05/09/2024

UNIT-1

→ Scalable Computing over the Internet:

* Scalability is the ability of a computer product or an application to continue to function well in order to meet user requirements.
* Changes are seen in: OS, network connection, machine architecture, application workload.

* **High Performance Computing (HPC)**
  - → super computers & centers.
  - → No longer optimal for measuring system performance.
  - → parallel processors.
  - → replaced by clusters
  - → Emphasize raw speed performance.

High Throughput Computing (HTC)

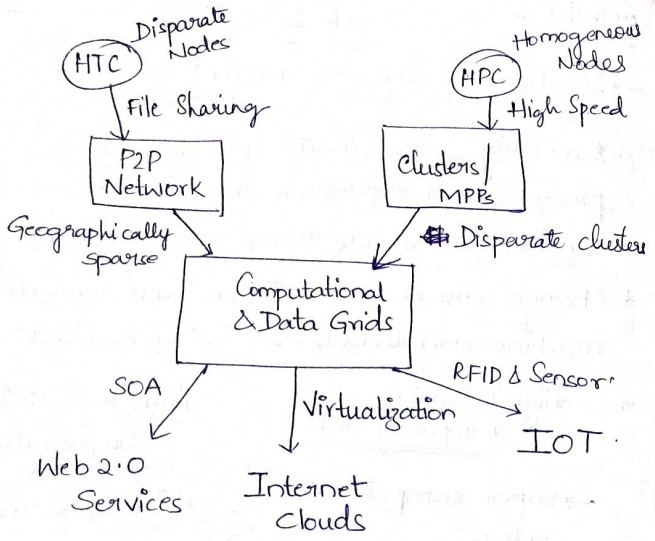
→ cloud computing

→ based on parallel & distributed technologies.

→ Peer to peer networks

→ distributed file sharing & content delivery.

→ improve processing speed, energy savings, security & reliability



* Computer Clusters: set of computers that work together (can be viewed as a single system)
→ Each node is set to perform the same task, controlled & scheduled by software.
→ Connected to each other through fast local area networks.

* P2P Networks: Simple network of computers.
→ Each computer acts as a node for file sharing & also acts as a server.
→ Allows sharing of a huge amount of data.
→ Tasks are equally divided amongst nodes.

*→ Computing Paradigms:

(1) Service Oriented Architecture:
- → services are provided to other components by application components.
- → a self-contained unit of software designed to complete a specific task.

(2) Internet Cloud:
- → Advances in virtualization

(3) IOT:
- → RFID, GPS, sensors: triggered development of IoT
- → Supported by internet clouds to achieve ubiquitous computing with any Object

*→ Computing Paradigm Distinctions:

(1) Centralized Computing:
- → computer resources are centralized in one physical system.
- → resources are fully shared & tightly coupled within one OS.

② Parallel Computing:
- Either coupled with centralized shared memory or loosely coupled with distributed memory.
- Inter process communication through shared memory or via message passing.

③ Distributed Computing:
- Distributed system consists of multiple autonomous computers, having private memory.
- Info exchange is through message passing.

④ Cloud Computing:
- Resources can be either a centralized or distributed computing system.
- Clouds: applies parallel, distributed or both.
- Clouds are built with physical or virtualized resources over large data centers.

⑤ Concurrent Computing: union of parallel & distributing computing.

⑥ Ubiquitous Computing: Computing with pervasive devices at any place & time using wired | wireless communication.

⑦ Internet computing: covers all paradigms.

* System efficiency is decided by speed, programming & energy factors.

Efficiency:
- HPC: utilization rate of resources.
- HTC: job throughput, data access, storage & power efficiency.

Dependability: measures the reliability & self management from the chip to the system & application levels.

Adaptation: ability to support billions of job requests over massive data sets.

Flexibility: ability of distributed systems to run well in both HPC & HTC.

* Degrees of Parallelism:

① bit-level parallelism

② Instruction level parallelism
③ Data-level parallelism
④ Task - level parallelism

* Utility computing focusses on a business model in which customers receive computing resources from a paid service provider.

