UNIVERSIDAD DE VALLADOLID

PARALLEL COMPUTING AND EMERGING MODELS

June 10, 2018

Michal Ormos

Contents

Chapter 1

Distributed systems

1.1 History

Over past 60 years, computing technologies had experienced many series of platform and environmental changes that were driven by rising demand of applications workload and large data sets. A result of this phenomenon was rising demand in every few decades. These effect occur both to hight performance computing (HPC) and hight throughput computing (HTC). These lead to changes in architecture, operating systems, platform and network connectivity. In early years only centralized system were used, which conclude in creating parallel and distributed computing systems. These systems became data-intensive and network-centric.

In present years billions of people are connected to Internet every day. In these demands supercomputers and large data centers must provide high-performance computing service to huge number of Internet users connected at the same time. Because of that high-performance computing are no longer optimal method for measuring system performance. The emerge of clouds instead demands high-throughput computing (HTC).

1.1.1 High-Performance Computing (HPC)

High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

HPC introduce first computing paradigm which empathize the raw speed performance in 1990s in Gflops to 2010 in Pflops. FLOPS represent number of floating point operations per second, which can be understand as a method of encoding real numbers within the limits of finite precision available in computers. Using floating-point encoding, extremely long numbers can be handled relatively easily. A floating-point number is expressed as a basic number or *mantissa* , an *exponent*, and a *radix*(number base).

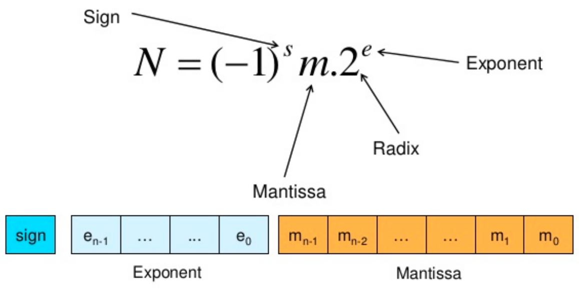


Figure 1.1: Explanation of FLOPS storing technique.

1.1.2 High-Throughput Computing (HTC)

For many experimental scientists, scientific progress and quality of research are strongly linked to computing throughput. In other words, most scientists are concerned with how many floating point operations per month or per year they can extract from their computing environment rather than the number of such operations the environment can provide them per second or minute. Floating point operations per second (FLOPS) has been the yardstick used by most High Performance Computing (HPC) efforts to evaluate their systems. Little attention has been devoted by the computing community to environments that can deliver large amounts of processing capacity over long periods of time. We refer to such environments as High Throughput Computing (HTC) environments.[2]

HTC paradigm pays more attention to high-flux computing, describing the use of many computing resources over long periods of time to accomplish a computational task.

1.1.3 Computing paradigms

In Figure 1.2 we introduced SOA. SOA is a style of applications architecture in such a way that they are composed of discrete software agents that have simple, well defined interfaces and are orchestrated through a loose coupling to perform a required function. As web 2.0 services, clouds and IoT which will be explained later.

Computing paradigm distinctions:

- **Centralized systems:** Every piece of the system is tight together in one place and

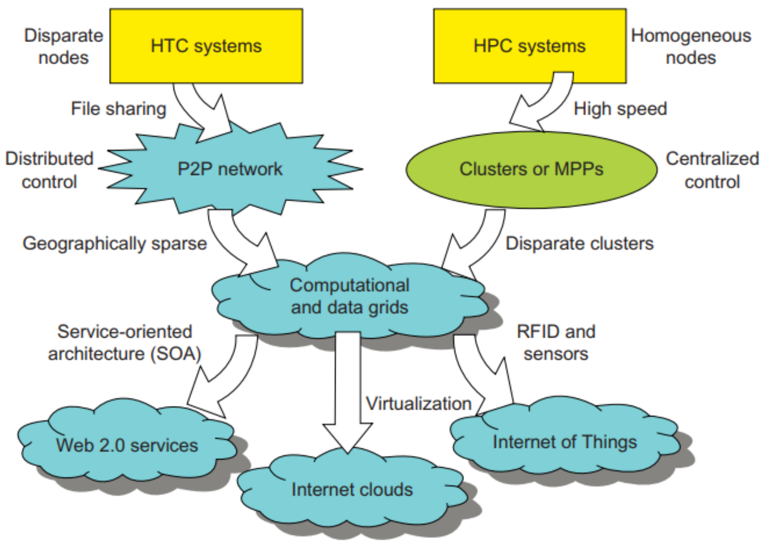


Figure 1.2: HPC vs. HTC diference throught history

accessed by terminal. Many data centers and supercomputers are centralized systems, but they are used in parallel, distributed, and cloud computing applications.

