**
  - Active Messages (AM)
  - basic communication primitives in Berkeley NOW
  - A simplified remote procedure call that can be implemented efficiently on a wide range of hardware

- **Global Layer Unix (GLUnix)**
  - An OS layer designed to provide transparent remote execution, support for interactive parallel & sequential jobs, load balancing, & backward compatibility for existing application binaries
  - Aim to provide a cluster-wide namespace and uses Network PIDs (NPIDs), and Virtual Node Numbers (VNNs)

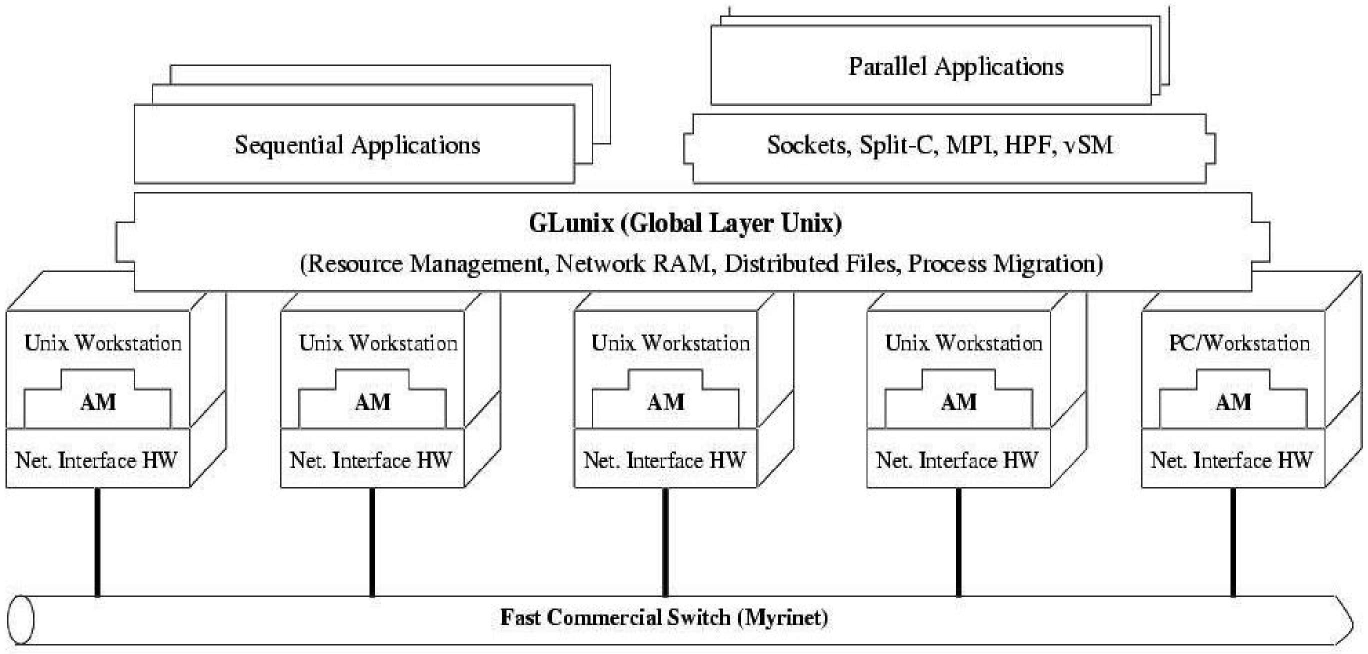


Figure 1.2 Berkley NOW architecture

- **Network RAM**
  - Allow to utilize free resources on idle machines as a paging device for busy machines
  - Serverless
  - any machine can be a server when it is idle, or a client when it needs more memory than physically available

- **xFS: Server-less Network File System**
  - A server-less, distributed file system, which attempt to have low latency, high bandwidth access to file system data by distributing the functionality of the server among the clients
  - The function of locating data in xFS is distributed by having each client responsible for servicing requests on a subset of the files
  - File data is striped across multiple clients to provide high bandwidth

8.3 High performance virtual machine (HPVM)

- Goal of HPVM is to deliver supercomputer performance on a low cost COTS (commodity of the shelf) system.
- It also aims to hide the complexities of a distributed system behind a clean interface.
- It provides software that enables high performance computing on clusters of PCs and workstations.
- Architecture components allow HPVM clusters to be competitive with dedicated MPP systems.

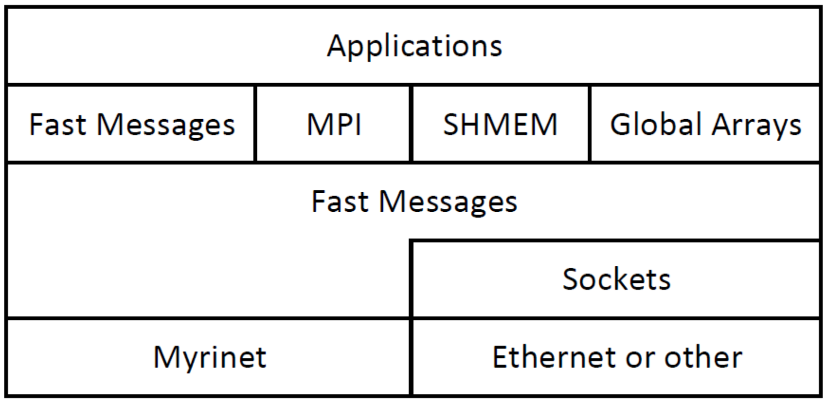


Figure 1.3 HPVM architecture

- The HPVM project aims to address the following challenges:
    - Delivering high performance communication to standard, high-level APIs.
    - Coordinating scheduling and resource management.
    - Managing heterogeneity.
- Critical part of HPVM is high-bandwidth and low-latency communication protocol known as Fast Messages(FM), which is based on Berkley AM.
- FM contains functions for sending long and short messages and for extracting messages from the network.
- FM guarantees reliable and ordered packet delivery as well as control over the scheduling of communication work.

8.4 Cluster of SMPs (CLUMPS)

- It is observed that clusters of multiprocessors (CLUMPS) promise to be the supercomputers of the future.
- In CLUMPS, multiple SMP with several network interfaces can be connected using high performance networks.
- This has two benefits:
    1. It is possible to benefit from the high performance
    2. Easy to use and program SMP systems with a small number of CPUs.
- Cluster can be set up with moderate effort.
- For example, a 32-CPU cluster can be constructed by using either commonly available eight 4-CPU SMPs or four 8-CPU SMPs instead of 32 single CPU machines.

- That results in easier administration and better support for data locality inside a node.
- A sub-broadcast over Cluster of SMPs consists of three levels,
  1. In each SMP node,
  2. Within each of switches called intra-switch broadcast, and
  3. Among switches called inter-switch broadcast.

Challenge:

- SMP clusters significantly increase the complexity of user application development when using the low-level application programming interfaces MPI and OpenMP.
- This forces users to deal with both distributed-memory and shared-memory parallelization details.

Solution:

- For above problem, extensions of HPF (High Performance Fortran) have been implemented for SMP clusters which enable the compiler to adopt a hybrid parallelization strategy.
- It efficiently combines distributed-memory with shared-memory parallelism.
- By means of a small set of new language features, the hierarchical structure of SMP clusters may be specified.
- This information is utilized by the compiler to derive inter-node data mappings for controlling distributed-memory parallelization across the nodes of a cluster.
- It also manages intra-node data mappings for extracting shared-memory parallelism within nodes.
- Additional mechanisms are also derived for specifying intra-node data mappings explicitly, for controlling specific SM parallelization issues, and for integrating OpenMP routines in HPF.
- The proposed features are being realized within the ADAPTOR and VFC compiler.

Project	Platform	Communication	OS	Other
Beowulf	PCs	Multiple Ethernet with TCP/IP	Linux and Grendel	MPI/PVM, Sockets and HPF
Berkley NOW	Solaris-based PCs and Workstations	Myrinet and Acrive Messages	Solaris + GLUnix + xFS	AM, PVM, MPI, HPF, Slpit-C
HPVM	PCs	Myrinet with Fast Messages	NT or Linux connection and global resource manager + LSF	Java-fontend, FM, Sockets, Global Arrays, SHMEM and MPI

Table 1.2 Cluster systems comparison matrix

1. Setting up the cluster

- Following are the **steps for establishing a cluster:**
  1. Starting from scratch
  2. Directory services inside the cluster
  3. Distributed computing environment (DCE) Integration
  4. Global clock synchronization
  5. Heterogeneous Clusters

1.1 Starting from scratch

Interconnection network:

- The first step is to develop a network topology and technology suitable for the cluster's communication needs.
- There are a few options available such as fast Ethernet, Myrinet, SCI (Scalable coherent interface), ATM (Asynchronous transmission mode) and so on.
- For example, fast Ethernet, where you can choose between direct links, hubs, switches and endless mixes of them.
- Using dynamic routing protocols inside the cluster to set the routes automatically introduces more traffic and complexity.
- It should be noted that lack of switches in a setup is big gain as there is no added latency.

Front-end setup:

- Most cluster include front end as some distinguished node where human users log in from the rest of the network.
- While designing front end, one or few nodes are used to design specifically for maintaining NFS (Network file system) for rest of the nodes.
- The additional advantage of front end is user can compile and test their software in exactly the same environment as the computing nodes.
- Advanced IP routing capabilities are also major goal of setting the front-end.
- For this various advance static and dynamic routing protocols are widely used.
- Security improvement and load balancing between nodes are also major features of setting the front-end of cluster.

Node Setup:

- To setup cluster nodes, get them to network boot and do automated remote installation.
- As cluster is a Homogeneous, all of the cluster nodes will have almost the same configuration.
- The fastest way for setting up all nodes is usually to install a single node and then clone its hard disk by following methods,
  - Copying its partition tables
  - Formatting and then copying files
  - Directly low-level copying from device to device

1.2 Directory services inside the cluster: NIS and NIS+

- Traditional approaches for directory access in clusters are NIS and NIS+.

NIS (Network Information System):

- It is a network naming and administration system for smaller networks that was developed by Sun Microsystems.
- Using NIS, each host client or server computer in the system has knowledge about the entire system.
- It is little utility in a cluster apart from keeping a common user database.
- It has no way of dynamically updating network routing information or any configuration changes to user-defined applications.

NIS+ (Network Information System plus):

- NIS+ is a later version that provides additional security and other facilities.
- NIS+ is basically designed for Solaris systems only.
- So it is difficult to use it unless all of the machines are running Solaris.
- Most clusters use either plain NIS for user authentication or solution of copying the password file to each node.
- For additional advantages with NIS and NIS+, LDAP (Lightweight directory access protocol) has been implemented and tested successfully with integration of directory services.

1.3 Distributed computing environment (DCE) Integration

- It provides following services
    - Highly scalable directory services
    - Security services
    - Distributed file system (DFS)
    - Clock synchronization (DTS)
    - Thread services
    - Remote process communication (RPC)
- The negative side of DCE is that it has never really become as main-stream as planned.
- It is not available on many platforms and some of its services have already been surpassed by further developments.

Advantages of DFS:

- The DFS features can be more useful with large campus-wide networks of work-stations.
- It supports replicated servers for read-only data, which can be deployed following the network topology to improve performance, while also getting the security benefits.
- It also provides caching mechanism on the local file system.

1.4 Global clock synchronization

- Whenever any kind of serialization is desired, a global time is needed.
- In order to implement a global time service, one possible choice is DTS (Distributed Time

Service), as a part of DCE.

- Whether the DCE is used or not, it is probably better to use the NTP (Network time protocol).
- A direct GPS receiver or other reliable source is also possible.

1.5 Heterogeneous Clusters

- Heterogeneous cluster means, few or many of the cluster components have difference in their behavior.

Benefits of using Heterogeneous clusters:

- Heterogeneous clusters are designed in order to exploit the higher floating point performance of certain architectures.
- It is also cost efficient compare to other systems for research purpose.
- Using vector super computer as a node is much easier with heterogeneous environment.
- Network of workstations (NOW), means existing workstations and other equipment can be a part of whole cluster.
- It facilitates the use of idle hardware.
- Examples of such heterogeneous systems are Condor and Piranha clusters.

Challenges for Heterogeneous cluster set-up:

- In Heterogeneous layout automating administration work will obviously become more complex because,
    - File system layouts are converging but still far from coherent situation
    - Software packaging is also different
    - The POSIX attempt at standardizing is not much successful
    - So finally administration commands are also different.
- For medium to large clusters, a solution is to develop a per-architecture and per-OS set of wrappers with a common external view.

2. Security

2.1 Security policies

- User must know the real need for security, and the reasons behind the security, and the way of using them properly.
- Using many passwords at every login might improve security but is not reasonable.
- There is a certain tradeoff between usability and security.
- So, for making easy and effective policies mostly two methods can be used,
    1. Finding the weakest point in NOW(Network of workstations) and COW(Cluster of Workstations)
    2. A little help while setting up the front end

1. Finding the weakest point in NOWs and COWs:

- In partial heterogeneous systems, isolating services from each other is almost impossible.
- It is sometimes difficult to track some services that, how these are related with other.

Example:

- Allowing `rsh` access from the outside to each node just by matching usernames and hosts with each user's `.rhosts` file is a bad idea.
- As a security incident in a single node compromises the security of all the systems who share that user's home.
- But a completely unrelated service, such as mail, can be abused in a similar way.
- Just change that user's `.forward` file to do the mail delivery via a pipe to an executable or script and the same effect is reached.
- The user's home might be exported from an external NFS server and can be shared by other workstations or entire cluster.
- A single intrusion in any of those systems implies a security compromise for all of them, even if the server and the cluster were initially secure.

2. A little help while setting up the front end:

- In a homogeneous cluster where all the nodes are equally configured and keep a consistent image, they should be equally secured.
- If it is true, connecting all the nodes directly to the external network should mean no security problem at all.
- There are two main problems with this approach,
  1. The human factor, we tend to make temporary changes and forget to restore them.
  2. System may have information leak.
- The operating system and its version can often be easily guessed just with IP access, even with just harmless services running.
- These reasons are enough to justify a front-end from a security viewpoint in most cases.
- Clusters are often used from external workstations in other networks, in this way a front-end can also serve as a simple firewall.

2.2 Security versus performance tradeoffs

- Most security measures have no impact on performance and proper planning can avoid that impact.
- Very often, they are a sign of bad designing.
- There are certain cases where security does have an impact on performance.
- This is usually the case with strong ciphers.
- Secure channels are becoming common and the overhead of cryptography can severely damage performance.

2.3 Meta Cluster or Clusters of Clusters

- **Meta cluster** can be defined as Cluster of Clusters.

Security in Meta cluster:

- Special care must be taken on the security implications while building clusters of clusters.
- The simplest approach for making these meta-clusters secure is building secure channels between the clusters, usually from front-end to front-end (Figure 2.1).
- Remote systems where the cluster-to-cluster network performance is far slower than inside each individual cluster.
- This is unlikely to become a performance bottleneck.
- It is just another security versus performance tradeoff.

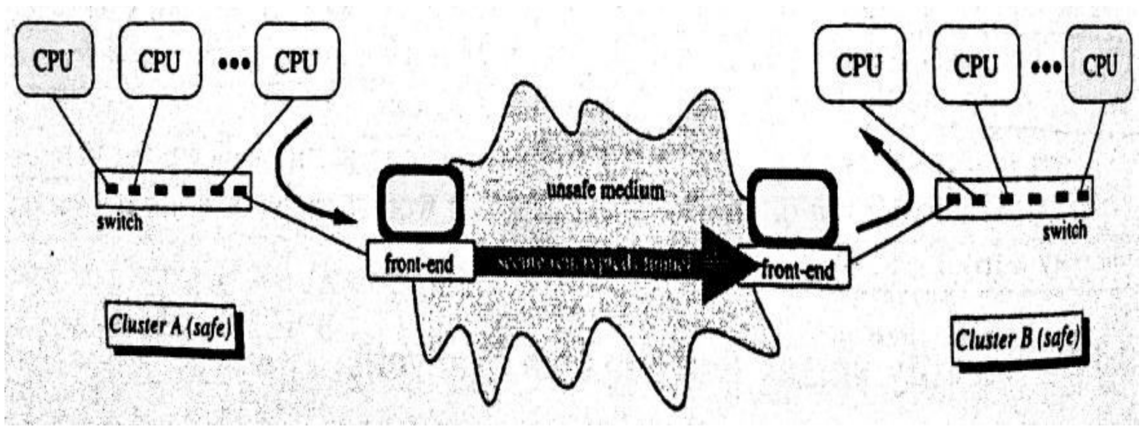


Figure 2.1: Inter-cluster communication using secure channel

- When security requirements are high a dedicated channel front-end or keeping the usual front-end free for minimizing the throughput penalty.
- Then also the latency will be higher due to an increased number of per-packet processing and cryptography algorithms used.
- Another additional performance tradeoff is stronger ciphers tend to mean slower ones.

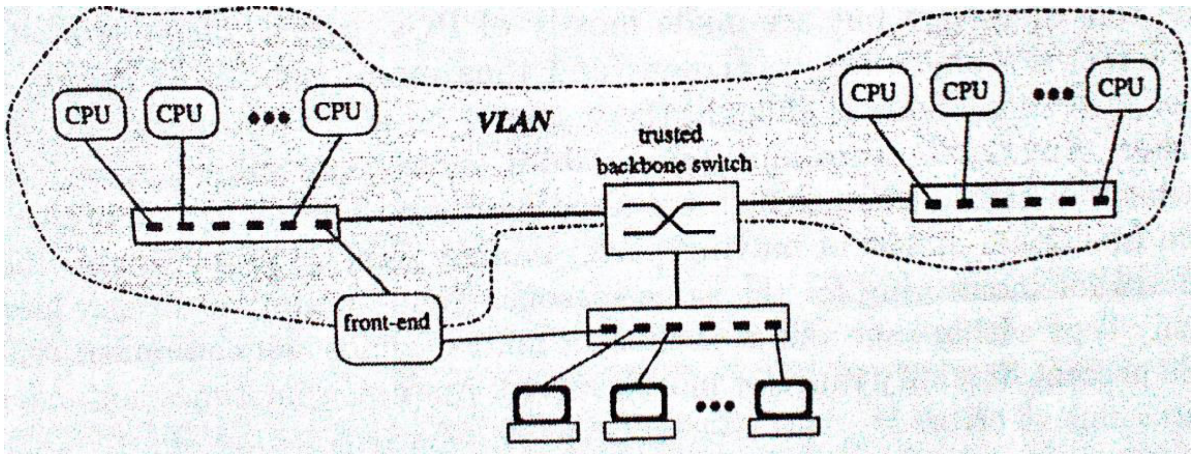


Figure 2.2: VLAN using a trusted backbone switch

- Intermediate backbone switches can be trusted and have the necessary software and resources.
- They can setup a VLAN joining the clusters, achieving greater bandwidth and lower latency than routing at the IP level via the front-ends (Figure 2.2).
- Routing to the rest of the network is still done through the front-ends.
- The decrement in latency may be significant because,
    - Data flow avoids two complete trips through the IP stack of the front-ends
    - Fewer resources are used in them.
- However, that adding resources to the routers is expensive than to the front-ends, and costly than adding complete dedicated nodes for this task.