* Hype Cycle of New Technologies:

① Hollow circles: techs will reach mainstream adoption in 2 years.

② Gray circles: techs will reach mainstream adoption in 2 to 5 yrs.

③ Solid circles: require 5 to 10 yrs.

④ Triangle: more than 10 yrs.

⑤ crossed circles: obsolete before they reach the plateau.

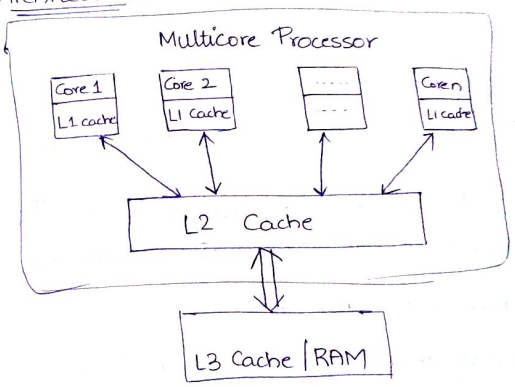
* Cyber Physical Systems: result of interaction b/w computational processes & the physical world.

* Technologies for Network-Based systems

→ Multi-core CPUs:

- Computer processor integrated with 2 (or) more separate processing units called cores.
- Each of these cores reads & executes program instructions.

* → Architecture:



→ Multicore CPUs may use tens of crores to hundreds or more in the future. CPU reached its limit in terms of DLP.

→ This triggered the development of many-core GPUs with hundreds or more thin cores.

* Multi-threading Technology:

→ Threads: basic unit of CPU utilization.

→ It shares its code, data section & other resources like memory with other threads.

→ 5 Independent Threads:

① Superscalar processor: single-threaded with 4 functional units.

② Fine-grain threading: Switches execution of instructions from diff threads per cycle.

③ Course-grain threading: executes many instructions. → Cellular Multi-Processing

④ Multicore CMP: executes instructions from diff threads.

⑤ Simultaneous multithreaded (SMT): allows simultaneous scheduling of instructions.

* GPU Computing:

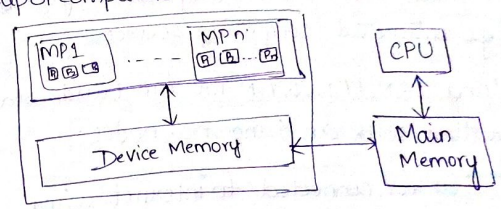
↓

Graphics coprocessor.

→ GPU offloads CPU from tedious graphics.

→ CPUs: only few cores.
GPUs: can be built with hundreds of processing cores.

→ GPUs are used in HPC systems to power super computers.



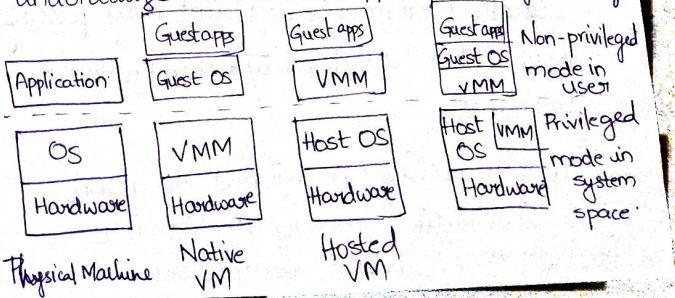
* System Area Interconnects:

① LAN: connect client hosts to big servers.

② SAN: (storage area NW) connects servers to network storage

③ NAS: (Network attached storage) connects client hosts to disk arrays.

* Virtual machines offer novel solutions to underutilized resources, application inflexibility



* System Models for distributed & cloud computing:

→ Computing cluster: consists of interconnected computers which work cooperatively as a single integrated computing source.

* Using SAN, LAN, WAN we can build scalable clusters with an ↑sing no. nodes.

* Cluster is connected to internet using a virtual private network (VPN) gateway.

* Ideal cluster: merge multiple system images into single system image.

→ Cluster design issues:

* without middleware cluster nodes cannot work together efficiently.