- **Parallel systems:** System is parallel connected together, many CPU's sharing one memory. An example of parallel computing would be two servers that share the workload of routing mail, managing connections to an accounting system or database, solving a mathematical problem etc.

- **Distributed systems:** A distributed system is a network that consists of autonomous computers that are connected using a distribution middle-ware. They help each other in sharing different resources and capabilities to provide users with a single and integrated coherent network.

- **Cloud:** Taking physical servers (WSC) and virtualized them to increase the number of accessible servers. This system is build on the idea that any of the servers is working full-time on 100% and mostly is idle. Based on Quality of Service (QoS) and Service Level Agreement (SLA). Using idea of utility computing, which means charging the clients by the amount of energy they consume from the server, pay-as-you-go. Cloud can work as centralized or distributed system.

1.1.4 Design objectives:

- **Efficiency:** Level of accomplishment to use resources wisely.
- **Dependability:** Reliability to provide high-throughput with Quality of Service (QoS). Guarantee that system will keep running or recover even under failure conditions.
- **Adaptation:** Measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workloads and service models.
- **Flexibility:** Measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

1.2 Scalable computing trends

It's well know that Moore's law and other predictable trends in technology used tob drive computing applications further. It's not sure if it will continue.

Besides the Moore's law we record also Gilder's law which, likewise to Moore's law logic indicates that network bandwidth will double each year.

Degrees of Parallelism

Bit-level parallelism (BLP) is the way of how historic big bulky computers were designed into bit-serial fashion over the years, user graduated from 4-bit microprocesors to 8-bit, 16-bit, 32-bit and now 64-bits CPUs.

This lead to next wave of improvement known as **instruction-level**, where the CPUs executes multiple instructions rather than only one instruction at a time, which later leads to pipeling, VLIW (very long instruction word) architectures, and multithreading requiring branch prediction, dynamic scheduling, speculation, and compiler support.

In that time **Data-level parallelism** (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vectors. From ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring **task-level parallelism** (TLP). Modern CPUs explores all of the parallelism types. As we move from parallel processing to distributed processing, we will see an increase in computing granularity to **job-level parallelism** (JLP).

Job-level brings the possibility of new applications as web services based on multiple tasks to provide complex services (maps, search, email) and Clusters service which is based on WSC (Warehouse-Scale Computers) Figure 1.3. Innovative applications have different requirements than traditional. Hundreds of thousands of users scalability.

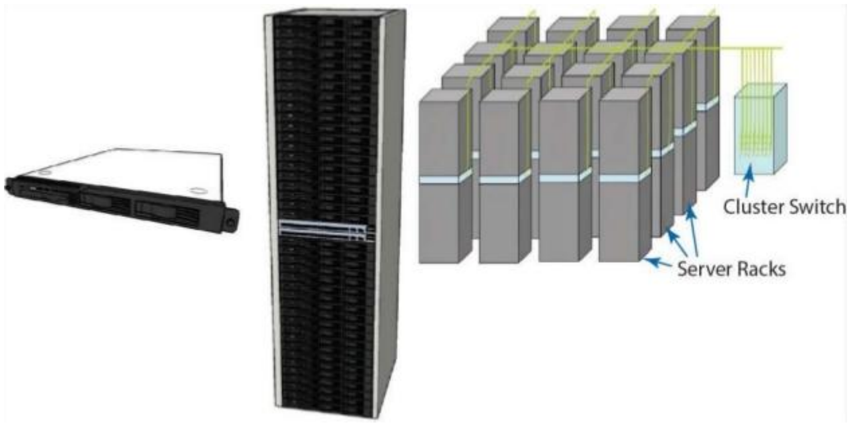


Figure 1.3: Sketch of the typical elements in warehouse. From (left) 1U server | (middle) 7' rack with Ethernet switch | (right) diagram of small clusters

Hardware and Software

Hardware of distributed systems was developing through years from single-core CPUs to Many-core CPUs, which lead to huge ability for instruction crunching - PFLOPS.

Many-core CPUs are usually partly GPU based, which can improve computing power.