3. System Monitoring

3.1 Unsuitability of general purpose monitoring tools

- It is vital to stay informed of any incidents that may cause unplanned downtime or intermittent problems.
- Most of the monitoring tools are designed for the management of middle or large networks.
- Their main purpose is network monitoring. It doesn't give the complete solution.
- In most cluster setups it is possible and worthy to install custom agents in the nodes.
- It is desirable to track their usage, load, network traffic statistics and whatever other metrics are considered locally.

3.2 Subjects of monitoring

Physical monitoring:

- Some environmental monitoring facilities are assumed for server systems.
- Most of the systems are workstations or individual PCs do not have this yet.
- Most clusters are physically contiguous and thus monitoring most environmental variables at a single node is enough.
- There should be at least a measuring point in cluster.
- Among all nodes, those who are subject to change with time, when not tightly conditioned, are best candidates for monitoring.
- For example, temperature and humidity, supply voltages and the functional status of moving parts.

Logical monitoring:

- The monitoring of logical services is aimed at finding current problems when they are already impacting the system.
- Logical services range from lower level like,
    - Raw network access and running processor to higher level.
    - RPC and NFS services running, correct routing etc.
- All monitoring tool provide some way of defining customized scripts for testing individual services.

- Some basic services are simple like ping responding, echo server working.

Performance meters:

- Unlike other monitors, performance itself is not an absolute term and much less well defined.
- Explicit instrumenting by code profiling is simple and effective, while effect of communication is not important.
- For switched Ethernet systems, it is possible to plug each endpoint to a small hub together with a card in the measuring system, then hub to switch.
- For load distribution active or passive probing is commonly used by load-balancing systems as a low weight way.

3.3 Self-diagnosis and automatic corrective procedures

- All monitors need to know the state of a system in order to take corrective measures.
- The next logical step is making the system take this decision itself.
- Cases where a simple symptom can be associated to a corrective action are rare in practice.
- So most actions end up being "page the administrator" which is far from optimal, both for the system uptime and for the administrator.
- To take some reasonable decisions system should know what sets of symptoms lead to suspect of what failures.
- Any monitor performing automatic corrections should be at least based on a rule-based system and rely on direct alert-action relations.

4. System Tuning

- Tuning a system can be seen as minimizing a cost function.
- Higher throughput for a job may not be worthwhile if it increases network or server load.
- On the basis of throughput, latency, I/O implications, caching and OS various strategies for cluster system tuning has been designed as follows.

Focusing on Throughput or Focusing on Latency

- Network latency tends to be critical for most applications.
- While on conventional systems the focus is on throughput, nodes in high performance clusters are usually best tuned for low latency.
- This is just a way for achieving the same goal, that is high global throughput.
- Each node can be considered as just a component of the whole cluster, and its tuning aimed at global – not local – performance.

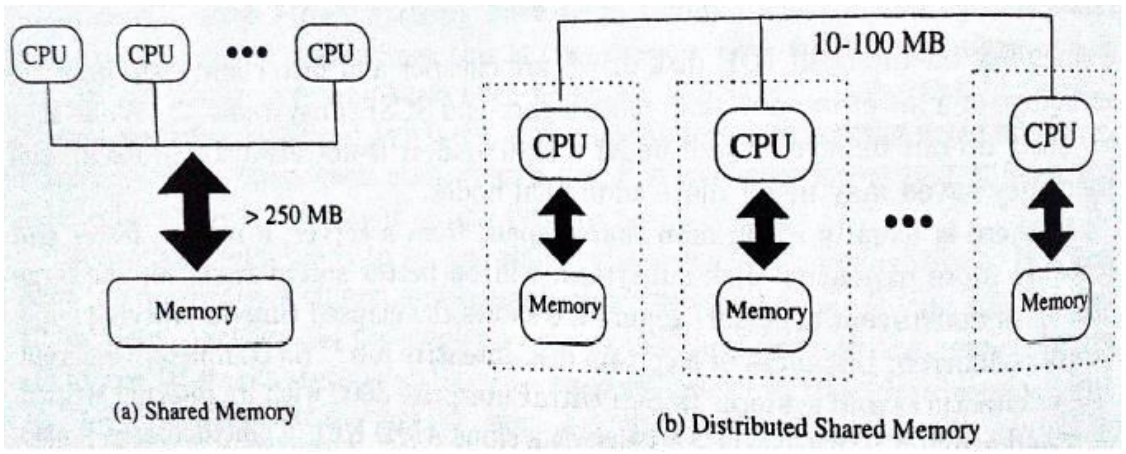
I/O Implications

- I/O subsystems as used in conventional servers are not always good choice for cluster nodes.

- Many conventional servers implement software RAID across their disks.
- Data should be distributed across nodes in a similar way in order to improve through put.
- Even at the expense of some write speed, performing distributed data replication or parity we can do same.
- Difference between raw disk and file system throughput becomes more evident as systems are scaled up.

Caching Strategies

- There is only one important difference between conventional multiprocessors and clusters that is the availability of shared memory.
- At present getting a data block from the network can provide both lower latency and higher throughput than from the local disk.
- The problem is how to provide enough data flow from the other end to saturate the network.
- The obvious solution is to aggregate flow issued from several nodes.
- Usual data caching strategies may often have to be inverted.
- While faster rates can be obtained from concurrent access to other nodes.
- This concurrent access has a price, wasting other nodes resources.
- In the extreme case of purely sequential task, we are introducing some data parallelism just with this distributed access, improving global throughput.



Fine-tuning the OS

- In practice, getting big improvements just by tuning the system is unrealistic most times, but certain changes do help.
- For fine tuning of the operating system following two methods are used widely.

Virtual memory(VM) subsystem tuning:

- Optimizations depend on the application.
- Some trivially parallel cryptography even fits in the caches, large jobs often benefit from some VM tuning.

- Highly tuned code will fit the available memory, so it will be wise to the system from paging until a very high watermark has been reached.

Networking:

- When the application is communication-limited, sometimes big gains are possible.
- The IP suite is almost always used directly, or as a layer under PVM or MPI.
- Wrong values can completely defeat the purpose of its algorithms and make performance far worse.
- For bulk data transfers, increa8sing the TCP and UDP receive buffers is almost a need in high speed networks.
- Large TCP window sizes and window scaling helps, although they are usually the default nowadays.

1. Environment

- Complex network systems consist of a collection of wide and local area networks.
- They differ in characteristics like speed, reliability and many others.
- All these networks are interconnected to form a single global complex network system.
- Nodes vary considerably in attributes, some of which are static and others are dynamic.
- Similarly, communication channels connecting nodes vary greatly by static attributes like bandwidth and dynamic ones like channel load.
- There are wired connections as well as wireless ones.
- Some of the latter may use satellite communication and may also be mobile.
- Before discussing resource sharing mechanism, we should aware about some factors of cluster network system.
- Cluster network system can be discussed through following factors.

1.1 Faults, Delays, and Mobility

- A distinguish feature of complex network systems is the inability to maintain consistent global state information at distributed points of control.
- So complex network system is a type of distributed system.
- This distributed system has various decision makers to achieve common goal.
- Network delays are the causes of these several multiple decision latencies.

1.2 Scalability Definition and Measurement

- Scalability is defined as the system's ability to increase speedup as the number of processors increase.
- Scale is recognized as a primary scalability factor.
- Speedup measures the possible benefits of a parallel performance over a sequential performance.
- Efficiency of any system can be given as follows:

$$\textbf{Efficiency} = \frac{\textbf{Speedup}}{\textbf{No of Processors}}$$

- A rigorous analysis of the scalability metric is required to identify its dominant factors and measure it in practice.
- A number of works propose different approaches to evaluate scalability.
- Some of these metrics measure the ability of the sytem to multiply resources to solve problems of proportionally increasing size while preserving efficiency.

1.3 Weak Consistency

- The environment described so far is characterized by a high degree of multiplicity(scale), variable fault rates(reliability), resource with reduced capacity(mobility), and variable interconnections resulting in different sorts of latencies.

- These characteristics make the environment complex to handle.
- On the other side of the coin lies another attribute of weak consistency which is permissible for resource access.
- Weak consistency allows inaccuracy as well as partiality.

1.4 Assumptions Summary

Assumption taken for these unit are as follow:

- Full logical interconnection is assumed so that a message sent by a node can reach any other node in the system
- Delays in the system are finite, and failed nodes are detected in finite time.
- Connection maintenance while mobile is assumed to be transparent to the application and is provided by the network layer responsible for routing decisions.
- Nodes have unique identifiers numbered sequentially.
- These identifiers are kept intact as nodes move and coross wirelesss connection boundaries.

1.5 Requirements for Resource Sharing Algorithm

- The resource sharing algorithm should meet the following requirements.
- Adaptability: The algorithm should respond quickly to changes in the system state and adapt its operations to new evolving conditions.
- Generality: The algorithm should be general to serve a wide range of applications and distributed environments.
- It should not assume prior knowledge of the system characteristics.
- Minimum Overhead: The algorithm should respond to requests quickly, incurring minimal overhead on the system.
- For example processing, memory usage, network I/O.
- Stability: Stability is the ability of the algorithm to prevent poor resource allocation.
- Scalability: An algorithm should be minimally dependent of the system's physical characteristics.
- Fault-Tolerance: A failure of one or a few nodes should have a minimal impact on the entire system.
- Heterogeneity: A complex network environment has to support heterogeneous hardware and software environments.
- To limit the scope of this chapter, heterogeneity is only potentially covered.

2. Resource Sharing

- Distributed systems require some mechanism for cooperation among the processors, attempting to assure that no processor is idle while there are tasks waiting for service.
- A solution to the resource access problem attempts to ensure that there are no such resources idling while requests are queued at other nodes.
- Load sharing algorithms provide an example of the cooperation mechanism required

when using the mutual interest relation.

- A location policy determines the approach used for locating a remote resource.
- Such load sharing algorithms has following components,
    - Information propagation
    - Request acceptance
    - Process transfer policies

2.1 FLS (Flexible Load Sharing) algorithm with its analysis

- In cluster, various load sharing algorithms are available for sharing or distributing load between nodes.
- FLS (Flexible Load Sharing) algorithm is one of the most famous algorithms for load sharing.
- FLS algorithm supports scalable and adaptive initial allocation of processes to processors.
- It is also used to avoid premature deletion information from the cache.

FLS working policy:

- A location policy determines the approach used for locating a remote resource.
- As basic study the performance of location policies with different complexity levels were compared.
- Three location policies were studied: random policy, threshold policy and shortest policy.
- Among those three policies FLS location policy is similar to threshold policy.
- Threshold policy probes limited number of nodes.
- It terminates the probing as soon as it finds a node with a queue length shorter than the threshold (specific value/point).
- Threshold results in a substantial further performance improvement.
- In contrast with threshold, FLS bases its decisions on local information which is possibly replicated at multiple nodes.

FLS workflow:

- FLS divides a system into small subsets to achieve scalability.
- Each of these subsets forms a local cache held at a node.
- Cache members are nodes of mutual interest which are first discovered by random selection.
- Biased random selection is used from then on in order to retain entries of mutual interest and select others to replace discarded entries.
- The cache actually defines a subset of nodes, within which the node sinks a partner.
- That way the search scope is constrained, no matter how large the system is as a whole.
- The algorithm supports both, mutual inclusion and exclusion.
- In order to minimize state transfer activity, the choice is biased and nodes sharing mutual interests are retained.
- In this manner premature deletion is avoided.

Policy of finding out the mutually interested partner:

- The necessary information for making partners, sharing a mutual interest, is maintained and updated locally on a regular basis.
- Cache entries of mutual interest are retained as long as they are of interest. Premature deletion is thus avoided.
- This policy shortens the time period that passes between issuing the request for matching and actually finding a partner having a mutual interest.

3. Analysis of Resource Sharing Algorithms

3.1 Characteristics of good Resource sharing algorithms

- While researching many of the resource sharing algorithms, it has been noted that some characteristics are sacrificed in favor of others.
- For good resource sharing algorithm basically four types of evaluations should be considered,
    1. Qualitative evaluation
    2. Information dissemination
    3. Decision making
    4. Quantitative evaluation

Following are the **basic requirements and characteristics of good resource sharing algorithms.**

Qualitative evaluation:

- Resource sharing algorithm should avoid single point of failure, means fault tolerance.
- For low overhead, they should employ simple techniques for fault-tolerance such as periodic information exchange.
- It should be symmetrically distributed resource sharing algorithm for fault tolerance and low overhead.

Information dissemination:

- It must be capable for holding local and non-local dynamic state information held in a state vector.
- Because information holding, exchange and update plays an important role in maintaining the local view of the system state at a node.
- It should fulfill the user requirement, weather to hold state information regarding all nodes in the system or only a subset of the system
- It must also specify all the criteria for selecting the nodes to be included in the subsets.

Decision making:

- It must able to specify that when the system is heavily congested, much higher delays than the average may be expected, which can severely degrade performance.
- While designing such algorithms numerous design choices and open issues left for the

designer to resolve.

- So, the algorithm has to examine those issues with respect to the required properties and applications environment.

Quantitative evaluation:

- For objective evaluation of different approaches a quantitative analysis techniques is needed.
- For characterizing the structure of distributed decision-making policies, the terms performance and efficiency are used.
- A resource sharing algorithm must be checked by the means of performance, efficiency and stability.

3.2 Methods for checking performance, efficiency and stability of any Resource Sharing Algorithm.

Performance:

- Performance is an absolute measure which is described in terms of response-time(RT), utilization or any other objective function specified.
- A performance metric relevant to our study defines the distance of an improved algorithm from the original as a percentage.

$$\text{distance(original, extended)} = \frac{\mathrm{RT(original)} - \mathrm{RT(extended)}}{\mathrm{RT(original)}}$$

- Here, Response-time(RT) is evaluated by following equation,

$$\mathrm{RT} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{System\ Size}} \mathrm{RT(HOST(i))}}{\mathrm{SystemSize}}$$

- A positive distance indicates the improvement in performance and negative indicates degradation.
- While comparing with other resource sharing algorithm, the same models must be used for hardware configuration and network as well as CPU delays.
- It must be noted that algorithms must operate under the same load during comparison.

Efficiency:

- It is a relative term concerned with the cost or penalty paid for the level of performance attained.
- After checking performance measure, efficiency measure can be used to check which of the specific requirements are satisfied.
- Efficiency measures quantify the overhead or cost associated with system of a specific level of performance in terms of the following:

- Memory requirements for algorithm constructs
- State dissemination cost in terms of the rate of resource sharing state messages exchanged per node
- In terms of CPU time run-time cost is measured as the fraction of time spent running the resource across software component
- Percent of remote resources accesses out of all resource access requests

Scalability:

- Stability is a precondition for scalability.
- Stability can be defined as a system property measured by resource sharing hit-ratio.
- It provides guidance as to the quality of the decisions made.
- A “hit” is defined as a lookup request, which could be answered with the current contents of the cache.
- Hit measures the percent of successful decisions.
- Another term “miss” measures a lookup request which could not be answered with the current contents of the cache.
- In short, miss measures the percent of unsuccessful decisions.

4. Resource Sharing Enhanced Locality

- Resource sharing decisions are based on both local and non-local dynamic state information.
- All type of information can be hold in the system or subsets of the system.
- Algorithms holding state information regarding all other nodes in the system are dependent on system size and therefore are not scalable.
- Such type of situations has a significant effect on performance in a complex network environment.

For enabling the analysis of resource sharing algorithm in complex environment following **characteristics** need to be focused.

- No message loss.
- Non-negligible (>0) but constrained latencies for accessing any node from any other node.
- Availability of unlimited resource capacity. For example, the number of nodes in a cache is not limited.
- The selection of new resource providers to be included in the cache is not a costly operation and need not be constrained.
- Maintaining a local cache at each node. Usually it is denoted by n, which is the number of nodes trying to include in cache.
- A cache contains the identification of other nodes with mutual interest currently known to the node together with their current state.

What is Grid Computing?

- **Grid computing** is the collection of various resources from multiple locations to reach a common goal.
- The **grid** can be thought of as a **distributed system** with non-interactive workloads that involve a large number of files and **heterogeneous nodes.**

1. Why there is need of Grid Computing?

- Share data between thousands of scientists with multiple interests consumes following **challenges over simple network**
    - Link major and minor computer centres
    - Ensure all data accessible anywhere, anytime
    - Grow rapidly, yet remain reliable for more than a decade
    - Cope with different management policies of different centres
    - Ensure data security
    - Be up and running routinely
    - Need to check up health of facility on 24X7 bases.
    - A huge man power is at work invisibly.

- To solve all such critical demands grid computing should be used, because grid its following **advantages**
    - Can solve larger, more complex problems in a shorter time
    - Easier to collaborate with other organizations
    - Make better use of existing hardware
    - No need to buy large six figure SMP servers for applications that can be split up and farmed out to smaller commodity type servers.
    - Much more efficient use of idle resources.
    - Grid environments are much more modular and don't have single points of failure.
    - If you need more resource at run time, grid model can scale them very well
    - Upgrading can be done on the fly without scheduling downtime.
    - Jobs can be executed in parallel speeding performance.

2. Building Blocks of the Grid

- Following are the main building blocks of Grid.
    1. Networks
    2. Computational nodes
    3. Combination of networks and nodes
    4. Common infrastructure: standards

Networks

- It connects geographically distributed resources to work together for achieving a common goal.

- “pipe” is the term, used to state connection bonding between resources.
- If networks provide “big/wider pipes”, applications can use distributed resources in a more integrated and data-intensive fashion.
- If networks provide “small/thinner pipes”, applications have minimal communication and data transfer between components and is able to tolerate high latency.
- High capacity networking increases the capability of the grid to support both parallel and distributed applications.
- Wired grid networks are enhanced by the use of wireless connectivity.
- Wireless resource drives integration of more and more devices into the grid.
- The desktop resource connectivity includes the pervasive PDA (Personal Digital Assistant).
- PDA is a term for any small mobile hand-held device that provides computing and information storage and retrieval capabilities for personal or business use.
- PDA will further promote the grid as a platform for e-Science, e-Commerce and e-Education.