* Cluster benefits come from scalable performance, efficient msg passing, high system availability, seamless fault tolerance

→ Grid computing, envisioned to allow close interaction among applications, running on distant computers.

→ Computers used in grids are primarily workstations, servers, clusters & super computers

* P2P network: client oriented.

* P2P overlay network characterizes the logical connectivity among the peers.

- → Unstructured Overlay Network.
  - → No fixed route to send messages or files among nodes.
  - → Flooding is applied to send a query to all nodes.
- → Structured Overlay Network:
  - → follow certain connectivity topology.
  - → Rules for inserting & removing nodes.
  - → Routing mechanisms are developed.

* Challenges Of P2P:

- → too many hardware models & architecture
- → incompatibility blw software & OS.
- → Lack of Security
- → Not centralized.

→ A cloud is a pool of virtualized computer resources.

→ A cloud infrastructure provides a framework to manage scalable, reliable, on-demand access to applications.

→ Cloud computing leverages its low cost & simplicity which benefits both users & providers.

*Internet Clouds:

→ Cloud computing is a virtualized platform with elastic resources on demand by provisioning hardware, software and data sets dynamically.

→ Cloud ecosystem must to be designed to be secure, trustworthy and dependable.

*Cloud Landscape:

① Traditional systems has several performance bottlenecks:
- → constant system maintainence
- → poor utilization
- → increasing costs associated with hardware/software upgrades.

② Cloud computing as an on demand computing paradigm resolves or relieves us from these problems.

* Cloud Service Models:

① Infrastructure as a Service (IaaS): → EC2
- → puts together infrastructures demanded by users.
- → users can deploy & run on multiple VMs running guest OSes
- → users can specify when to request & release the needed resources.

② Platform as a Service (PaaS): → Google App Engine
- → enables the user to deploy user-built applications onto a virtualized cloud platform.
- → both hardware & software are integrated with specific programming interfaces.

③ Software as a Service (SaaS): → Salesforce.com.
- → browser-initiated application software over thousands of paid cloud customers.
- → Customer side: no upfront investment in servers.

provider side: costs are rather low, compared with conventional hosting.

** Internet clouds offer 4 deployment modes: private, public, managed & hybrid.

* Software Environments for distributed systems and clouds:

① Service-Oriented Architecture: (SOA):

→ In grids/web-services: an entity is a service.

→ In CORBA: an entity is a CORBA distributed object.

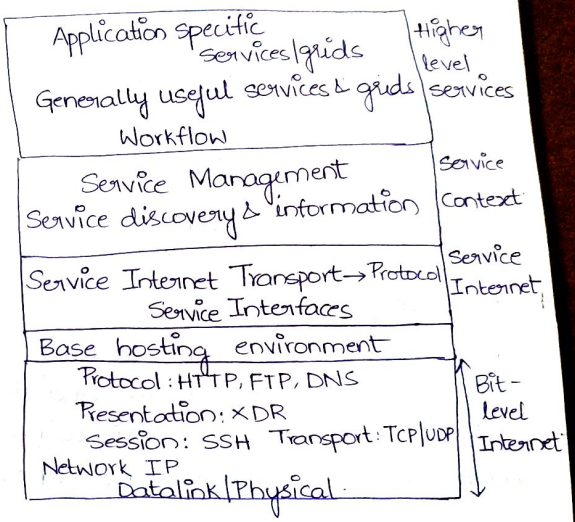
** These architectures are build on traditional seven OSI layers that provide base networking abstractions.

On top of this we have a software environment:

- → .NET or Apache Axis for web services
- → JVM for java.
- → a broker network for CORBA.

On top of this software environment, we have a higher level environment, reflecting the special features of distributed computing environment.

* Layered architecture for web services and grids:



→ The entity interfaces correspond to:

- → Web Services Description language (WSDL) ↓SOAP
- → Java method - RMI
- → CORBA & interface defⁿ language (IDL) - IIOP

CORBA: Common Object Request Broker Architecture.

IIOP: Internet Inter-ORB protocol is used for distributed programs written in diff. programming languages to communicate over the internet.

→ SOAP, RMI, IIOP: high-level communication systems.

* These communication systems supports following features:
  - → message patterns
  - → fault recovery
  - → specialized routing.

* Web Services & Tools:

→ In web services, one aims to fully specify all aspects of service and its environment. This is carried out ~~using~~ with communicated messages using SOAP.

→ REST architectures are clearly more appropriate for rapid technology environments.

* Evolution of SOA:

→ SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds and systems of systems in general.

→ A large no. of sensors provide data-collection services; known as sensor service.

→ Raw data is collected by sensor devices

→ All the SS services interact with large or small computers, many forms of grids, databases, etc.

→ Filter services are used to eliminate unwanted raw data.

→ Most distributed systems require a web interface or portal.

→ Raw data is transformed into useful information by going through a sequence of compute, storage, filter & discovery clouds.

* Grids v/s Clouds:

Grids → applies static resources.

Clouds → emphasizes on elastic resources

* Distributed Operating Systems:

→ computers in most distributed systems are loosely coupled.

→ To promote resource sharing and fast communication among node machines, it is best to have a distributed OS that

manages all resources coherently and efficiently.

* Tanenbaum identifies 3 approaches for distributing resource management functions in a distributed computer system:

① to build a network OS over a large no. of heterogeneous OS platforms.

② to develop middleware to offer a limited degree of resource sharing

③ to develop a truly distributed OS to achieve higher use or system transparency.

* Amoeba V/s DCE:

→ Amoeba:

① distributed OS

② light-weight microkernel approach

DCE

① middleware based system for distributed computing environments

② Open Software foundation pushed the use of DCE.

- MOSIX2: distributed OS, which runs with a virtualization layer in the Linux environment

* Supports both sequential & parallel applications

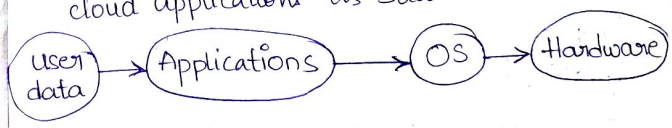
* can manage a LINUX cluster or a grid of multiple clusters.

→ Programming Environments:

* Data is owned by users, independent of applications.

* OS provides clear interfaces, standard programming interfaces (or) system calls.

* To separate user data from specific application programs, users can enable cloud applications as SaaS.



* Parallel & Distributed Programming Models:

① MPI: (Library of subprograms.) Specify synchronous/asynchronous point to point & collective communication commands and I/O operations in user programs.

Message Passing Interface.

② MapReduce: Generates a set of intermediate web model for key-value pairs; reduce-fn scalable data merges all intermediate values processing with same key.

③ Hadoop: Scalable, Economical & Reliable
⇓ tool.
software platform.

* Open Grid Services Architecture (OGSA):

Features:
- → A distributed execution environment.
- → Public Key infrastructure.
- → Trust management.
- → security policies in grid computing.

* Introduction to Cloud Computing:

→ Computing to be considered fully virtualized.
  - → must allow computers to be built from distributed components; such as processing, storage, data and software resources.

→ Technologies such as cluster, grid, cloud all are aimed at access to large amounts of computing power; fully virtualized manner by aggregating resources and offering a single system view.

* Utility Computing:
  - → a business model for on-demand delivery of computing power.
  - → pay-as-you-go.

* Characteristics of cloud:

① pay per use.
② elastic capacity and the illusion of infinite resources.
③ self-service interface.
④ resources that are abstracted (or) virtualised.

* Roots of Cloud computing:

① Hardware: virtualization, multi-core chips
② Internet Technologies: web services, SOA
③ distributed computing: clusters, grids
④ systems management (automatic computing, automation).

→ cloud computing helps both consumers & providers:

*→ For consumers: reduction of IT-related costs
opposition to heavy investments.

* For providers:
  - better operational costs
  - hardware & software infrastructures serve many users; thus ↑sing efficiency.
  - Faster return on investment.
  - Lower total cost of ownership.