Computational Nodes

- At the research level grid, nodes are themselves high-performance parallel machines or clusters.
- Such high-performance grid nodes provide major resources for simulation, analysis, data mining and other compute-intensive activities.
- The performance of the most high-performance nodes on the grid tracked by the Top500.org website.
- Study of such nodes indicates that we can expect a peak signal machine performance of 1 petaflops/sec (10^{15} operations per second) in 2nd generation of grid.

Combination of networks and nodes

- Some projects are developing high-end, high-performance grids with fast networks and powerful grid nodes.
- Such nodes provide a foundation of experience for the grid of the future.
- Much of the critical grid software is built as part of infrastructure activities.
- The grid application development system gives combination of grid program development and execution environment.

Common infrastructure: standards

- Common infrastructure and standards are needed to promote interoperability and reusability.
- They are also required to base their systems on a growing body of robust community software.
- Advanced technologies will provide greater and greater potential capability and capacity and will need to be integrated within grid technologies.
- Grid utilizes a common infrastructure to provide a virtual representation to software developers and users, while allowing the incorporation of new technologies.

- IETF (Internet engineering task force) and W3C have defined key standards such as TCP/IP, HTTP, SOAP, XML and now WSDL (Web service definition language) that underlines OGSA (Open grid service architecture).
- Following are some other important standard on which grid is built.
    - **Linux OS:** Most of the grid uses Linux as standard node operating system.
    - **OASIS (Organization for the Advancement of Structured Information Standards)** is standardizing web services for remote portals.
    - **The apache project** supplies key infrastructure such as servers and tools to support such areas as WSDL – Java interfaces and portals.
    - **Modern languages** like java with good run-time and development environments.

3. Grid Applications

- The grid is serving as a best technology for a broad set of applications.
- Following are the main applications of grid computing.
    - Life science applications
    - Engineering-oriented applications
    - Data-oriented applications
    - Physical science applications
    - Trends in research: for example e-Science project
    - Commercial applications
    - Adaptive applications
    - Real-time and on-demand applications
    - Coordinated applications
    - Poly applications

Life science applications

- Life science applications include following technologies.
    - Biology
    - Bioinformatics
    - Genomics
    - Computational neuroscience etc.
- Above technologies uses grid technology as a way to access, collect and mining data and accomplish large-scale simulation and analysis.
- One of the most popular examples of life science grid project is BIRN (Biomedical Informatics Research Network).

Engineering-oriented applications

- The grid provides cost effective platform for making resource-intensive engineering applications.
- NASA IPG (Information power grid) and UkGrid projects are very famous examples of engineering-oriented applications of grid.

Data-oriented applications
- The grid is used to collect, store and analyze the data and information from various sources.
- Grid also synthesizes the gathered data.
- Example of data-oriented application is DAME (Distributed aircraft maintenance environment) developed in UK.

Physical science applications
- Physical science applications consider highly innovative particle physics-dominated projects.
- The astronomy community has targeted the grid for collecting, sharing and mining critical data about universe.
- Examples of such projects are NVO (National virtual observatory) project of US, and AstroGrid project of UK.

Trends in research: e-Science in a collaborator
- e-Science captures the new approach to science involving distributed global collaborations.
- It can be done by the internet and using very large data collections, terascale computing resources and high-performance visualizations.
- e-Science is about global collaboration in key areas of science, and grid.

Commercial applications
- In commercial world grid is widely used for inventory control, enterprise computing, and games and so on.
- Enterprise or commercial areas where grid approach can be applied includes following features
    - End-to-end automation
    - End-to-end security
    - Virtual server hosting
    - Disaster recovery
    - Heterogeneous workload management
    - End-to-end systems management
    - Scalable clustering
    - Accessing the infrastructure
    - Utility computing
    - Accessing new capability more quickly
    - Better performance
    - Reducing up-front investment
    - Gaining expertise not available internally
    - Web based access(Web portal) for control of enterprise function

Adaptive applications

- It is useful when we can find resources satisfying criteria X.

Real-time and on-demand applications

- It is useful when we find on demand operations or on the spot actions.

Coordinated applications

- It is useful for dynamic programming, like branch and bound conditions.

Poly applications

- It is useful when we need to decide resources for different system components.

4. Future of Grid Computing

- In many ways, the research, development and deployment of large-scale Grids are just beginning.
- Both the major application drivers and Grid technology itself will greatly change over the next decade.
- The future will expand existing technologies and integrate new technologies.
- In the future, more resources will be linked by more and better networks.
- At the end of the decade, sensors, PDAs, health monitors and other devices will be linked to the Grid.
- Petabyte data resources and petaflop computational resources will join low-level sensors and sensornets to constitute Grids of unprecedented heterogeneity and performance variation.
- Over the next decade, Grid software will become more sophisticated, supporting unprecedented diversity, scale, globalization and adaptation.
- Applications will use Grids in sophisticated ways, adapting to dynamic configurations of resources and performance variations to achieve goals of Autonomic computing.
- Accomplishing these technical and disciplinary achievements will require an immense research, development and deployment effort from the community.
- Technical requirements will need to be supported by the human drivers for Grid research, development and education.
- Resources must be made available to design, build and maintain Grids that are of high capacity.

5. The Grid: Next Generation Internet

- Grid computing is increasingly being viewed as the next phase of internet.
- It enables organizations to share computing and information resources across department and organizational boundaries in a secure, highly efficient manner.
- The next generation Grid will virtualize the notion of distribution in computation, storage and communication over unlimited resources.
- The NGG (Next generation grid) vision, which has emerged as the vision for Grid

Technologies, consists of three complementary perspectives

1. **The end-user perspective** which implies the simplicity of access to and use of Grid technologies.
2. **The architectural perspective** where the Grid is seen as a large evolutionary system made of billions of interconnected nodes of any type and the software perspective of a programmable and customizable Grid.
3. **Emerging pervasive** wide area Grid computing environments are enabling a new generation of applications that are based on seamless aggregation and integration of resources, services and information.

- Grid also provides many features as follows, which makes the system better than traditional.
  - Heterogeneity
  - Scalability
  - Dynamicity or Adaptability
  - Transparency
  - Openness for multiple components
  - Fault Tolerance
  - Load balancing

5.1 Restricted or secure access to Grid components/resources

- Grid can provide restricted or secure access to their resources using following features

  1. **Single sign on** using proxy certificate
  2. **Delegation of privileges:** The proxy certificate along with private key is called proxy credentials.
  3. The delegation of privileges in this way called credential delegation.
  4. **Inter domain security support**, means proxy or an agent running on the local system on behalf of the remote client to provide access to the local resource.
  5. **Secure communication** through TLS (Transport layer security)
  6. **Authentication and Authorization** through CA (Certification Authorities)
  7. **Uniform credentials**: GSI (Grid security infrastructure uses X.509 certificate for representing the credentials of the entities in the grid)

6. Grid Middleware

Middleware introduction

- "Middleware" is the software that organizes and integrates the resources in a grid.
- Middleware is made up of many software programs, containing thousands of lines of programming code.
- This code automates all the "machine to machine" (M2M) interactions that create a single, seamless computational grid.
- Middleware automatically negotiate deals in which resources are exchanged, passing from a grid resource provider to a grid user.

Middleware as Agents and Brokers

- Some middleware programs act as "agents" and others as "brokers".
- Agent programs present "metadata" (data about data) that describes users, data and resources.
- Broker programs undertake the machine to machine negotiations required for user authentication and authorization,
- Then broker strike the "deals" for access to, and payment for, specific data and resources.
- Once a deal is set, the broker schedules the necessary computational activities and oversees the data transfers.
- At the same time, special home agents optimize network routings and monitor quality of service.

Driving inside middleware

- There are many other layers within the middleware layer.
- For example, middleware includes a layer of "resource and connectivity protocols", and a higher layer of "collective services".
- Resource and connectivity protocols handle all grid-specific network transactions between different computers and grid resources.
- This is done with communication protocols, which allow the resources to
    - Communicate with each other.
    - Enabling exchange of data.
    - Authentication protocols, which provide secure mechanisms for verifying the identity of both users and resources.

Collective services

- The collective services obtain information about the structure and state of the resources on grid resources.
- Collective service includes
    - Updating directories of available resources
    - Brokering resources
    - Monitoring and diagnosing problems
    - Replicating data for generating multiple copies
    - Providing membership/policy services for tracking.

Condor Middleware

- Condor is a software package for executing batch jobs on a variety of UNIX platforms, which are idle.
- The major features of condor are as follow
    - Automatic resource location
    - Job allocation
    - Check pointing
    - Migration of processes

- All above features are implemented without modification to the underlying UNIX kernel.
- It is required for user to link source code with condor library.
- Condor monitors the activities on all the participating nodes.
- Condor maintains pool/log, which has dynamic entity.
- When any resource becomes idle that enters in the pool, and leaves when they get busy.

Condor-G (Condor Grid)

- The Condor-G job manager is a grid task broker that provides brokering service between resources.
- Condor-G is a client-side service and must be installed on the submitting systems.
- A Condor manager server is started by the user and then jobs are submitted to this user job manager.
- This manager deals with refreshing the proxy that the grid resource must have in order to run the user's jobs.
- The user must supply new proxies to the Condor manager.
- The manager must stay alive while the jobs are running on the remote grid resource in order to keep track of the jobs as they complete.
- Condor-G can recover from both server-side and client-side crashes.
- This job model is also called 'peer-to-peer' systems.
- PVM (Parallel virtual machine) is another distributed memory programming system that can be used in conjunction with Condor and Globus.
- PVM is used to provide grid functionality for running tightly coupled processes.

7. The Evolution of The Grid

7.1 First generation

- The first generation grid was started to link various supercomputing websites. This approach is called **meta-computing.**
- The **objective of meta-computing** was to provide computational resources to a range of high performance applications.
- There were **two major projects,** started to implement this concept
    1. FAFNER (Factoring via network enabled recursion)
    2. I-WAY (Information wide area year)

FAFNER project

- FAFNER was set up to factor RSA130 public key encryption standard using a new numerical technique called the NFS (Number field sieve) method using web servers.
- The consortium produced a web interface to NFS (Network file system).
- There was a web form for contributor to invoke server side CGI (Common gateway interface).
- Contributor could form one set of web pages, that access a wide range of support services for factorisation as follow
    - NFS software distribution

- Project documentation
- Anonymous user registration
- Dissemination of sieving tasks
- Collection of relations
- Relation archival services
- Real-time sieving status reports
- The FAFNER project won an award in TeraFlop challenge at supercomputing 1995(SC95) project at San Diego.
- It enhances the way of web based meta computing.

I-WAY project

- The information wide area year (I-WAY) project was started in early 1995 with the idea to integrate existing high bandwidth networks.
- The virtual environments, datasets, and computers used at 17 different US sites were connected by 10 networks of varying bandwidths and protocols,
- Different routing and switching technologies in this project.
- The network was based on Asynchronous transfer mode (ATM).
- It supported both Transmission control protocol/Internet protocol (TCP/IP) over ATM and direct ATM-oriented protocols.
- For standardization, key sites installed point-of-presence (I-POP) servers to act as gateways to I-WAY.

I-POP service

- The I-POP servers were UNIX workstations configured uniformly and possessing a standard software environment called I-Soft.
- I-Soft attempted to overcome issues concerning heterogeneity, scalability, performance, and security.
- Each site participating in I-WAY ran an I-POP server which provided uniform I-WAY authentication, resource reservation, process creation, and communication functions.
- Each I-POP server was accessible via the Internet.
- An ATM interface was there for monitoring and potential management of the site's ATM switch.
- The I-WAY project developed a resource scheduler known as the Computational resource broker (CRB) with user-to-CRB and CRB-to-local-scheduler protocols.
- The I-WAY project was application driven and defined several types of applications like supercomputing, Access to Remote Resources, Virtual Reality etc.
- I-WAY unified the resources at multiple supercomputing centres.

Limitations of I-WAY

- I-WAY lacked scalability.
- I-WAY was limited by the design of components that made up I-POP and I-Soft.
- I-WAY embodied a number of features that would today seem inappropriate.

- The installation of an I-POP platform made it easier to set up I-WAY services in a uniform manner.
- In addition, the I-POP platform and server created one, of many, single points of failure in the design of the I-WAY.
- However, despite of the aforementioned features I-WAY was highly innovative and successful.
- I-WAY provided the base for Globus and Legion.

7.2 Second generation

- 2nd Generation requires large scale of data and computations, Some basic requirements to achieve it are as follows:
    - Administrative hierarchy
    - Communication services
    - Information services
    - Naming services
    - Security
    - Resource management and scheduling
    - User and administrative GUIs

Issues for 2nd generation of grid

- Heterogeneity
- Scalability
- Adaptability

Some of the core technologies or 2nd generation

- Globus
- Legion
- Jini and RMI
- SRB
- Nimrod/G
- Grid Architecture for Computational Economy(GRACE)
- Grid portals
- Integrated system

Introduction of some 2nd generation core Grid technologies

Globus

- It provides software infrastructure for applications to handle distributed heterogeneous computing resources.
- It also helps to create single system image.
- The central element of the Globus is Globus Toolkit, which gives basic services and capabilities required to construct a computational grid.
- Global services are built upon the core-local services.

- For resource allocation and monitoring HTTP based GRAM (Globus toolkit resource manager) is used.
- For high speed transfer extended FTP protocol, GridFTP (Grid file transfer protocol) is used.
- LDAP (Lightweight directory access protocol) is built with TCP for querying state or remote database.
- LDAP is used in Globus for distributed access to state any information.

Legion

- Legion is an object based meta-system.
- Provides a software infrastructure to perform integration between different heterogeneous, geographically distributed, and high-performance machines.
- Difference between Globus and Legion is, Legion encapsulates it's all components as objects.
- So it has all advantages of object oriented approaches like abstraction, encapsulation, inheritance and polymorphism.
- Legion system had the following core object types.
    - Classes and metaclasses
    - Host objects
    - Vault objects
    - Implementation objects and caches
    - Binding agents
    - Context objects and context spaces
- It also has Jini architecture that can form distributed computing environment with plug and play facility.
- It uses RMI method to deal with remote resources.

Grid resource brokers and schedulers used in 2nd generation projects

Batch and Scheduling system

- Several systems are available with primary focus of batching and scheduling the resources.
- Examples of such systems are as follow
    - Condor
    - PBS (Portable batch system)
    - SGE (The sun grid engine)
    - LSF (Load sharing facility)

SRB (Storage resource broker)

- Provides uniform access to the storage that is distributed over a network using an API.
- Also supports metadata of the file system.
- Nimrod/G is an example of storage resource broker.
- Its components are as follow

- Task-farming engine
- Dispatcher
- Resource agents
- Scheduler

Role of grid portals in 2nd generation projects:

- They provided single point of access to compute-based resources
- They also give simplified access of the distributed resources across different member organizations.
- It is also possible to view the distributed resources as an integrated grid system or as separate machines.

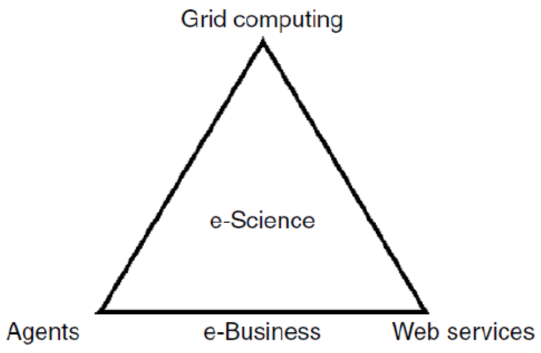
The main design features required at the data and computational fabric of the Grid are as follows:

- Administrative hierarchy
- Communication services
- Information services
- Naming services
- Distributed file systems and caching
- Security and authorization
- System status and fault tolerance
- Resource management and scheduling
- User and administrative GUI

7.3 Third generation

Service-oriented model

- The service-oriented paradigm provides the flexibility required for the third-generation Grid.
- Figure below depicts the three technologies (Grid, Agents, Web services).



Web services

- The creation of Web services standards an initiative by industry, with some of the emerging standards in various states of progress through the World Wide Web

Consortium (W3C).

- The established standards include the following:
    - **SOAP (XML protocol):** Simple object access protocol (SOAP) provides an envelope that encapsulates XML data for transfer through the Web.
    - **Web services description language (WSDL):** Describes a service in XML, using an XML Schema; there is also a mapping to the RDF (Resource Description Framework).
    - **Universal description discovery and integration (UDDI):** This is a specification for distributed registries of Web services (like yellow and white pages services).
    - UDDI supports ‘publish, find and bind’.
    - A service provider describes and publishes the service details to the directory, service requestors make requests to the registry to find the providers of a service, the services ‘bind’ using the technical details provided by UDDI.

- The next Web service standards attracting interest are at the process level.
- For example, Web Services Flow Language (WSFL) defines workflows as combinations of Web services and enables workflows to appear as services.
- Web services are closely aligned to the third-generation Grid requirements:
  o They support a service-oriented approach
  o They adopt standards to facilitate the information aspects such as service description

The Open Grid Services Architecture (OGSA) framework

- The OGSA Framework supports the creation, maintenance, and application of ensembles of services maintained by Virtual Organizations (VOs).
- Here a service is defined as a network-enabled entity that provides some capability, such as computational resources, storage resources, networks, programs and databases.

  Followings are the standard interfaces defined in OGSA:
    - Discovery:
    - Dynamic service creation:
    - Notification:
    - Manageability:
    - Simple hosting environment:
    - The Grid resource allocation and management (GRAM) protocol.
    - The information infrastructure, meta-directory service (MDS-2)
    - The Grid security infrastructure (GSI)

- The future implementation of Globus Toolkit may be based on the OGSA architecture.

Agents

- Web services provide a means of interoperability, However, Web services do not provide a new solution to many of the challenges of large-scale distributed systems, or provide new techniques for the engineering of these systems.

- Hence, it is important to look at other service-oriented models that are agent-based computing.
- The agent-based computing paradigm provides a perspective/view on software systems in which entities typically have the following properties, also known as weak agency.
    - Autonomy: Agents operate without intervention and have some control over their actions and internal state
    - Social ability: Agents interact with other agents using an agent communication language
    - Reactivity: Agents perceive and respond to their environment
    - Pro-activeness: Agents exhibit goal-directed behavior
- For interoperability between components, agreed common vocabularies are required which are provided by Agent Communication Languages (ACLs).

1. The Grid Context

- Grid Computing is an approach for building dynamically constructed problem-solving environments
- It contains geographically and organizationally dispersed, high-performance computing and data handling resources.
- Grids also provide important infrastructure supporting multi-institutional collaboration.
- The overall motivation for most current Grid projects is to enable the resource and human interactions that facilitate various scientific and computational fields.

  Functionally, Grids are tools, middleware, and services for,
  - Building the application frameworks that allow disciplined scientists to express and manage the simulation, analysis, and data management aspects of overall problem solving
  - Providing a uniform and secure access to a wide variety of distributed computing and data resources
  - Supporting construction, management, and use of widely distributed application systems
  - Facilitating human collaboration through common security services, and resource and data sharing
  - Providing support for remote access to, and operation of, scientific and engineering instrumentation systems
  - Managing and operating this computing and data infrastructure as a persistent service

  Two aspects of Grid:
  1. A set of uniform software services that manage and provide access to heterogeneous, distributed resources
  2. A widely deployed infrastructure

- Grid software is not a single, monolithic package, but rather a collection of interoperating software packages.
- The reason is that the Globus software is modularized and distributed as a collection of independent packages, and as other systems are integrated with basic Grid services.
- There is a set of basic functions that all Grids must have in order to be called a Grid: The Grid Common Services.

- **These common services include:**
  - The Grid Information Service ('GIS' – the basic resource discovery mechanism)
  - The Grid Security Infrastructure ('GSI' – the tools and libraries that provide Grid security)
  - The Grid job initiator mechanism (e.g. Globus GRAM)
  - A Grid scheduling function
  - A basic data management mechanism such as GridFTP.

- The software architectural hierarchy of a Grid is depicted in the figure below.

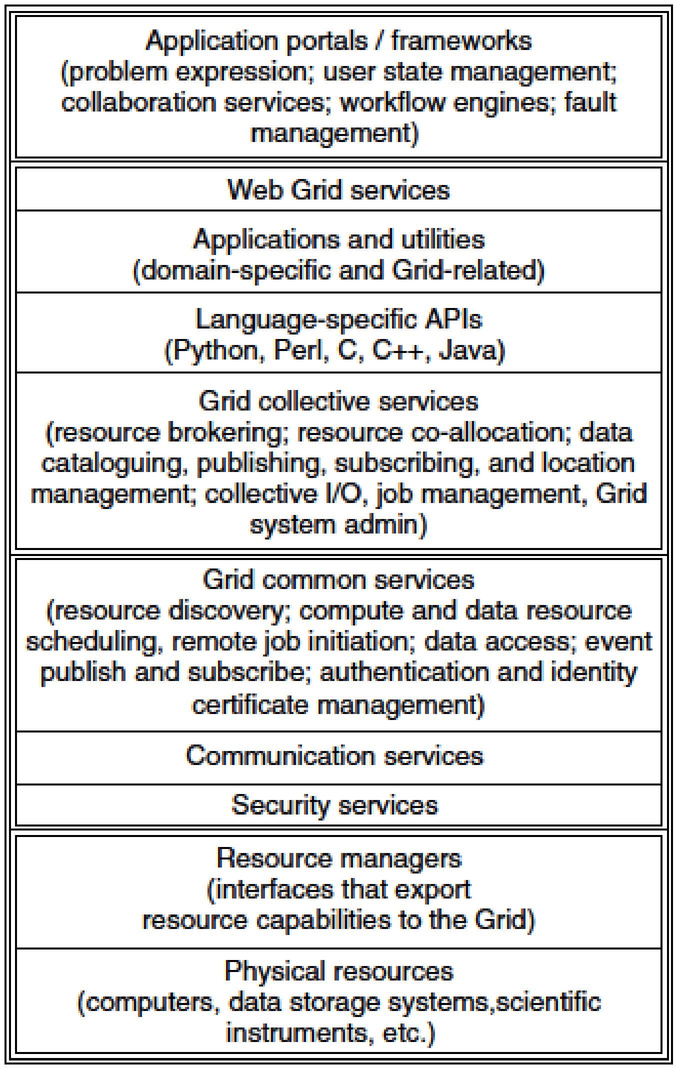

Application portals / frameworks (problem expression; user state management; collaboration services; workflow engines; fault management)
Web Grid services
Applications and utilities (domain-specific and Grid-related)
Language-specific APIs (Python, Perl, C, C++, Java)
Grid collective services (resource brokering; resource co-allocation; data cataloguing, publishing, subscribing, and location management; collective I/O, job management, Grid system admin)
Grid common services (resource discovery; compute and data resource scheduling, remote job initiation; data access; event publish and subscribe; authentication and identity certificate management)
Communication services
Security services
Resource managers (interfaces that export resource capabilities to the Grid)
Physical resources (computers, data storage systems,scientific instruments, etc.)

Figure 5.1 Grid Architectural Hierarchy

2. Grid Computing Models

- There are a several computing models in Grids that range from single resource to tightly coupled resources, and each requires some variations in Grid services.
- It means while the basic Grid services provide all the support needed to execute a distributed program, things like coordinated execution of multiple programs or management of many thousands of parameter study or data analysis jobs, will require following additional services.
    - Export existing services
    - Loosely coupled processes
    - Workflow managed processes

- Distributed-pipelined/coupled processes
- Tightly coupled processes

2.1 Grid Data Models

- Many of the current production Grids are greatly interested in wide-area data management.
- For example, Particle Physics Data Grid (PPDG), Grid Physics Network (GriPhyN), and the European Union DataGrid.
- There are several styles of data management in Grids, and these styles result in following different requirements for the software of a Grid.

- **Occasional access to multiple tertiary storage systems**
    - Data mining, as, for example, in Reference, can require access to metadata and uniform access to multiple data archives.
    - SRB/MCAT provides capabilities that include uniform remote access to data and local caching of the data for fast and/or multiple accesses.
    - Through its metadata catalogue, SRB provides the ability to federate multiple tertiary storage systems

- **Distributed analysis of massive datasets followed by cataloguing and archiving**
    - These disciplines are driving the development of data management tools for the Grid that provides naming and location transparency, and replica management for very large data sets.

- **Large reference data sets**
    - A common situation is that a whole set of simulations or data analysis programs will require the use of the same large reference dataset.
    - The management of such datasets, the originals of which almost always live in a tertiary storage system, could be handled by one of the replica managers.
    - However, another service that is needed in this situation is a network cache: a unit of storage that can be accessed and allocated as a Grid resource, and that is located ‘close to’ (in the network sense) the Grid computational resources that will run the codes that use the data.

- **Grid metadata management**
    - The Metadata Catalogue of SRB/MCAT provides a powerful mechanism for managing all types of descriptive information about data: data content information, fine-grained access control, physical storage device (which provides location independence for federating archives), and so on.

3. Building and Initial Multisite, Computation & Data Grid

3.1 The Grid building team

- The concept of an Engineering WG has proven successful as a mechanism for promoting cooperation and mutual technical support among those who will build and manage the Grid.
- The WG involves the Grid deployment teams at each site and meets weekly via teleconference.
- There should be a designated WG lead responsible for the agenda and managing the discussions.
- If at all possible, involve some Globus experts at least during the first several months while people are coming up to speed.
- There should also be a WG mail list that is archived and indexed by thread. Notes from the WG meetings should be mailed to the list.
- This, then, provides a living archive of technical issues and the state of your Grid.
- Grid software involves not only root-owned processes on all the resources but also a trust model for authorizing users that is not typical.
- Local control of resources is maintained, but is managed a bit differently from current practice.
- It is therefore very important to set up liaisons with the system administrators for all systems that will provide computation and storage resources for your Grid.
- This is true whether or not these systems are within your organization.

3.2 Grid resources

- As early as possible in the process, identify the computing and storage resources to be incorporated into your Grid.
- In doing this be sensitive to the fact that opening up systems to Grid users may turn lightly or moderately loaded systems into heavily loaded systems.
- Batch schedulers may have to be installed on systems that previously did not use them in order to manage the increased load.
- When choosing a batch scheduler, carefully consider the issue of co-scheduling!
- Many potential Grid applications need this, for example, to use multiple Grid systems to run cross system MPI jobs or support pipelined applications as noted above, and only a few available schedulers currently provide the advance reservation mechanism that is used for Grid co-scheduling (e.g. PBSPro and Maui).
- If you plan to use some other scheduler, be very careful to critically investigate any claims of supporting co-scheduling to make sure that they actually apply to heterogeneous Grid systems.

3.3 Build the initial test bed

3.3.1 Grid information service

- The Grid Information Service provides for locating resources based on the characteristics needed by a job (OS, CPU count, memory, etc.).

- The Globus MDS [25] provides this capability with two components.
- The Grid Resource Information Service (GRIS) runs on the Grid resources (computing and data systems) and handles the soft-state registration of the resource characteristics.
- The Grid Information Index Server (GIIS) is a user accessible directory server that supports searching for resource by characteristics.
- Other information may also be stored in the GIIS, and the GGF, Grid Information Services group is defining schema for various objects [64].
- Plan for a GIIS at each distinct site with significant resources.

- This is important in order to avoid single points of failure, because if you depend on a GIIS at some other site and it becomes unavailable, you will not be able to examine your local resources.
- Depending upon the number of local resources, it may be necessary to set up several GIISs at a site in order to accommodate the search load.
- The initial test bed GIS model can be independent GIISs at each site. In this model, either cross-site searches require explicit knowledge of each of the GIISs that have to be searched independently or all resources cross-register in each GIIS.

3.3.2 Build Globus on test systems

- Use PKI authentication and initially use certificates from the Globus Certificate Authority ('CA') or any other CA that will issue you certificates for this test environment.
- Then validate access to, and operation of the, GIS/GIISs at all sites and test local and remote job submission using these certificates.

4. Cross-site trust management

- Availability of large-scale collaboration between nodes is the great contribution of grid.
- It can be done by grid entity like naming and authentication mechanisms provided by the GSI (Global system information).
- For success of this mechanism, the collaborating sites/organizations must establish mutual trust in the authentication process.
- The widely used software mechanisms are PKI (Public key infrastructure) and, X.509 identity certificates.
- Their use in the GSI through Transport Layer Security (TLS)/Secure Sockets Layer (SSL), are understood and largely accepted.
- The real issue is that of establishing trust in the process that each CA (Certification authority) uses.
- CA is used for issuing the identity certificates to users and other entities, such as host systems and services.
- This involves following **two steps**, which are defined in CA policy.
    1. The 'physical' identification of the entities, verification of their association with the VO that is issuing identity certificates, and then the assignment of an appropriate name.
    2. The process by which an X.509 certificate is issued.

Trust

- Trust is confidence on some quality or attribute of a person or thing, or the truth of a statement.
- Cyberspace trust starts with clear, transparent, negotiated, and documented policies associated with identity.
- It is relatively easy to establish a policy for homogeneous communities, such as in a single organization, because an agreed upon trust model will probably already exist.
- It is difficult to establish trust for large, heterogeneous VOs involving people from multiple, international institutions, because the shared trust models do not exist.
- The typical **issues** related to establishing trust may be summarized as follows:
    - Across administratively similar systems
    - Administratively diverse systems
    - Administratively heterogeneous

Establishing an operational CA

- Set up, or identify, a Certification Authority to issue Grid X.509 identity certificates to users and hosts.
- Both the IPG (Information power grid) and DOE (Department of energy) Science Grids use the Netscape CMS (Certificate management system) software for their operational CA.
- Make sure that you understand the issues associated with the CP of your CA.
- Do not try and invent your own CP.
- The GGF is working on a standard set of CPs that can be used as templates, and the DOE Science Grid has developed a CP that supports international collaborations.
- Think carefully about the space of entities for which you will have to issue certificates.
- These typically include human users, hosts (systems), services (e.g. GridFTP), and possibly security domain gateways (e.g. the PKI to Kerberos gateway, KX509 [70]).
- Establish and publish your Grid CP as soon as possible so that the issues involved can be addressed.

Naming

- One of the important issues in developing a CP is the naming of the principals.
- There is an almost universal tendency to try and pack a lot of information into the subject name.
- Increasingly there is an understanding that the less information of any kind put into a certificate, the better.
- This simplifies certificate management and re-issuance when users forget pass phrases.
- The CA run by ESnet for the DOE Science Grid, for example, will serve several dozen different VO, several of which are international in their makeup.
- The certificates use a namespace, with a ‘reasonable’ common name (e.g. a ‘formal’ human name) to which has been added a random string of alphanumeric digits to ensure name uniqueness.
- Consider their full organizational hierarchy in defining the naming hierarchy.

4.1 The certification authority model

- There are several models for certification authority (CA).
- One such model has a central CA that has an overall CP and subordinate policies for a collection of VOs.

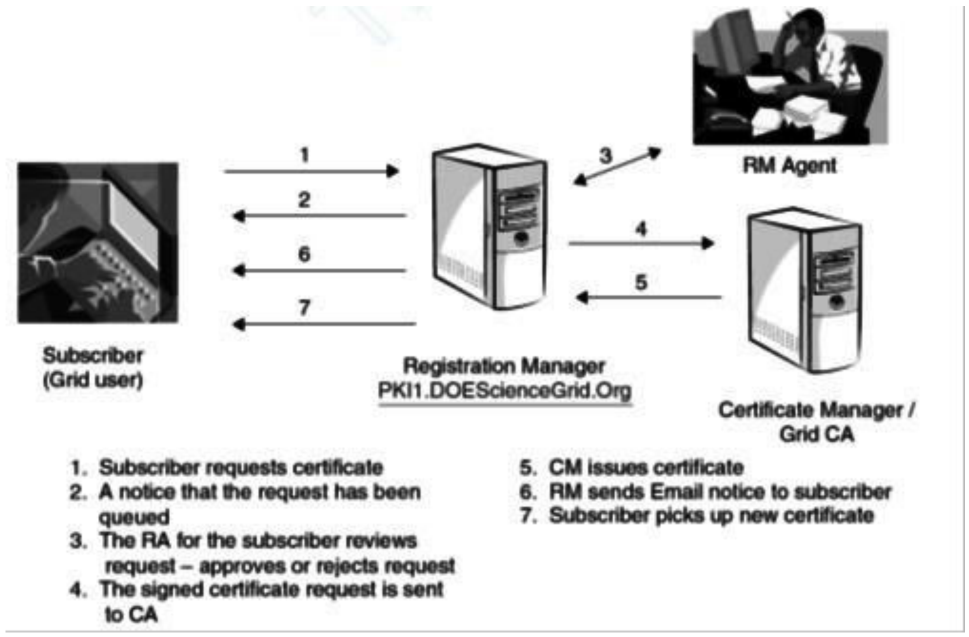

1
2
6
7
3
4
5
RM Agent
Subscriber (Grid user)
Registration Manager
PKI1.DOEScienceGrid.Org
Certificate Manager / Grid CA
1. Subscriber requests certificate
2. A notice that the request has been queued
3. The RA for the subscriber reviews request – approves or rejects request
4. The signed certificate request is sent to CA
5. CM issues certificate
6. RM sends Email notice to subscriber
7. Subscriber picks up new certificate

Figure 5.1 Certificate issuing process

- Each VO has an appendix in the CP that describes VO specific issues.
- VO Registration Agents are responsible for applying the CP identity policy to their users and other entities.
- Once satisfied, the RA authorizes the CA to issue (generate and sign) a certificate for the subscriber.
- This is the model of the DOE Science Grid CA, for example, and it is intended to provide a CA that is scalable to dozens of VO and thousands of users.
- The process of issuing a certificate to a user ('subscriber') is indicated in the figure below.

Example: CDAC Grid Computing

- Garuda is India's national grid infrastructure of HPC systems, connecting 70 academic and research institutions across 17 cities of the country with India's Nation Knowledge Network (NKN).
- The Indian Grid Certification Authority (IGCA) accredited by the APGrid-PMA (Asia Pacific Grid Policy Management) has been set up in C-DAC.
- The IGCA helps scientists, researchers and collaborative community in India and neighboring countries to obtain an internationally recognized digital certificate to interoperate with state-of-the-art grids worldwide.
- Some of the applications that make use of this grid infrastructure include:
    - Open Source Drug Discovery (OSDD)

- Collaborative Class Room
- Computer Aided Engineering
- Oncology Research
- Disaster Management using Synthetic Aperture Radar
- Biodiversity Conservation

5. Transition to a Prototype Production Grid

1. First steps

- Issue host certificates for all the computing and data resources and establish procedures for installing them.
- Issue user certificates.
- Count on revoking and re-issuing all the certificates at least once before going operational.
- This is inevitable if you have not previously operated a CA.
- Using certificates issued by your CA, validate correct operation of the GSI [72], GSS libraries, GSISSH [62], and GSIFTP [73] and/or GridFTP [28] at all sites.
- Start training a Grid application support team on this prototype.

2. Defining/understanding the extent of 'your' Grid

- The apparent 'boundaries' of most Grids depend on who is answering the question.
- The 'boundaries' of a Grid are primarily determined by three factors:
  - Interoperability of the Grid software
  - What CAs you trust
  - How you scope the searching of the GIS/GIISs or control the information that is published in them.

3. The model for the Grid Information System

- Directory servers above the local GIISs (resource information servers) are an important scaling mechanism for several reasons.
- Directory servers above the local GIISs (resource information servers) are an important scaling mechanism for several reasons.
- There are currently two main approaches that are being used for building directory services above the local GIISs.
  1. An X.500 style hierarchical name component space directory structure
  2. Index server directory structure

4. Local authorization

- Almost all current Grid software uses some form of access control lists ('ACL'), which is straightforward, but typically does not scale very well.
- The Globus mapfile is an ACL that maps from Grid identities (the subject names in the identity certificates) to local user identification numbers (UIDs) on the systems where jobs are to be run.
- The mapfile mechanism is fine in that it provides a clear-cut way for locally controlling access to a system by Grid users.

- The first step in the mapfile management process is usually to establish a connection between user account generation on individual platforms and requests for Globus access on those systems.

5. Site security issues

- Incorporating any computing resource into a distributed application system via Grid services involves using a whole collection of IP communication ports that are otherwise not used.
- If your systems are behind a firewall, then these ports are almost certainly blocked, and you will have to negotiate with the site security folks to open the required ports.
- Globus can be configured to use a restricted range of ports, but it still needs several tens, or so, in the mid-700s.
- The number depending on the level of usage of the resources behind the firewall.
- A Globus 'port catalogue' is available to tell what each Globus port is used for, and this lets you provide information that your site security folks will probably want to know.
- Alternate approaches to firewalls have various types of service proxies manage the intra-service component communication so that one, or no, new ports are used.

6. High performance communications issues

- If you anticipate high data-rate distributed applications, whether for large-scale data movement or process-to-process communication, then enlist the help of a WAN networking specialist and check and refine the network bandwidth end-to-end using large packet size test data streams.
- To identifying problems in network and system hardware and configurations, there are a whole set of issues relating to how current TCP algorithms work, and how they must be tuned in order to achieve high performance in high-speed, wide-area networks.