* Web services can glue together:
  - apps running on diff. product platforms.
  - enabling info from one app - available to others
  - enabling internal apps - available over the internet.

* In SOA: software resources are packaged as "services".
  - which are well-defined, self-contained modules.
  - provide standard functionality.
  - independent of the state.

* In consumer Web:
  - info & services are aggregated
  - acting as building blocks of complex compositions
  - called as service mashups.

* Grid Computing:
  - enables aggregation of distributed resources.
  - Key aspect is to build a standard web-services based protocols that allow distributed resources to be discovered, accessed, allocated, monitored & accounted for.

* OGSA:
  - addresses this by defining a set of core capabilities & behaviours
  - that address key concerns in grid systems.

→ Globus Toolkit - a middleware that implements a several standard grid services.

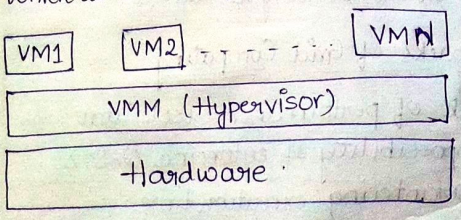
→ Drawbacks of Grid Computing:
  - Lack of performance isolation
  - Impossibility of enforcing QoS & guaranteeing execution time.
  - availability of resources with diverse software configurations, including desparate OS, libraries, compilers, etc.

→ Virtualization technology is the soln for the drawbacks of grids.

* Utility Computing:
  - users assign a utility value to their jobs; where utility is a fixed or time-varying valuation that captures various QoS constraints.

→ Hardware virtualization can be considered as a perfect fit to overcome most operational issues of data center building & maintainence.

Ex: A software layer, the virtual machine monitor (VMM) also called a hypervisor, mediates access to the physical hardware presenting to each guest OS a virtual machine, which is a set of virtual platform interfaces.



* Capabilities of virtualized system:
  - Isolation
  - Consolidation
  - migration

(1) Workload isolation is achieved
(2) Better reliability
(3) Better performance control
(4) Consolidation of several individual & heterogeneous workloads onto a single physical platform - leads to better system utilization.
(5) Workload migration targets at faulitating hardware maintainence, load balancing & disaster recovery.

* Virtual appliance: an application combined with environment needed to run it.
  - Amazon allows developers to share specialized amazon machine images (AMI) & monetize their usage of Amazon EC2.

→ In order to facilitate packing & distribution of software to be run on VMs several vendors devised the Open Virtualization format (OVF).

* Autonomic Computing:
  - systems shd manage themselves, with high level guidance from humans.

→ IBM's autonomic computing initiative - define the 4 properties of autonomic systems:
  - → self-configuration
  - → self-optimization
  - → self-healing
  - → self-protection

→ Autonomic computing inspire software technologies:
  - → management of service levels of running applications
  - → management of data center capacity
  - → disaster recovery.
  - → automation of VM provisioning.

* Layers & Types of Clouds:
  - → Iaas, Paas, Saas.

* Stack:

Service	Main Access & Management Tool.	Service Content
Saas.	Web Browser	Cloud Applications - CRM, videoprocessing
Paas	Cloud development environment.	Cloud Platforms: - Frameworks, Editors
Iaas.	Virtual Infrastructure Manager	Cloud Infrastructure - Firewall, Storage

→ Core middleware: manages physical resources & provide required features.

→ Cloud development environments are built on top of infrastructure services to offer application development & deployment.

* Deployment Models:

→ Public/Internet clouds:	Private/Enterprise clouds	Hybrid/Mixed clouds
• 3rd party, multi-tenant cloud services. • pay as you go	• Runs within a company data center for internal or partner use.	• Leasing public cloud services when private cloud capacity is insufficient.

→ public: pay as you go cloud for general public.
→ private: internal data center of a business.
→ community: shared by several organizations and supports specific community.
→ Hybrid: private + public clouds.

* Features of a Cloud:
1. Self-service
2. pay as you go.
3. Elastic
4. Customizable.

* Principles of Cloud Computing:

(1) Federation: capable of collaborating & resource sharing
  ↳ must be kept transparent & carried out in a secure & independent way.