7. Batch schedulers

- There are several functions that are important to Grids that Grid middleware cannot emulate: these must be provided by the resources themselves.
- Some of the most important of these are the functions associated with job initiation and management on the remote computing resources.
- Development of the PBS batch scheduling system was an active part of the IPG project, and several important features were added in order to support Grids.
- For access control, every queue in PBS has an ACL that can include and exclude specific users and groups.

8. Preparing for users

- Try and find problems before your users do. Design test and validation suites that exercise your Grid in the same way that applications are likely to use your Grid.
- As early as possible in the construction of your Grid, identify some test case distributed applications that require reasonable bandwidth and run them across as many widely separated systems in your Grid as possible, and then run these test cases every time

something changes in your configuration.

- Establish user help mechanisms, including a Grid user e-mail list and a trouble ticket system.
- Provide user-oriented Web pages with pointers to documentation.

9. Moving from test bed to prototype production Grid

- At this point, Globus, the GIS/MDS, and the security infrastructure should all be operational on the test bed system(s).
- The Globus deployment team should be familiar with the install and operation issues and the system admins of the target resources should be engaged.
- Deploy and build Globus on at least two production computing platforms at two different sites. Establish the relationship between Globus job submission and the local batch schedulers (one queue, several queues, a Globus queue, etc.).
- Validate operation of this configuration.

10. Grid systems administration tools

- Grids present special challenges for system administration owing to the administratively heterogeneous nature of the underlying resources.
- The harder issues in Grid Admin tools revolve around authorization and privilege management across site boundaries.
- Tools and techniques are being developed for extending Trouble Ticketbased problem tracking systems to the Grid environment.

11. Data management and your Grid service model

- Establish the model for moving data between all the systems involved in your Grid.
- GridFTP servers should be deployed on the Grid computing platforms and on the Grid data storage platforms.

12. Take good care of the users as early as possible

- Identify specific early users and have the Grid application specialists encourage/assist them in getting jobs running on the Grid.
- One approach that we have seen to be successful in the IPG and DOE Science Grid is to encourage applications that already have their own ‘frameworks’ to port those frameworks on the Grid.
- This is typically not too difficult because many of these frameworks already have some form of resource management built in, and this is easily replaced/augmented with Grid resource management functions. This hides some of the ‘low-level functionality’ problem.

1. Virtual Organizations (VOs)

- In grid computing, a **virtual organization (VO)** refers to a dynamic set of individuals or institutions defined around a set of resource-sharing rules and conditions.
- All these VOs share many common utilities among them, including common concerns and requirements.
- But all utilities may vary in size, scope, duration, sociology, and structure.
- A virtual organization has the characteristics of a formal organization while not being one.
- It comprises a complex network of smaller organizations which each contribute a part of the production process.
- Boundaries between organizations are fuzzy; control is generally by market forces, reinforced by the certainty of long- term contracts.

Example: EGI (European Grid Infrastructure)

- Researchers must join a VO in order to use grid computing resources provided by EGI.
- Each virtual organisation manages its own membership list, according to the VO's requirements.
- EGI provides support, services and tools to allow VOs to make the most of their resources.
- EGI currently hosts more than 200 VOs for communities with interests as diverse as Earth Sciences, Computer Sciences and Mathematics, Fusion, Life Sciences or High-Energy Physics.

How to join a VO of EGI?

- For that you have to consult the list of VOs and select which one is relevant to your research work (you can filter the VOs by scientific discipline, or by country).
- Contact the VO manager. Each VO has different membership procedures and the VO manager will walk you through the process.

1.1 Protocol, Services and Tools to build VOs

- Research and development have produced protocols, services, and tools that address challenges that arise while building scalable VOs.
- Various tools are used to provide communication between virtual organizations.
- For example, Virtual reality tools, Social networking tools, Video conferencing etc.
- Such research includes security solutions that support management of credentials and policies when computations span multiple institutions.
- For example, Resource management, Information query protocols and Data management services provide such functionalities.
- **Resource management protocols and services:** It supports secure remote access to computing and data resources and the co-allocation of multiple resources.
- **Information query protocols and services:** It provides configuration and status information about resources, organizations, and services.
- **Data management services:** It locates and transport datasets between storage systems and applications.

1.2 Examples of VOs

- Each of following examples represents an approach to computing and problem solving.
- All of them are based on collaboration in computation and data-rich environments.
- Following are few examples of VO:
  - The application service providers.
  - Storage service providers.
  - Cycle providers, and consultants engaged by a car manufacturer to perform scenario evaluation during planning for a new factory.
  - Members of industrial consortium bidding on a new aircraft.
  - A crisis management team and the databases and simulation systems
  - Members of large, international, multiyear high-energy physics collaboration.

1.3 The emergence of VOs with Example Scenarios

- For study of emergence of VOs, here some scenarios are given below to make it more understandable.

Following are some various scenarios for emergence of VOs:

1. A company needs to reach a decision on the placement of a new factory

- It invokes a sophisticated financial forecasting model from an ASP (Application service provider).
- It is provided with access to appropriate proprietary historical data from a corporate database on storage systems operated by an SSP (Shared Socioeconomic Pathways).
- The division heads participating in the decision may located in different cities.
- The ASP itself contracts with a cycle provider for additional energy during particularly demanding scenarios.
- Those cycles must meet desired security and performance requirements.

2. An industrial consortium formed to develop a feasibility study for a next-generation supersonic aircraft undertakes a highly accurate multidisciplinary simulation.

- This simulation integrates proprietary software components developed by different participants.
- Each of this components operating on that participant’s computers.
- All of them are having access to appropriate design databases made available to the consortium by its members.

3. A crisis management team responds to a chemical spill by,
   - Using local weather and soil models to estimate the spread of the spill.
   - Determining the impact based on population location as well as geographic features such as rivers and water supplies.
   - Creating a short-term mitigation plan (perhaps based on chemical reaction models).
   - Tasking emergency response personnel by planning and coordinating evacuation, notifying hospitals, and so on.

4. Thousands of physicists come together to design, create, operate, and analyze the products of a major detector at CERN (European Organization for Nuclear Research).

- During the analysis phase, they pool their computing, storage, and networking resources to create a 'Data Grid' capable of analyzing petabytes of data.

Difference between such scenarios:

- These four examples differ in many respects as follow:
  - The number and type of participants.
  - The types of activities.
  - The duration.
  - Scale of the interaction, and the resources being shared.

Similarity in resource sharing for such scenarios:

- All of the scenarios have things in common also. For example following resource sharing flow.
- It can involve direct access to remote software, computers, data, sensors, and other resources.
- For example, members of a consortium may provide access to specialized software and data and/or pool their computational resources.
- In each case, a number of mutually distrustful participants want to share resources in order to perform some task. Sharing is about more than simply document exchange.
- It can involve direct access to remote software, computers, data, sensors, and other resources.

Example of Resource sharing between VOs:

- Above scenarios has much in common, as discussed in the following (Figure 6.1).
- Resource sharing can also be explained for above scenarios by below example (Figure 6.1).

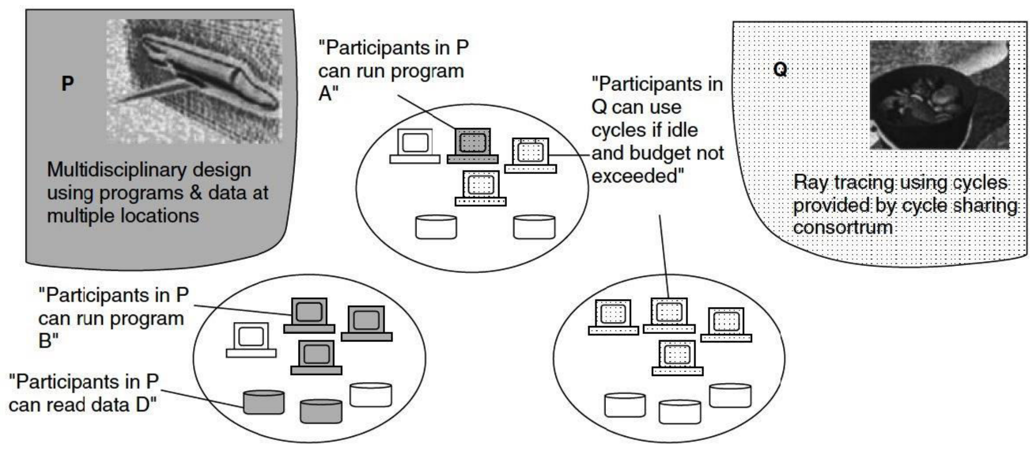

P
Multidisciplinary design using programs & data at multiple locations
"Participants in P can run program A"
"Participants in Q can use cycles if idle and budget not exceeded"
Q
Ray tracing using cycles provided by cycle sharing consortrum
"Participants in P can run program B"
"Participants in P can read data D"

Figure 6.1 An actual organization can participate in one or more VOs by sharing some or all of its resources.

- In above figure, three actual organizations (the ovals), and two VOs are there.
- P, links participants in an aerospace design consortium.
- Q, links colleagues who have agreed to share spare computing cycles, for example, to run ray tracing computations.
- The organization on the left participates in P, the one to the right participates in Q, and the third is a member of both P and Q.
- The policies governing access to resources (summarized in quotes) vary according to the actual organizations, resources, and VOs involved.

2. Nature of Grid Architecture

- Sharing relationships of Dynamic, cross-organizational VOs require new technologies.
- Discussion of these technologies can be structured in terms of grid architecture.
- **Grid architecture** defines following
  - It Identifies fundamental system components
  - It Specifies the purpose and function of these components
  - It Indicates how these components interact with one another
- A standards-based open architecture facilitates extensibility, interoperability, portability, and code sharing.
- Standard protocols make it easy to define standard services that provide enhanced capabilities.
- We can also construct application programming interfaces and software development kits to provide the programming abstractions required to create a usable Grid.
- Together, this technology and architecture constitute middleware.
- We discuss each of these points in the following.

Why is interoperability such a fundamental concern of grid architecture?

- Various mechanisms give fewer benefits if they are not defined and implemented as interoperable across various organizations.
- Without interoperability, VO applications and participants are forced to enter into multiple sharing arrangements.
- Because there is no assurance that the mechanisms used between any two parties will extend to any other parties.
- Without such assurance, dynamic VO formation is all but impossible.

Why are protocols critical to interoperability in grid architecture?

- A protocol definition specifies how distributed system elements interact with one another.
- It also specifies behavior, and the structure of the information exchanged during this interaction.
- This focus on externals (interactions) rather than internals (software, resource characteristics) has important pragmatic benefits.
- VOs needs easy flow, so the mechanisms used to discover resources, establish identity,

determine authorization, and initiate sharing must be flexible and lightweight.

Why are services important for grid architecture?

- A service is defined solely by the protocol that it speaks and the behaviors that it implements.
- The definitions of standard services include access to computation, access to data, resource discovery, scheduling, data replication, and so on.
- All of them allow us to enhance the services offered to VO participants and also to abstract away resource-specific details.

Why do we also consider application programming interfaces (APIs) and software development kits (SDKs)?

- Some additional efforts are required to VOs than interoperability, protocols, and services.
- Developers must be able to develop sophisticated applications in complex and dynamic execution environments.
- Users must be able to operate these applications.
- Application robustness, correctness, development costs, and maintenance costs are all important concerns.
- Standard abstractions, APIs, and SDKs can accelerate code development, enable code sharing, and enhance application portability.
- APIs and SDKs are an adjunct to, not an alternative to, protocols.
- Without standard protocols, interoperability can be achieved at the API level only by using a single implementation everywhere.

3. Grid Architecture with Layered Architectural Description

- Our goal in describing Grid architecture to identify requirements for general classes of components.
- Grid architecture can be drawn in **two different patterns.**
  1. Hourglass architecture (Figure 6.2)
  2. Layered architecture (Figure 6.3)

- Components within each layer share common characteristics but can build on capabilities and behaviors provided by any lower layer.
- In specifying the various layers of the Grid architecture, we follow the principles of the ‘hourglass model’.
- The narrow neck of the hourglass defines a small set of core abstractions and protocols.
- By definition, the number of protocols defined at the neck must be small.
- The neck of the hourglass consists of Resource and Connectivity protocols, which facilitate the sharing of individual resources.
- Protocols at these layers are designed so that they can be implemented on top of a diverse range of resource types, defined at the Fabric layer, and can in turn be used to construct a wide range of global services and application specific behaviors at the Collective.

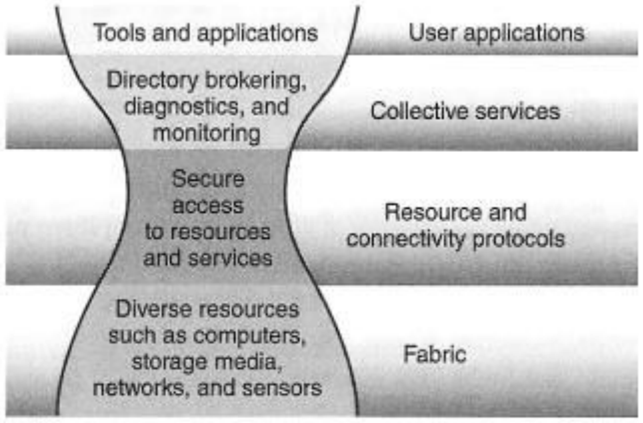

Tools and applications
User applications
Directory brokering, diagnostics, and monitoring
Collective services
Secure access to resources and services
Resource and connectivity protocols
Diverse resources such as computers, storage media, networks, and sensors
Fabric

Figure 6.2 Grid Architecture (Hourglass model)

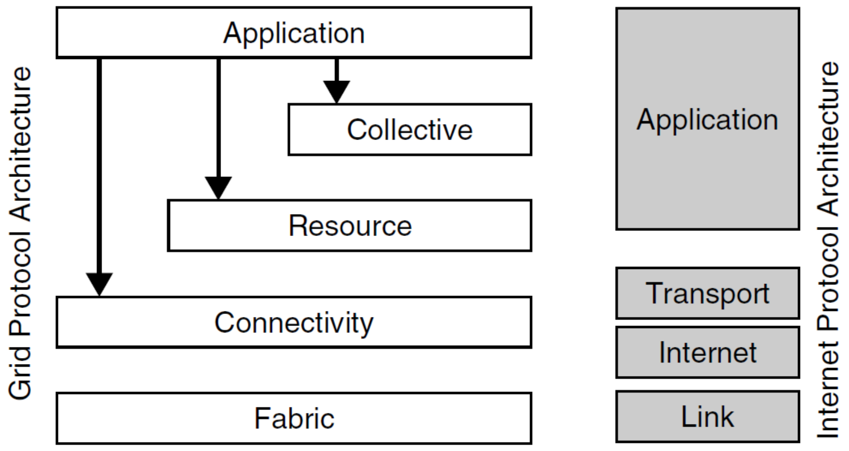


Figure 6.3 Grid Architecture(Layered)

Each of the layer of architecture from bottom to top can be described in brief as follows:

Fabric layer: Interfaces to local control

- The Grid Fabric layer provides the resources to which shared access is mediated by Grid protocols.
- For example, computational resources, storage systems, catalogs, network resources, and sensors.
- A ‘resource’ may be a logical entity, such as a distributed file system, computer cluster, or distributed computer pool.
- Richer Fabric functionality enables more sophisticated sharing operations.
- For example, at the same time, if we place few demands on Fabric elements, then deployment of Grid infrastructure is simplified.
- For example, resource-level support for advance reservations makes it possible for higher-level services to aggregate (co-schedule) resources.

The following list provides a resource-specific characterization of capabilities.

Computational resources:

- For such resources mechanisms are required for starting programs and for monitoring and controlling the execution of the resulting processes.
- Management mechanisms that allow control over the resources allocated to processes are useful, as are advance reservation mechanisms.
- Enquiry functions are needed for determining hardware and software characteristics.
- Enquiry functions also give relevant state information such as current load and queue state in the case of scheduler-managed resources.

Storage resources:

- For storage resources mechanisms are required for putting and getting files.
- Third-party and high-performance (e.g., striped) transfers are also useful.
- Management mechanisms that allow control over the resources allocated to data transfers are useful.
- Enquiry functions are needed for determining hardware and software characteristics.
- Such characteristics include relevant load information such as available space and bandwidth utilization.

Network resources:

- Management mechanisms that provide control over the resources allocated to network transfers (e.g., prioritization, reservation) can be useful.
- Enquiry functions should be provided to determine network characteristics and load.

Code repositories:

- This specialized form of storage resource requires mechanisms for managing versioned source and object code: for example, a control system such as CVS.

Catalogs:

- This specialized form of storage resource requires mechanisms for implementing catalog query and update operations: for example, a relational database.

Connectivity layer: Communicating easily and securely

- The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions.
- Communication protocols enable the exchange of data between Fabric layer resources.
- Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.
- Communication requirements include transport, routing, and naming.
- While alternatives certainly exist, we assume here that these protocols are drawn from the TCP/IP protocol stack.

- With communication, many of the security standards developed within the context of the Internet protocol suite are applicable.
- Authentication solutions for VO environments should have the following characteristics.
  - Single sign-on
  - Delegation
  - Integration with various local security solutions
  - User-based trust relationships

- Grid security solutions should also provide flexible support for communication protection.
- It enables stakeholder control over authorization decisions, including the ability to restrict the delegation of rights in various ways.

Resource layer: Sharing single resources

- The Resource layer builds on Connectivity layer communication and authentication protocols to define protocols (and APIs and SDKs).
- It is used for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources.
- It implements these protocols call Fabric layer to access and control local resources.

  Two primary classes of Resource layer protocols can be distinguished:
  1. **Information protocols** are used to obtain information about the structure and state of a resource, for example, its configuration, current load, and usage policy (e.g., cost).
  2. **Management protocols** are used to negotiate access to a shared resource.

Collective layer: Coordinating multiple resources

- While the Resource layer is focused on interactions with a single resource, the next layer in the architecture contains protocols and services (and APIs and SDKs).
- For this reason, we refer to the next layer of the architecture as the Collective layer.
- Collective components build on the narrow Resource and Connectivity layer 'neck' in the protocol hourglass.
- They can implement a wide variety of sharing behaviors without placing new requirements on the resources being shared.
- For example:
  - Directory services
  - Co-allocation, scheduling, and brokering services
  - Monitoring and diagnostics services
  - Data replication services
  - Grid-enabled programming systems
  - Workload management systems and collaboration
  - Software discovery services
  - Community authorization servers
  - Community accounting and payment services
  - Collaboratory services

4. Intergrid Protocols

- Grid Protocol is a protocol that mediates between user and resource during access to any resource in Grid environment.
- Grid architecture establishes requirements for the protocols and APIs that enable sharing of resources, services, and code.
- It forms the technologies that might be used to implement these protocols and APIs.
- It is quite feasible to define multiple instantiations of key Grid architecture elements.
- For example, we can construct both Kerberos and PKI-based protocols at the Connectivity layer – and access these security mechanisms via the same API, thanks to GSS-API (see Appendix).
- Grids constructed with these different protocols are not interoperable and cannot share essential services – at least not without gateways.
- For this reason, the success of Grid computing requires that we select and achieve widespread deployment of one set of protocols at the Connectivity and Resource layers.
- It forms lesser extent to collective layer.
- Much as the core Internet protocols enable different computer networks to interoperate and exchange information.
- Such Intergrid protocols enable different organizations to interoperate and exchange or share resources.
- Resources that use these protocols can be said to be 'on the Grid.'
- Standard APIs are also highly useful if Grid code is to be shared.
- The identification of these Intergrid protocols and APIs was used in Globus Toolkit.

Categories of core Intergrid protocols:

1. Grid network communication protocol and Data transfer protocols
2. Grid information security protocols
    - Secure communication protocol
    - User proxy creation protocol
    - Global subject to local subject mapping registration protocol
    - Authentication protocol
    - Single sign-on protocol
    - Delegation protocol
3. Grid resource information protocols
    - Resource registration protocol
    - Resource discovery protocol
    - Resource inquiry protocol
4. Grid management protocols
    - Grid monitoring protocol
    - Job submission and job management protocol
    - Advance reservation and co-allocation protocol
    - SLA negotiation protocol
5. Grid interface protocols
    - Basic protocols for web service based interaction

- Protocol for stateful web service
  - Web services resource framework protocol
  - Web services addressing protocol
  - Web services notification protocol
- Protocols for web services security

5. Production Grid and Open Grid Service Architecture (OGSA)

- A production Gris is an infrastructure comprising a collection of multiple administrative domains.
- It provides a network that enables large-scale resource and human interactions in a virtual manner.
- The OGSA Framework supports the creation, maintenance, and application of ensembles of services maintained by Virtual Organizations (VOs).
- Here a service is defined as a network-enabled entity that provides some capability, such as computational resources, storage resources, networks, programs and databases.
- It tailors/modifies the Web services approach to meet some Grid specific requirements.

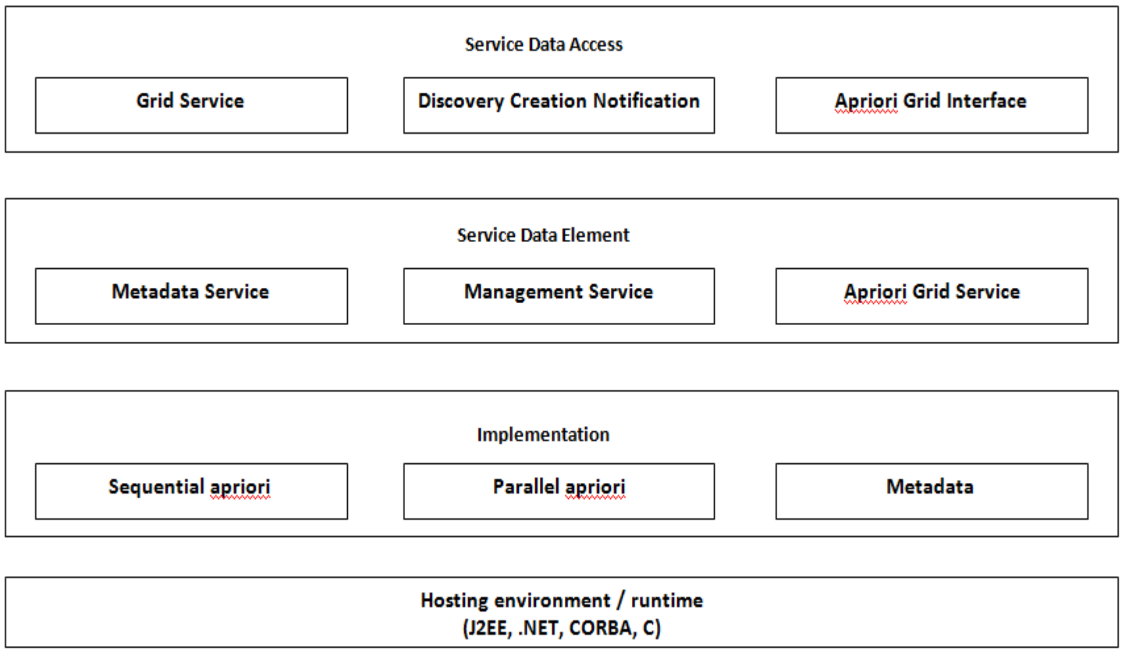


Figure 6.4 OGSA (Open Grid Service Architecture)

Following are the standard interfaces defined in OGSA:

- **Discovery:** Clients require mechanisms for discovering available services and for determining the characteristics of those services.
- **Dynamic service creation:** A standard interface (Factory) and semantics that any service creation service must provide.
- **Lifetime management:** In a system that incorporates transient and state-ful service

instances, mechanisms must be provided for reclaiming services and state associated with failed operations.

- **Notification:** A collection of dynamic, distributed services must be able to notify each other asynchronously of interesting changes to their state.
- **Manageability:** The operations relevant to the management and monitoring of large numbers of Grid service instances are provided.
- **Simple hosting environment:** A simple execution environment is a set of resources located within a single administrative domain and supporting native facilities for service management: for example, a J2EE application server, Microsoft. NET system etc.

- The parts of Globus that are impacted most by the OGSA are
    - The Grid resource allocation and management (GRAM) protocol.
    - The information infrastructure, meta-directory service (MDS-2)
    - The Grid security infrastructure (GSI)
    - The future implementation of Globus Toolkit may be based on the OGSA architecture.
- Examples of Production grid are: e-Science program, Asia pacific, NASA's IPG etc.

6. Other Perspectives on Grid

Critique perspective of grid can be described as follows:

- **The Grid is a next-generation Internet :**
    - Grid is not an alternative to the Internet, it is rather a set of additional protocols and services.
    - That builds on Internet protocols and services to support the creation and use of computation- and data-enriched environments.
    - Any resource that is 'on the Grid' is also, by definition, 'on the Net.'

- **The Grid is a source of free cycles:**
    - Grid computing does not imply unrestricted access to resources.
    - Grid computing is about controlled sharing.
    - Resource owners will typically want to enforce policies that constrain access according to group membership, ability to pay, and so forth.
    - Grid architecture must incorporate resource and collective protocols for exchanging usage and cost information.

- **The Grid makes high-performance computers superfluous:**
    - The hundreds, thousands, or even millions of processors that may be accessible within a VO represent a significant source of computational power.
    - This does not imply, however, that traditional high-performance computers are obsolete.
    - Many problems require tightly coupled computers, with low latencies and high communication bandwidths;

- Grid computing may increase, demand for such systems by making access easier.

Why grid computing requires distributed operating system and new programming model?

- **The Grid requires a distributed operating system:**
    - Grid software should define the operating system services to be installed on every participating system.
    - With these services providing for the Grid what an operating system provides transparency with respect to location, naming, security, and so on.
    - This perspective views the role of Grid software as defining a virtual machine.
    - Also this perspective is inconsistent with our primary goals of broad deployment and interoperability.
    - The tremendous physical and administrative heterogeneities encountered in Grid environments.
    - For that first all the node must be connected in distributed manner.
    - On the other hand, it does appear feasible to obtain agreement on standard protocols.
    - The grid architecture is deliberately open rather than prescriptive.

- **The Grid requires new programming models:**
    - Programming in Grid environments introduces challenges that are not encountered in sequential (or parallel) computers
    - That includes multiple administrative domains, new failure modes, and large variations in performance.
    - In once context, abstraction and encapsulation can reduce complexity and improve reliability.
    - But, in other contexts, it is desirable to allow a wide variety of higher-level abstractions to be constructed, rather than enforcing a particular approach.

  For example:
    - A developer believes that a universal distributed shared-memory model can simplify grid application development.
    - He should implement this model in terms of Grid protocols, extending or replacing those protocols only if they prove inadequate for this purpose.
    - Similarly, a developer who believes that all Grid resources should be presented to users as objects.
    - Than he needs simply to implement an object-oriented API in terms of Grid protocols.

7. WSRF (Web Service Resource Framework) for Grid

- Web Services Resource Framework (WSRF) is a family of OASIS-published specifications for web services. Major contributors include the Globus Alliance and IBM.
- A web service by itself is nominally stateless, i.e., it retains no data between invocations.

- WSRF provides a set of operations that web services may implement to become stateful.
- Web service clients communicate with resource services which allow data to be stored and retrieved.

Working of WSRF:

- When clients talk to the web service they include the identifier of the specific resource that should be used inside the request, encapsulated within the WS-Addressing endpoint reference.
- This may be a simple URIaddress, or it may be complex XML content that helps identify or even fully describe the specific resource in question.

WSRF component specifications:

- **WS-Resource:** defines a WS-Resource as the composition of a resource and a Web service through which the resource can be accessed.
- **WS-ResourceProperties:** describes an interface to associate a set of typed values with a WS-Resource that may be read and manipulated in a standard way.
- **WS-ResourceLifetime:** describes an interface to manage the lifetime of a WS-Resource.
- **WS-BaseFaults:** describes an extensible mechanism for rich SOAPFaults.
- **WS-ServiceGroup:** describes an interface for operating on collections of WS-Resources.

What is Cloud Computing?

- **The cloud** itself is a set of hardware, networks, storage, services, and interfaces that enable the delivery of computing as a service.
- Cloud services include the delivery of software, infrastructure, and storage over the Internet (either as separate components or a complete platform) based on user demand.
- It is a model for enabling ubiquitous (present everywhere) network access to a shared pool of configurable computing resources.

Four basic characteristics of the cloud:

- Elasticity and the ability to scale up and down
- Self-service provisioning and automatic deprovisioning
- Application programming interfaces (APIs)
- Billing and metering of service usage in a pay-as-you-go model

1. Basics of Cloud Computing

1.1 Benefits and Limitations of Cloud Computing

Benefits:

- **Cost reduction:** Cloud computing lowers transaction costs, and minimizes the investment in hardware.
- **Scalability**: Like electricity and water, some cloud computing services allow businesses to only pay for what they use.
- As your business grows, you can accommodate by adding more server space.
- **Levels the playing field** - Cloud computing providers offers small and mid-size businesses access to more sophisticated technology at lower prices.
- Sharing IT resources with other companies reduces the cost of licensing software and buying servers.
- **Easier collaboration** - Services in the cloud can be accessed anytime from any computer, so it’s easy to collaborate with employees in distant locations.
- The ability to **scale up IT capacity on-demand.**
- The ability to **align use of IT resources** directly with cost.
- The ability to provide **more IT agility** to adapt to new business opportunities.
- The ability to **mix and match the right solutions** for the business without having to purchase hardware & software.
- The ability to place **business volatility into a single domain** means on the cloud.
- The ability to **reduce operational costs,** while increasing IT effectiveness.
- The ability to **manage huge data sets.**
- The ability to create customer-facing systems quickly.
- The **ability balance processing** between on-premise and cloud systems.

Limitations:

- **Availability:** Rarely your cloud services may go down unexpectedly for maintenance, without concerning about the type or importance of work you are doing at present.

- **Data mobility and ownership**: Once you have cancelled the onwnership of the cloud, you will not able to
- **Privacy**: How much data are cloud companies collecting and how might that information be used?
- **Security** is largely immature, and currently requires specialized expertise.
- Your dependent on the cloud computing provider for your IT resources, thus you could be exposed around outages and other **service interruptions.**
- Using the Internet **can cause network latency** with some cloud applications.
- In some cases cloud providers are **more expensive** than on-premise systems.
- **Integration** between on-premise and cloud-based systems can be problematic.
- **Data privacy issues** could arise, if your cloud provider seeks to monetize the data in their system.

1.2 Convenient and Critical Situations for Use of Cloud Computing

1.2.1 Situations in which cloud computing should be chosen as a platform

When we need to meet following listed functionalities, We should choose Cloud computing as a platform.

- Security, privacy, and compliance, including data ownership and availability in shared environments, regulation and legal issues, corporate policies, and identity management for access control
- Lack of functionality or inflexibility of SaaS applications.
- Dependency on an Internet connection.
- Vendor lock-in as a result of lack of standards, and portability issues due to immaturity of cloud products.
- Vendor management, which requires a different approach, in which SLAs are critical
- Change management and testing, which can be a challenge, especially in shared environments
- Integration with on-premise systems, which may be difficult, or even impossible
- Lack of transparency of interfaces between SaaS vendors, particularly with regard to managing the interfaces
- Lack of experience with cloud financial models and licensing

1.2.1 Situations in which cloud computing cloud computing should not be used

If following issues or limitations are not affordable to you then it is critical to use cloud computing as a platform:

- Security is largely immature, and currently requires specialized expertise.

- Much of the technology is proprietary, and thus can cause lock-in.
- Using the Internet can cause network latency with some cloud applications.
- In some cases cloud providers are more expensive than on-premise systems.
- Not in control of costs if subscription prices go up in the future.
- Integration between on-premise and cloud-based systems can be problematic.
- Compliance issues could raise the risks of using cloud computing.
- Data privacy issues could arise, if your cloud provider seeks to monetize the data in their system.
- Maintenance activity around cloud providers could mean constantly adjusting and readjusting your cloud computing solutions.

1.3 Features of Cloud Computing that can benefit organization

1. Resource Pooling and Elasticity

- In cloud computing, resources are pooled to serve a large number of customers.
- Cloud computing uses multi-tenancy where different resources are dynamically allocated and de-allocated according to demand.

2. Self-Service and On-demand Services

- Cloud computing is based on self-service and on-demand service model.
- It should allow the user to interact with the cloud to perform tasks like building, deploying, managing, and scheduling.

3. Pricing

- Cloud computing does not have any upfront cost. It is completely based on usage.
- The user is billed based on the amount of resources they use. This helps the user to track their usage and ultimately help to reduce cost.

4. Quality of Service

- Cloud computing must assure the best service level for users.
- Services outlined in the service-level agreements must include guarantees on round-the-clock availability, adequate resources, performance, and bandwidth.

5. Small and medium scale industries

- Small and medium scale industries can also be accommodated but only large companies can take advantage of the availability, versatility, and power of cloud computing.

6. Agility

- Agility improves with users' ability to rapidly and inexpensively re-provision technological infrastructure resources.

7. Cost

- Cost is claimed to be greatly reduced and capital expenditure is converted to operational expenditure.

8. Device and location

- Device and location independence enable users to access systems using a web browser regardless of their location or what device they are using.
- As infrastructure is off-site and accessed via the Internet, users can connect from anywhere.

9. Multi-tenancy

- Multi-tenancy enables sharing of resources and costs across a large pool of users thus allowing for:
    - Centralization of infrastructure in locations with lower costs
    - Peak-load capacity increases
    - Utilization and efficiency improvements for systems that are often only 10–20% utilized.

10. Reliability

- Reliability is improved if multiple redundant sites are used, which makes well designed cloud computing suitable for business continuity and disaster recovery.

11. Scalability

- Via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis near real-time, without users having to engineer for peak loads.

12. Security

- Security is often as good as or better than under traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford.

13. Maintenance

- Cloud computing applications are easier to maintain, since they don't have to be installed on each user's computer.
- They are easier to support and to improve since the changes reach the clients instantly.

14. Metering

- Cloud computing resources usage should be measurable and should be metered per client and application on daily, weekly, monthly, and annual basis.
- This will enable clients on choosing the vendor cloud on cost and reliability (QoS).

2. Functions of Cloud Technical Interface

- Organizations that already have well-designed interfaces between application and infrastructure components may find it easier to transition to the cloud.
- Companies that have moved to a service-oriented architecture (SOA) are well positioned

to make the move.
- With SOA, organizations build modular business services that include standardized interfaces.

There are basically **three important functions of cloud technical interface,**
1. APIs and Data transformation
2. Data and application architecture
3. Security in cloud

APIs and Data transformation
- A cloud's Application Programming Interface (API) is the software interface that lets your company's infrastructure or applications plug in to the cloud.
- This is perhaps the most important place for standardization.
- Many vendors in the cloud space would like to claim overall leadership and control over the interfaces.
- Therefore, many different vendors are developing their own interfaces.
- This, in turn, means that customers are likely to be forced to support multiple APIs.
- Managing multiple APIs means that when applications are changed, there's more programming involved.
- Even if vendors agree to a set of API standards, there will be data transformation issues (as data moves from one physical machine to another).

Data and Application architecture
- New internally created services that support the changing business's changing demands must operate with cloud ecosystems.
- These services may need to migrate to and from the cloud.
- This means that it will have to build an architecture that's modular enough to allow services to move between various cloud platforms.
- The consistency and flexibility of an SOA approach makes it a good fit for the cloud.
- In an SOA environment, software components are put into services or containers. These containers hold software that executes a specific task.
- To be effective in a cloud environment, data also has to be packaged and managed.
- The IT organization needs to manage data independently of the underlying packaged application, transactional system, or data environment such as a warehouse.
- Important data needs to easily move between internal data centers and external cloud-based environments.
- Your organization needs to start with consistent definitions of data elements to manage cloud-based information services.

Security
- Security in the cloud has been designed to assure tight, well defined security services.
- Many levels of security are required within a cloud environment as follows:

- **Identity management:** For example, so that any application service or even hardware component can be authorized on a personal or group role basis.
- **Access control:** There also needs to be the right level of access control within the cloud environment to protect the security of resources.
- **Authorization and authentication:** There must be a mechanism so the right people can change applications and data.

3. Cloud Computing Architecture

- Cloud computing architecture refers to the components and subcomponents required for cloud computing.
- These components typically consist of a front end platform (fat client, thin client, mobile device), back end platforms (servers, storage), a cloud based delivery, and a network (Internet, Intranet, Intercloud).
- Combined, these components make up cloud computing architecture.