(2) Independence:
  ↳ cloud services must be independent of provider's specific tool.

(3) Isolation:
  ↳ data in the same cloud must ~~ensure the~~ be separated from diff users and shd not be accessed.

(4) Elasticity:
  ↳ ease of accessing & releasing resources.

(5) Business Orientation:
  ↳ ensures quality of service & assist SLA.

(6) Trust:
  ↳ b/w consumers & service providers.

* Challenges, Risks:

→ Security, Privacy & Trust.
→ Data Lock-In & Standardization (portability & operability).
→ Availability, Fault Tolerance, Disaster Recovery
→ Resource Management & Energy Efficiency.

* SLA Management in Cloud Computing:
  ↳ Service Level-Agreement.

→ For web applications; response time & throughput are end user requests.

→ SLA is a legal agreement among enterprises ~~is regarded as~~ and infrastructure service providers to ensure QOS. (availability, storage, network).

These SLAs are known as infrastructure SLAs and service providers are known as application service providers.

→ Co-hosting of applications means deploying more than one application on single server.

→ To overcome the challenges of performance isolation & security guarantees; virtualization technologies are used.

→ Applications are hosted on VMs where resource allocation has 2 modes: conserving & non-conserving

→ Conserving mode: VM demanding more system resources than the specified quota cannot be allocated with spare resources.

→ Non-conserving mode: Spare resources not used by other VMs can be allocated.

→ Service Providers allocate resources more efficiently based on applications, which are called Application SLAs are used to monitor the metrics and such providers are called Managed Service Providers (MSP).

* TYPES OF SLA:

→ SLA can be modelled using web-service level agreement (WSLA)

→ Components of WSLA:
  - → metrics
  - → functions
  - → measurement directive
  - → service level parameter & objective
  - → penalty

① Infrastructure SLA:
  - → provider manages & offers guarantees on availability of infrastructure.
  - → machines are leased to customers & isolated from other users.

② Application SLA:
  - → co-location hosting model, the server capacity is available to the applications based on demand.
  - → Flexible in allocating & deallocating resources.

* Challenges of SLAs:

→ Application being a black box to MSP.

→ MSP needs to understand performance bottlenecks & scalability of app^n.

→ MSP analyzes the app^n; but further improvements or operations may effect the performance.

→ Risk of capacity planning.

* Lifecycle of SLA:

① Contract defn: defines a set of service offerings & corresponding SLA templates.

② Publication & Discovery: customers shd locate service providers through ads/catalogues

③ Negotiation:
↳ SLA terms & conditions are mutually agreed upon before signing.

④ Operalization:
→ Monitoring : measuring & calculating metr
→ accounting: capturing & archiving SLA adherence.
↳ Enforcement: appropriate actions for violation.

⑤ De-Commissioning:
↳ termination of activities

* SLA Management in the cloud: Slide 22-24 (SLA)

① Feasibility Analysis:
→ Technical – Compatibility, availability
→ Infrastructure – resources.
↳ Financial – cost & charges to customer.

② On-boarding application:
→ Moving application to MSP platform.
→ understanding runtime characteristics
↳ creation of necessary rules.

Steps
→ Packing application for deploying on physical or virtual environments
→ Packaged app^n is executed directly on physical servers to capture & analyze performance.
→ app^n is executed on VM and performance values are noted.
→ Based on performance values, SLAs are decided.
↳ Customer agrees to set of SLOs, policies are created.

→ Types of policies:
→ business : access to resources
→ operational : actions for diff thresholds
→ provisioning : sequence of actions for requests
↳ PP = collection (Request, Action)
OP = collection (Condition, Action)

→ Based on these policies applications are deployed on cloud.

③ Preproduction: verify & validate MSP findings on application's runtime & agree on defined SLA.

④ Production:

- appⁿ is accessible to end users under agreed SLA.
- On customer demand; ~~we can~~ provider can include new terms & conditions

⑤ Termination:

- When the customer decides to withdraw hosted application; all related data is transferred to customer, except for legal data.