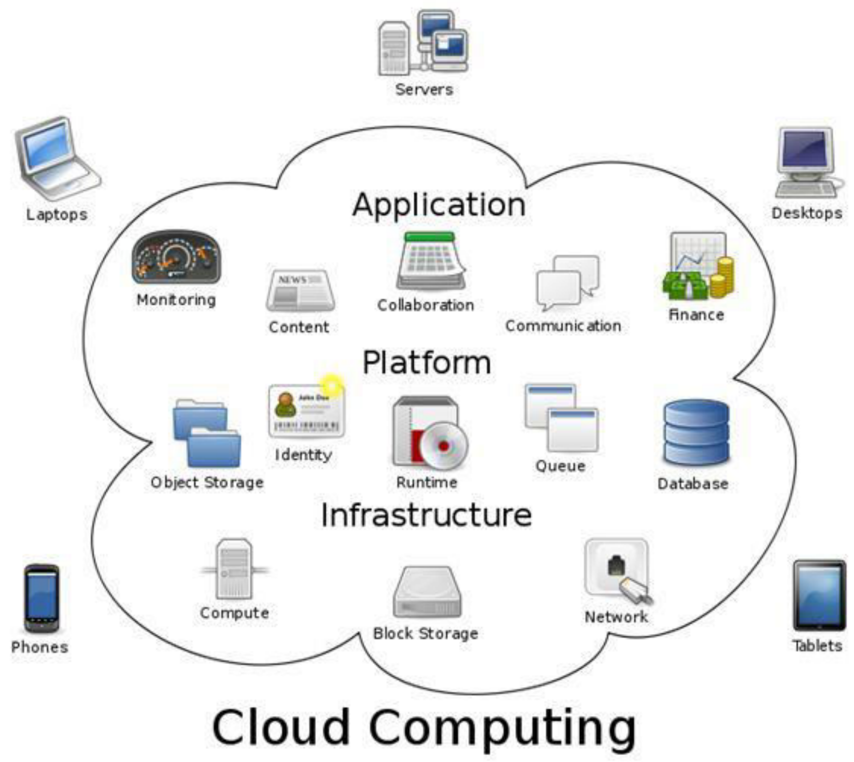


Figure 7.1 Cloud Computing Architecture

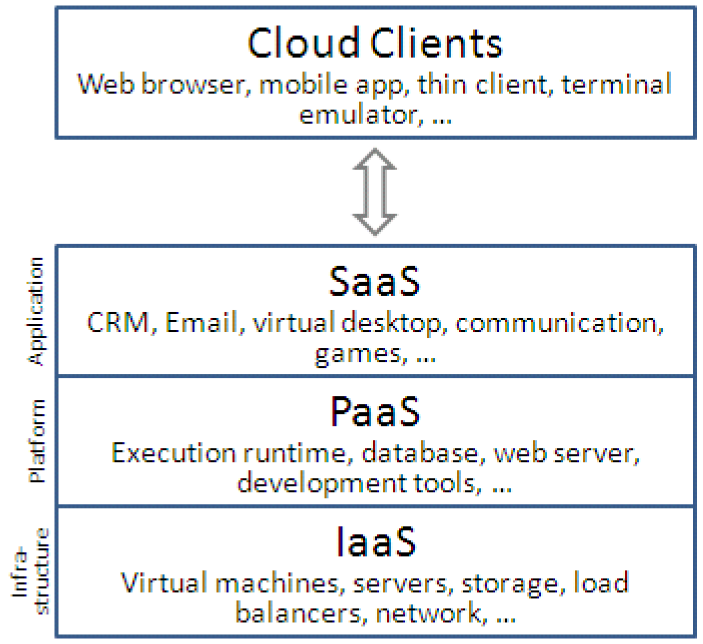

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure

Figure 7.2 Layers of Cloud Computing Architecture

- Cloud computing architectures consist of front-end platforms called clients or cloud clients.
- These clients comprise servers, fat (or thick) clients, thin clients, zero clients, tablets and mobile devices.
- These client platforms interact with the cloud data storage via an application (middleware), via a web browser, or through a virtual session.

Cloud models:

- Cloud computing basically provides three service models as follows:

1. IaaS (Infrastructure as a Service)

- Infrastructure as a Service (IaaS) is the delivery of computer hardware (servers, networking technology, storage, and data center space) as a service.

2. Paas (Platform as a Service)

- With Platform as a Service (PaaS), the provider delivers more than infrastructure.
- It delivers what you might call a solution stack — an integrated set of software that provides everything a developer needs to build an application.

3. Paas (Platform as a Service)

- The SaaS application needs to be generalized enough. So that lots of customers will be interested in the service.

Generally, the cloud network layer should offer:

- High bandwidth (low latency)
    - Allowing users to have uninterrupted access to their data and applications.
- Agile network
    - On-demand access to resources requires the ability to move quickly and efficiently between servers and possibly even clouds.
- Network security

4. Cloud Service Models

- Cloud computing basically provides **three service models** as follows:
    1. IaaS (Infrastructure as a Service)
    2. Paas (Platform as a Service)
    3. Paas (Platform as a Service)

All of the cloud service models can be described in brief as follows:

4.1 IaaS (Infrastructure as a Service) in Cloud

- Infrastructure as a Service (IaaS) is the delivery of computer hardware (servers, networking technology, storage, and data center space) as a service.
- It may also include the delivery of operating systems and virtualization technology to manage the resources.

Characteristics of IaaS:

- The IaaS customer **rents computing resources instead of buying and installing** them in their own data center.
- The service is typically paid for on a usage basis.
- The service may include dynamic scaling, so that if the customer winds up needing more resources than expected, he can get them immediately).
- Also the infrastructure can be automatically scaled up or down, based on the requirements of the application.
- The arrangement involves an agreed-upon service level.
- The service level states what the provider has agreed to deliver in terms of availability and response to demand.
- Example:
- Specify that the resources will be available 99.999 percent of the time and that more resources will be provided dynamically if greater than 80 percent of any given resource is being used.

Example: Amazon EC2

- The most famous example of IaaS operation is Amazon's Elastic Compute Cloud (Amazon EC2).
- It provides a Web interface that allows customers to access virtual machines.

- EC2 offers scalability under the user's control with the user paying for resources by the hour.
- The use of the term elastic in the naming of Amazon's EC2 is significant.
- The elasticity refers to the ability that EC2 users have to easily increase or decrease the infrastructure resources assigned to meet their needs.
- The user needs to initiate a request, so this service provided isn't dynamically scalable.
- Users of EC2 can request the use of any operating system as long as the developer does all the work.
- Amazon itself supports a more limited number of operating systems (Linux, Solaris, and Windows).
- It's easy to join Amazon EC2 through http://aws.amazon.com/ec2.

Other applications of Iaas:

- Companies with research-intensive projects are a natural fit for IaaS.
- It allows scientific and medical researchers to perform testing and analysis at levels that aren't possible without additional access to computing infrastructure.
- Other organizations with similar needs for additional computing resources may boost their own data centers by renting the computer hardware.
- Instead of laying out the capital expenditure for the maximum amount of resources to cover their highest level of demand.
- Means they purchase computing power when they need it.

4.2 PaaS (Platform as a Service) in Cloud

- With Platform as a Service (PaaS), the provider delivers more than infrastructure.
- It delivers what you might call a solution stack — an integrated set of software that provides everything a developer needs to build an application.
- PaaS can be viewed as an evolution of Web hosting.
- In recent years, Web-hosting companies have provided fairly complete software stacks for developing Web sites.
- PaaS takes this idea by providing lifecycle management.
- It also provides capabilities to manage all software development stages from planning and design, to building and deployment, to testing and maintenance.

Benefits/Features of Paas:

- The primary benefit of PaaS is having software development and deployment capability based entirely in the cloud.
- Every aspect of software development, from the design stage onward (including source-code management, testing, and deployment) lives in the cloud.
- PaaS is inherently multi-tenant and naturally supports the whole set of Web services standards and is usually delivered with dynamic scaling.
- In reference to Platform as a Service, dynamic scaling means that the software can be automatically scaled up or down.

- Platform as a Service typically addresses the need to scale as well as the need to separate concerns of access and data security for its customers.

Drawback of Paas:

- The major drawback of Platform as a Service is that it may lock you into the use of a particular development environment and stack of software components.

Examples of PaaS:

- Some examples of Platform as a Service include the
    - Google App Engine
    - AppJet
    - Etelos
    - Qrimp
    - Force.com, which is the official development environment for Salesforce.com.

4.3 SaaS (Software as a Service) in Cloud

Characteristics have to be in place for an SaaS to be commercially viable:

- The SaaS application needs to be generalized enough. So that lots of customers will be interested in the service.
- It needs sophisticated navigation and ease of use.
- It needs to be modular and service oriented.
- It needs to include measuring and monitoring so customers can be charged actual usage.
- It must have a built-in billing service.
- It needs published interfaces and an ecosystem of partners who can expand the company's customer base and market reach.
- It has to ensure that each customer's data and specialized configurations are separate and secure from other customers' data and configurations.
- It needs to provide sophisticated business process configurators for customers.
- It should provide fast releases of new features and new capabilities.
- It has to protect the integrity of customer data.

Comparision: Working of SaaS model Vs. Traditional software providing system

- SaaS or Software as a Service can offer a variety of changes to traditional model that may make it a better or more economical solution for many customers.
- Companies rent licenses or access to the software from a Service Provider or Managed Service Provider (MSP), that actually runs on servers or devices owned and maintained by the service provider.
- SaaS offerings are usually run from large datacenter facilities with redundant power, battery backup, climate controls, etc. that are often out of reach of small and midsized companies.

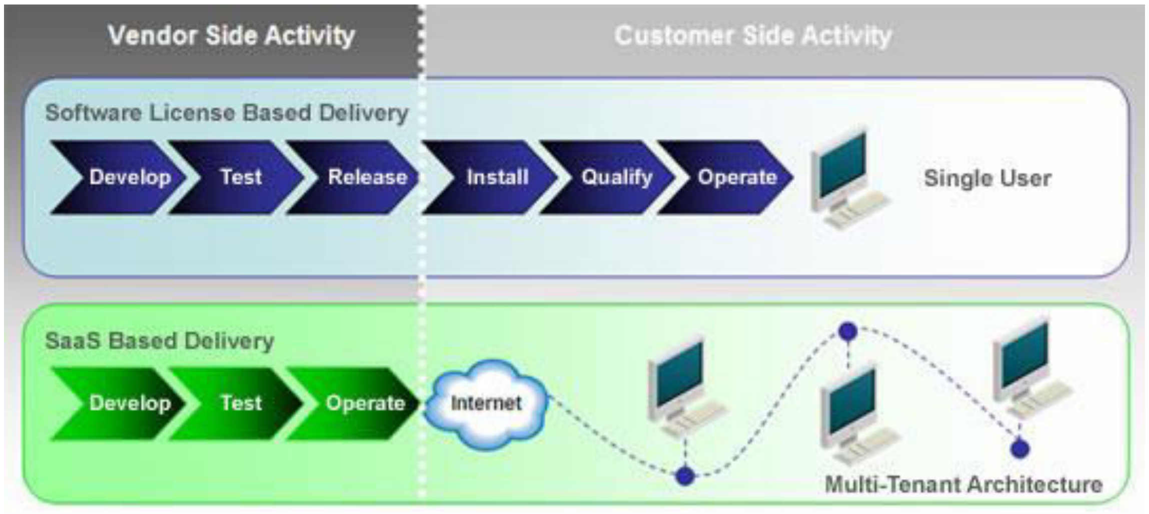


Figure 7.2 Traditional software system Vs. Saas

- Users may need to install something on their PC to allow the MSP to manage the software
- The Managed Service Provider is generally responsible for maintaining and updating the software, and often includes some level of support for users in the monthly (Quarterly or Annual) fee.

Example:

- If you buy a traditional software product, it will cost you a one-time fee of $100,000.
- Now you have to add an annual fee of 20 percent for maintenance and support.
- If you look at the costs over five years, for example, you may determine the following: Software will cost $100,000; maintenance expenses will add another $100,000 over five years, for a total five-year cost of $200,000.

5. Microsoft Windows Azure

- Microsoft defines the Azure platform as “An Internet-scale cloud services platform hosted in Microsoft data centers, which provides an operating system and a set of developer services that can be used individually or together.”
- It provides both platform as a service (PaaS) and infrastructure as a service (IaaS) services and supports many different programming languages, tools and frameworks.
- It works with both, Microsoft-specific and third-party software and systems.
- Microsoft’s overall strategy is a combination of on-premise computing with cloud-based services.
- The idea is that developers want to build some applications that live on-site, while other components will live in the cloud.
- Microsoft also intends to support other programming models then ASP .net, including Ruby on Rails and Python etc. with it.
- For interoperability, Microsoft supports various Internet protocols, including HTTP, REST,

SOAP, and XML.

Basic Components

- Windows Azure Environment: This Windows environment runs applications locally and stores the related data on servers inside the Microsoft data center.
- Microsoft .NET Services: These are the same .Net services that Microsoft has in its Windows environment. It has been extended to support cloud-based as well as on-premise applications.
- Microsoft SQL Services: If you want to take advantage of Azure's scaling capability, you must rewrite the SQL code.
- Live Services: This set of services allows developers to connect code developed in the Windows Live platform into the cloud.

Key Features

- Websites allows developers to build sites using ASP.NET, PHP, or Node.js and can be deployed using FTP, Git, or Team Foundation Server.
- VM(Virtual machines) let developers migrate applications and infrastructure without changing existing code, and can run both Windows and Linux virtual machines.
- Its VMs comprise the Infrastructure as a Service (IaaS) offering from Microsoft for their public cloud.
- Customers can create VM, of which they have complete control, to run the Microsoft Data Centers.
- Customers can create Virtual Machines, of which they have complete control, to run the Microsoft Data Centers.
- Cloud services - Microsoft's Platform as a Service (PaaS) environment that is used to create scalable applications and services. Supports multi-tier scenarios and automated deployments.
- Data management - SQL Database, formerly known as SQL Azure Database, works to create, scale and extend applications into the cloud using Microsoft SQL Server technology.
- Media services - A PaaS offering that can be used for encoding, content protection, streaming, and/or analytics.

Key Services

- High density hosting of web sites.
- Customers can create Virtual Machines, of which they have complete control, to run the Microsoft Data Centers.
- The Cloud Services for Windows Azure comprises one aspect of the PaaS offerings from the Windows Azure Platform. The Cloud Services are containers of hosted applications.
- Data management
- SQL database, tables and BLOB storage
- Business analytics

- SQL reporting, data market place, and Hadoop
- Active directory and access control service
- Messaging
- Media and mobile services
- For this a PaaS offering that can be used for encoding, content protection, streaming, and/or analytics.

6. Salesforce.com

- Salesforce.com (stylized as salesforce) a cloud computing company headquartered inSan Francisco, California.
- Its revenue comes from a customer relationship management (CRM) product.
- Salesforce also tries capitalizing on commercial applications of social networking through acquisition.
- Salesforce.com built and delivered a sales force automation application (which automates sales functions such as tracking sales leads and prospects and forecasting sales) that was suitable for the typical salesperson and built a business around making that application available over the Internet through a browser.
- The company then expanded by encouraging the growth of a software ecosystem around its extended set of customer relationship management (CRM) applications.
- It began, for example, by allowing customers to change tabs and create their own database objects.
- The company also added what it called the AppExchange, which added published application programming interfaces (APIs) so that third-party software providers could integrate their applications into the Salesforce.com platform.
- Most AppExchange applications are more like utilities than full-fledged packaged apps. Many of the packages sold through the AppExchange are for tracking. For example, one tracks information about commercial and residential properties, another optimizes the sales process for media/advertising companies; still another package analyzes sales data.
- Salesforce.com took its offerings a step further by offering its own language called Apex. Apex is used only within the Salesforce.com platform and lets users build business applications and manage data and processes.
- A developer can use Apex to change the way the application looks.
- It is, in essence, the interface as a service. With the advent of cloud computing, Salesforce.com has packaged these offerings into what it calls Force.com, which provides a set of common services its partners and customers can use to integrate into their own applications.
- Salesforce.com has thus started to also become a Platform as a Service vendor.
- Among the hundreds of applications that run on Force.com, it now offers a variety of HR software, and financial, supply chain, inventory, and risk management components.
- Just as Amazon is currently the trailblazer among the Infrastructure as Service vendors, Salesforce.com is the trailblazer among the Software as a Service vendor.

7. Google App Engine

- When you visit the Google App Engine Web site at https://cloud.google.com/appengine/docs you will notice following phrases:
  1. **Introduction of Google App Engine**
  2. **Features**
  3. **Pricing and Quotas**
  4. **Downloads link for Google SDK**

- The developer provides the Google App Engine with a URL (Web address) for the application it's building and the engine maps that code to Google's development platform.
- The App Engine handles the Web application lifecycle, including routine tasks such as request logs, checking the application status, updating the application version, operating the underlying database, and handling workflow.
- Google has integrated all the development tools into a single integrated environment.
- When customers tie their development into the lifecycle environment provided by Google, they also gain access to Google's IaaS. In this way, customers can add more capacity on demand.

Google also provides **other infrastructure services** as follows:

- Google Accounts for authentication
- Google native file system called GFS (Google File System)
- BigTable platform (for data management), a distributed storage system that manages very large-scale structured data

Google development stack

- In addition to these infrastructure tools, Google App Engine also includes a development stack.
- Google calls this a scalable serving infrastructure that connects the Web application code to the Google environment.

It does this **by integrating with the following tools:**

- **Python runtime:** To create an application for the platform requires a programming language.
- The first one that Google supported was Python, a high-level programming language that makes it easier to rapidly build complex applications with minimal programming.
- Python includes models and packages and supports code reuse.
- **Java runtime:** Google added Java as a second supported programming language platform.
- This runtime is integrated with Google's toolkits and is intended to be used for AJAX (asynchronous JavaScript and XML) or interactive Web applications.
- **A Software Development Kit (SDK):** This set of development tools enables developers to write application code.
- **A Web-based administration console:** The console helps developers manage their applications.
- **Datastore:** A datastore is a software layer that stores a Web application's data. It is built on the Bigtable (a high-performance database) structure.

1. Types of Cloud

- Cloud Computing can broadly be classified into 4 types as follows,
  1. **Public Cloud**
  2. **Private Cloud**
  3. **Hybrid Cloud**
  4. **Community Cloud**

- The types of cloud mentioned above are based on,
  - The type of application and environment being considered before developing the application and
  - The base of the location is where it is been hosted.

1.1 Public Cloud

- The name **"public"** in the public cloud comes from the fact that application is hosted on the Hosting Providers location (Vendors).
- Though it is hosted in shared system, each resource operates with encrypted securely.
- With Public Cloud all the resources and the services are dynamically added and removed (Scalable) based on the usage.
- Public cloud has more advantages for Small and Medium scale industries.
- We are going to pay for the resources which we are going to use and specifically the hardware and the bandwidth are going to be maintained by the hosting provider.
- Some of the examples of Public Cloud in market are Amazon Web Services, Microsoft Azure and Google Apps.

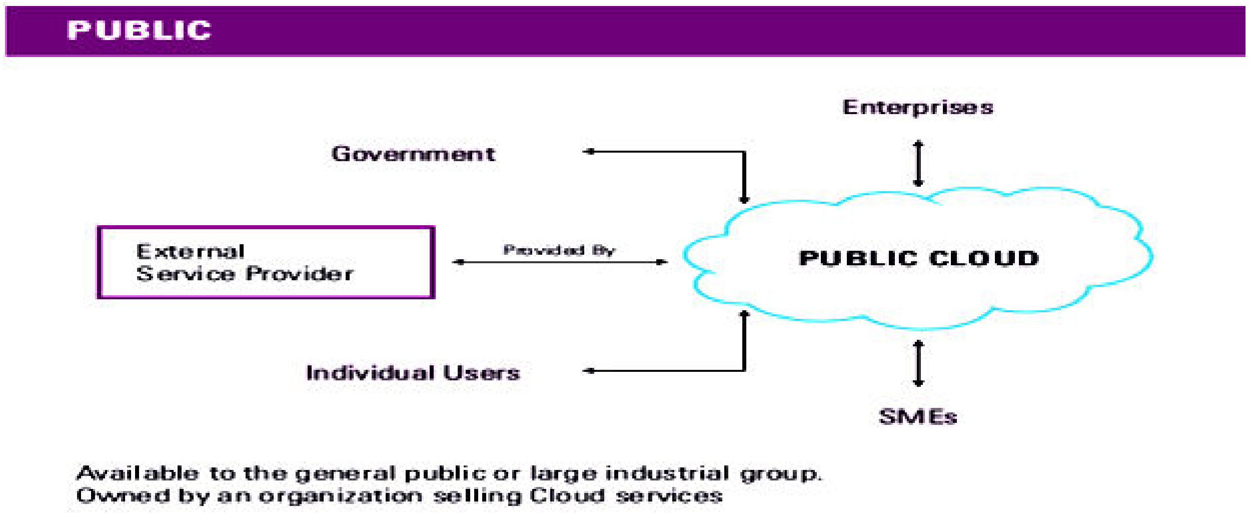

PUBLIC
Enterprises
Government
External Service Provider
Provided By
PUBLIC CLOUD
Individual Users
SMEs
Available to the general public or large industrial group.
Owned by an organization selling Cloud services

Figure 8.1 Public cloud

Following are the situation in which public clouds are preferable.

- Your standardized workload for applications is used by lots of people.

- Email is an excellent example.
- You need to test and develop application code.
- You have SaaS (Software as a Service) applications from a vendor who has a well-implemented security strategy.
- You need incremental capacity (to add compute capacity for peak times).
- You're doing collaboration projects.
- You're doing an ad-hoc software development project using a Platform as a Service (PaaS) offering.

1.2 Private Cloud

- In this form, the cloud is deployed with in a corporate firewall and runs on premise IT infrastructure.
- Private cloud is **more expensive** compared to the public cloud since the operating and bandwidth costs are to be maintained by the organization, but this cloud is **more secure** than the public cloud.
- Private Cloud provides more benefits to the corporate by capitalizing on the Data Security and Corporate Governance.
- It provides more control to administrators over the operating environment.

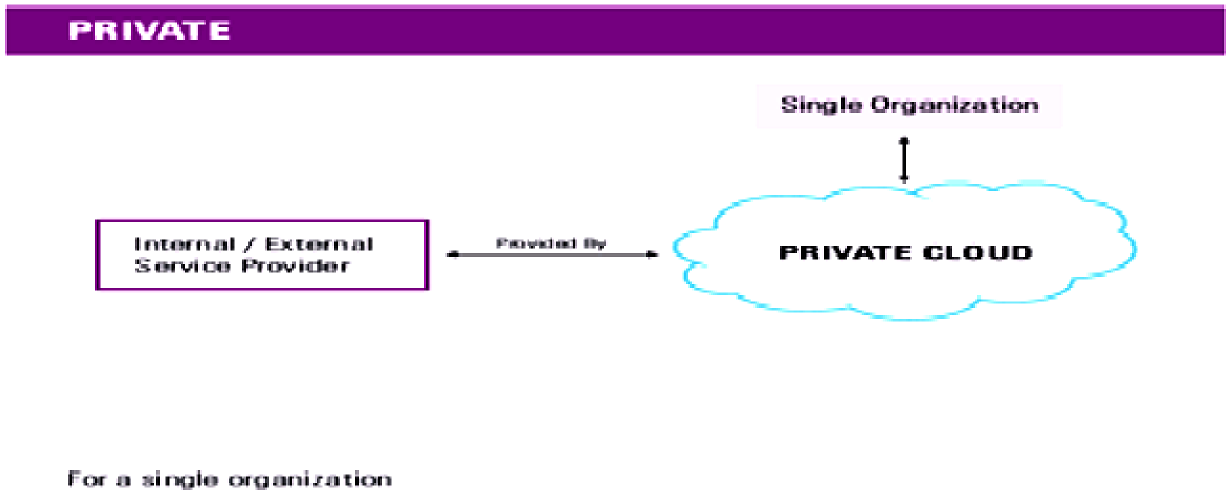

PRIVATE
Single Organization
Internal / External Service Provider
Provided By
PRIVATE CLOUD
For a single organization

Figure 8.2 Public cloud

Characteristics of private cloud:

- It Allows IT Company to provision services and compute capability to internal users in a self-service manner.
- Automates management tasks and lets you bill business units for the services they consume
- Provides a well-managed environment
- Optimizes the use of computing resources such as servers

- Supports specific workloads
- Provides self-service based provisioning of hardware and software resources

1.3 Hybrid Cloud

- Hybrid Cloud are increasingly being used by corporations where there is a need to maintain some of their applications on their internal infrastructure.
- It also needs the scalability and cost benefits of the public cloud.
- Cloud bursting is the term normally used for this type of environment where internal applications are deployed on private cloud for normal usage while internet applications are deployed on the public cloud to handle variable loads.

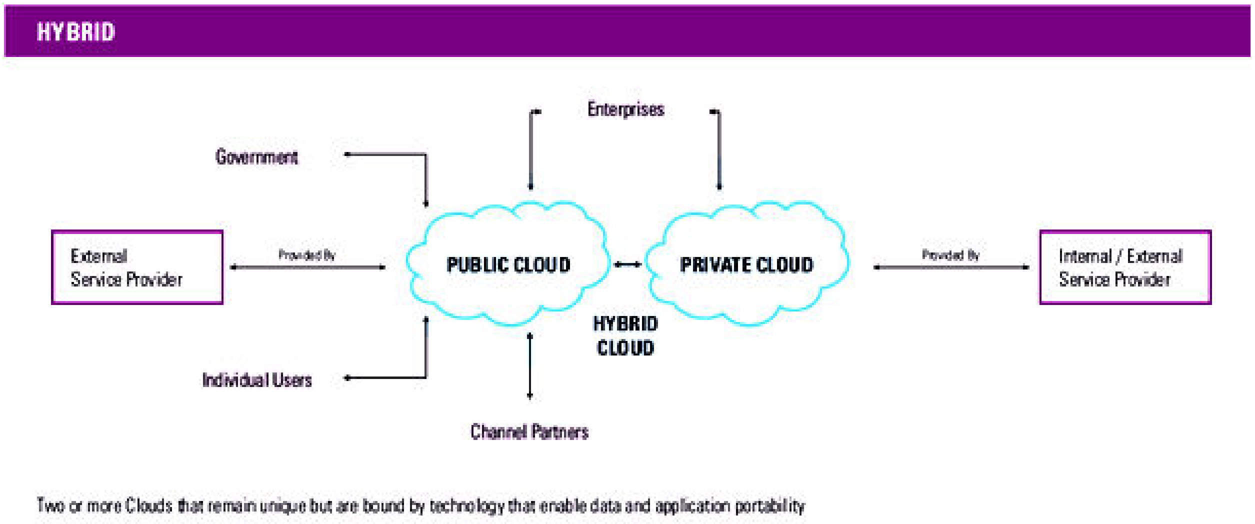

HYBRID
Enterprises
Government
External Service Provider
Provided By
PUBLIC CLOUD
PRIVATE CLOUD
Provided By
Internal / External Service Provider
HYBRID CLOUD
Individual Users
Channel Partners
Two or more Clouds that remain unique but are bound by technology that enable data and application portability

Figure 8.3 Public cloud

1.4 Community Cloud

- This type of cloud is specifically used for organizations that are shared such as different government organizations.
- In this case, non-government organizations will not have access to this cloud. The cloud could be located in-house, or on public cloud based on the needs of the organization.

2. Technical foundation for scaling computer systems in cloud

- There are basically four areas where IT system spends money.
- Those areas can be listed as Hardware, A power supply, Networking and communications equipment, and Electricity.
- For scaling up proper system following areas needs to be considered.

2.1 Server-ing up some hardware

- Costs for your data center hardware will vary dramatically depending on the type of workloads you support.

2.1.1 Traditional system Vs Cloud

- **Traditional:** In a traditional data center, IT management has a structured process for purchasing hardware.
- Each year they talk to business units, determine what new applications to add or expand, and work with vendors on the procurement process.
- In addition, most IT organizations refresh their hardware on a regular basis to make sure that things run smoothly and old systems are retired before they cause problems.
- **Cloud:** When a business is creating a cloud data center (either a private one inside the firewall or a service provider) the process of procuring systems is very different.
- Because the cloud supports very different workloads, IT management doesn't buy traditional hardware.
- IT management might go directly to an engineering company that designs the system boards and networking switches for them, and then take the contract to a manufacturer to have them build the precise hardware that they want.

2.1.2 Consideration of Cloud hardware

- While setting up the cloud data center following concepts need needs to be focused.
    - Cooling
    - CPU, memory and local disk
    - Data storage and networking
    - Redundancy checking
    - Software embedded within the data center

2.1.3 Open-source dynamic

- The cloud is an economic and business model as much as a technology model.
- It isn't surprising, then, that open-source software is an important element for almost all cloud providers.
- Some of it is very high quality and nearly all of it can be used for no license fee, as long as you obey the restrictions of the associated license.
- Open-source software has already become a business factor in the Internet service provider (ISP) business, with most ISPs providing an easily installed, highly functional software stack for building Web sites.
- Many cloud providers take open-source software as a foundation and customize it to optimize support for their workloads.

2.1.4 Economics of scale

- Cloud gives scale of economics on following bases
    - Better communications prices:
    - Predictable network traffic:
    - Network virtualization:
    - Backup and disaster recovery
    - System management

- Security
- Client caching

3. Introduction of Traditional and Cloud data centers

3.1 Traditional data center

- The reality of the traditional data center is further complicated because most of the costs maintain existing (and sometimes aging) applications and infrastructure.

The nature of the applications and the workloads at the core of data centers:

- Most data centers run a lot of different applications and have a wide variety of workloads.
- Many of the most important applications running in data centers are actually used by only a relatively few employees.
- For example, transaction management applications which are confidential might only be used by a few employees.
- Some applications that run on older systems are taken off the market but are still necessary for business.

Note: Because of the nature of above applications, it probably wouldn't be cost effective to move these environments to the cloud.

3.2 Cloud data center

- **Cloud data centers** means data centers with thousands or more servers on site.
- All data centers are devoted to running very few applications that are built with consistent infrastructure components.
- Infrastructure components are such as racks, hardware, OS, networking, and so on.
- One of the most important differences with traditional data center is that cloud data centers aren't remodeled traditional data centers.

Basic characteristics of cloud data centers are,

- Constructed for a different purpose.
- Created at a different time than the traditional data center.
- Built to a different scale.
- Not constrained by the same limitations.
- Perform different workloads than traditional data centers.

Note: Because of above characteristics, the economics of a cloud data center are significantly different.

4. Traditional Vs. Cloud data center

- Before making any decisions about moving your data center to a cloud, you need to take the time to consider how cloud data centers compare to traditional data centers.
- Because of the high costs of infrastructure and administration, data centers are often one of the first business areas that companies consider switching to a cloud service.

Traditional Corporate Data Center	**Cloud Data Center**
Non-Technological Costs	
Thousands of different applications	Few applications (maybe even just one)
Mixed hardware environment	Homogeneous hardware environment
Multiple management tools	Standardized management tools
Frequent application patching and updating	Minimal application patching and updating
Complex workloads	Simple workloads
Multiple software architectures	Single standard software architecture
Due to complexity and variety of workloads, more labor force is required.	Labor cost is lower since workloads are few.
These costs can be much greater when co pared to clouds.	Lower electricity, location and taxation costs due to mass consumption
Technological Costs	
• IT management has a structured process for purchasing hardware. • Various applications and hardware are added according to the organization's needs.	• The process of procuring systems in a cloud data center is very different. • Because the cloud supports very different workloads, IT management doesn't buy traditional hardware. • Rather, IT management contracts to a manufacturer to have them build the precise hardware that they want.
• Air Cooling is generally used in traditional data centers which is less efficient.	• Due to the huge cooling requirements, cooling by water is affordable which is more efficient.
• Traditional data tends to be filled with a lot of surplus equipment (CPU, memory & local disk) • Surplus memory, CPUs, and disks take up valuable space and they need to be cooled too.	• The cloud data center typically supports self-service provisioning of resources so capacity is added only when needed.

• Data centers must always move data around the network for backup and disaster recovery. • Traditional data centers support so many different workloads that many approaches to backup and recovery have to be taken. • This makes backing up and recovering data complicated and expensive.	• The cloud is designed to handle data workloads consistently. • For example, in a cloud data center you can establish a global policy about how and when backups will be handled. • This can be then handled in an automated manner, reducing the cost of handling backup and recovery.
• Software linked at system level is a big cost in the traditional data center simply because there are so many more workloads with so many operating systems and related software elements.	• Cloud data centers have fewer elements because they have simpler workloads. • Cloud providers understand these costs well and design their offerings to maximize revenue.

5. Cloud Workload

- The cloud requires that workloads have to be handled in a very abstracted manner.
- The abstraction is a way to keep the technical details away from the user.
- The result of this abstraction is a type of service that makes it easier to have a well-defined function with a defined purpose.
- This service lives inside a container with an Application Programming Interface (API) so it can be easily moved from one place to another.
- If you're familiar with a service-oriented architecture, you probably recognize that this might sound a lot like a business service.
- A business service is a function or process designed to include well-defined Web services interfaces.
- Therefore, this type of service is designed for many different situations, which is an important concept for the cloud.

Different workload types

- There are basically **two types** of workloads exist:
  1. Workloads that **can be executed at any time in batch mode**
  2. Workloads that **need to be executed in real time**

- For example, an insurance company is likely to have a workload that calculates interest rate. This doesn't have to happen immediately.
- In contrast, an online retail system that calculates taxes on a purchase needs to be executed in real time.
- Many business information systems that help management understand the status of their business are batch workloads.
- A credit card payment system is a real-time workload. You might have a single workload that's an entire application used by a group of customers.
- In other situations, a smaller service may be used in many different contexts. There might

be a workload that's a payment service platform.

- This payment service might be live in a cloud and may be used by many different software developers who all need a payment engine.
- Many Platform as a Service vendors offer workloads or services like them to their partners.

Characteristics of cloud data centers workload:

- A workload has no dependencies.
- The workload interface must be consistent.
- A workload may have rules or policies that apply in specific situations.
- There may be authorization and security policies associated with using a service for a particular function.
- There may be a rule about when to use a specific workload. For example, a workload such as an accounting process might need to be executed at the end of a specific cycle.
- Therefore, although a workload can be thought of as a container or service, it will be used in conjunction with both simple and very complex processes.

6. Security and Privacy of Data in Cloud

- There are three key areas of concern related to security and privacy of data:
  1. Data location in the cloud
  2. Data control in the cloud
  3. Securing data for transport in cloud

Data location in the cloud

- After data goes into the cloud, you may not have control over where it's stored geographically. Consider these issues:
- **Specific country laws:** Laws governing data differ across geographic boundaries. Your own country's legal protections may not apply if your data is located outside of the country. A foreign government may be able to access your data or keep you from fully controlling your data when you need it.
- **Data transfer across country borders:** A global company with subsidiaries or partners (or clients for that matter) in other countries may be concerned about cross-border transfer of data due to local laws. Virtualization makes this an especially tough problem because the cloud provider might not know where the data is at any particular moment.
- **Co-mingling of data:** Even if your data is in a country that has laws you're comfortable with, your data may be physically stored in a database along with data from other companies. This raises concerns about virus attacks or hackers trying to get at another company's data.
- **Secondary data use:** In public cloud situations, your data or metadata may be vulnerable to alternative or secondary uses by the cloud service provider.
- Without proper controls or service level agreements, your data may be used for marketing purposes (and merged with data from other organizations for these alternative

uses).

- The recent uproar about Facebook mining data from its network is an example.

Data control in the cloud

- Controls include the governance policies set in place to make sure that your data can be trusted.
- The integrity, reliability, and confidentiality of your data must be beyond reproach. And this holds for cloud providers too.
- You must understand what level of controls will be maintained by your cloud provider and consider how these controls can be audited.

- Here is a sampling of the different types of controls designed to ensure the completeness and accuracy of data input, output, and processing:
- Input validation controls to ensure that all data input to any system or application are complete, accurate, and reasonable.
- Processing controls to ensure that data are processed completely and accurately in an application.
- File controls to make sure that data are manipulated accurately in any type of file (structured and unstructured).
- Output reconciliation controls to ensure that data can be reconciled from input to output.
- Access controls to ensure that only those who are authorized to access the data can do so. Sensitive data must also be protected in storage and transfer. Encrypting the data can help to do this.
- Change management controls to ensure that data can't be changed without proper authorization.
- Backup and recovery controls. Many security breaches come from problems in data backup.

Securing data for transport in cloud

- Regarding data transport, keep two things in mind:
  1. Make sure that no one can intercept your data as it moves from point A to point B in the cloud.
  2. Make sure that no data leaks (malicious or otherwise) from any storage in the cloud.

- None of these concepts are new; the goal of securely transporting data has been around as long as the Internet.
- In the cloud, the journey from point A to point B might take on three different forms:
- Within a cloud environment
- Over the public Internet between an enterprise and a cloud provider
- Between clouds
- The security process may include segregating your data from other companies' data and then encrypting it by using an approved method. In addition, you may want to ensure the security of older data that remains with a cloud vendor after you no longer need it.

VPN for security

- A virtual private network (VPN) is one way to manage the security of data during its transport in a cloud environment.
- A VPN essentially makes the public network your own private network instead of using dedicated connectivity.

  A well-designed VPN needs to incorporate **two things:**

  1. A firewall to act as a barrier to between the public Internet and any private network (like at your enterprise).
  2. Encryption to protect your sensitive data from hackers; only the computer that you send it to should have the key to decode the data.