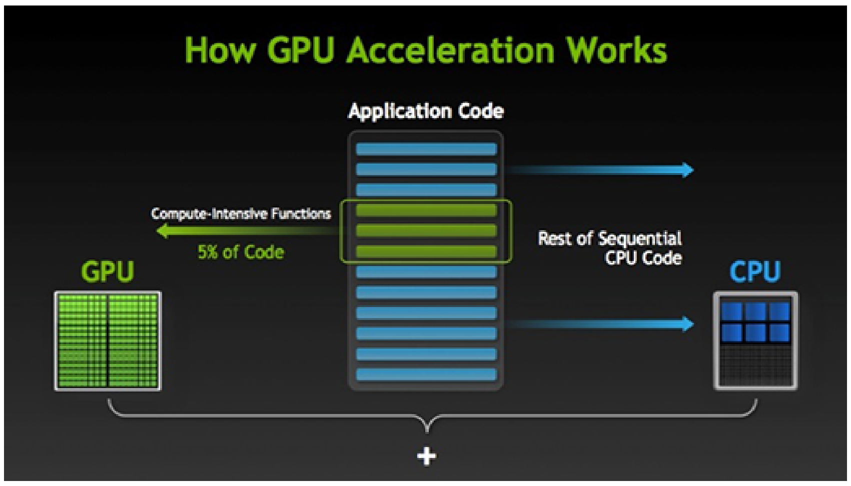

How GPU Acceleration Works
Application Code
Compute-Intensive Functions
5% of Code
GPU
Rest of Sequential CPU Code
CPU
+

Figure 1.4: Diagram of how GPU can improve computing time of some operations, better than CPU [3]

In case of Hardware also Simultaneous Multi-Threading (SMT) is used. SMT is a processor design that combines hardware multi threading with super-scalar processor technology to allow multiple threads to issue instructions each cycle.

In past years also memory as part of hardware was significantly improved mostly by increasing of capacity four times every three years. But access time didn't significantly changed over time. Which was caused by problems like memory walls and caches.

In case of solid disk capacity, we saw increasing by size, 10x every 8 years, but not increase by access time caused by mechanical hardware limits. Solid state disks (SSD) improve access time but have a durability problem and still don't have enough inner space requirements for reasonably price.

By type the type saving the data, we distinguish Storage Area Network (SAN), where disk are unite with the given system that they are not accessible by other system. As second type we use Network Attached Storage (NAS), where disk are connected to the network, so they are accessible by the whole network of other systems and machines.

Software Service-Oriented Architecture (SOA) is style of software design where services are provided to the other sections by components of applications. Trough communication protocols. They are independent unit of functionality. A set of provided services become a software application.

Distributed operation systems is a software over a collection of heterogeneous OS platforms. Independent, network communicating, physically separate, middle-ware with a limited degree of resource sharing.

Parallel and distributed programming models contribute to computing scalability and flexibility. For example distributed shared memory, shared address space in threads or synchronization by messages. Performed by tools tests as MPI, MapReduce or Hadoop. For example Hadoop library by Yahoo allow write and run applications over vast amounts of distributed data.

Performance metrics and scalability analysis

Are the ways of measuring the throughput of the system. Mostly used metrics nowadays are MIPS, TPS or already mentioned TFLOPS. Further metrics as job response time, latency, QoS etc.

System scalability can be amend by changing the size of machine, by software upgrades in OS or other OS components as libraries and software. Other ways for application scalability are matching the problem size scalability with machine size scalability or by technology as ability to adapt and fully use new core generations, new way of communication with hardware etc.

System availability

Probability that a system will work as required during the period of a mission. Measured by Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR).

$$System \quad availability = \frac{MTTF}{MTTF + MTTR} \tag{1.1}$$

System availability can be improved by hardware redundancy, design of testability, component reliability etc.

Networks threats and data integrity

Most of the systems are trying ensure security of their system by monitoring loss of confidentiality, integrity, system availability or inappropriate authentication. With right security approach and caution we can prevent this attacks.

Energy efficiency

Energy consumption generate the most expensive part of server costs. We want to ensure that server are running and not just in idle condition, so we can maximally used components, which are defined by their lifetime. Further we apply energy-aware applications or middleware layer with energy-efficient techniques for task scheduling.

1.2.1 Amdahl's and Gustafson's Law

Amdahl's law is formula which can give theoretical speedup(S) in latency of the task execution time(α) at fixed workload(W) by number of processors(n)

$$S = \frac{1}{\alpha + \frac{1-\alpha}{n}} \tag{1.2}$$

Gustafson's Law is formula which unlike of Amdahl's law give the theoretical speed at fixed execution time, which can be expected of a system whose resources are used. W = workload of a given program with 1 processor; α= fraction of sequential code in W; W = workload scaled for n processors.

$$S' = \frac{\alpha W + (1-\alpha) n W}{W} \tag{1.3}$$

Chapter 2

Clusters

Cluster is build of one or more computers, workstations, servers, blades, multi-core computers, etc. Possibly of different speeds and number of processors, called “nodes”, that are owned and managed by the same entity (a person, a group of people or a project) in which all the nodes run the same version of MOSIX and are configured to work tightly together. Clusters have evolved to data centers and WSC, to later became the base for clouds.

In other words cluster is collection of stand-alone interconnected computers which work together as single computer resource. In this fashion cluster contain high potential of availability, scalability, tolerance of faults, modular growth and massive parallelism. Massively parallel processors can begin high performance of processors as network.

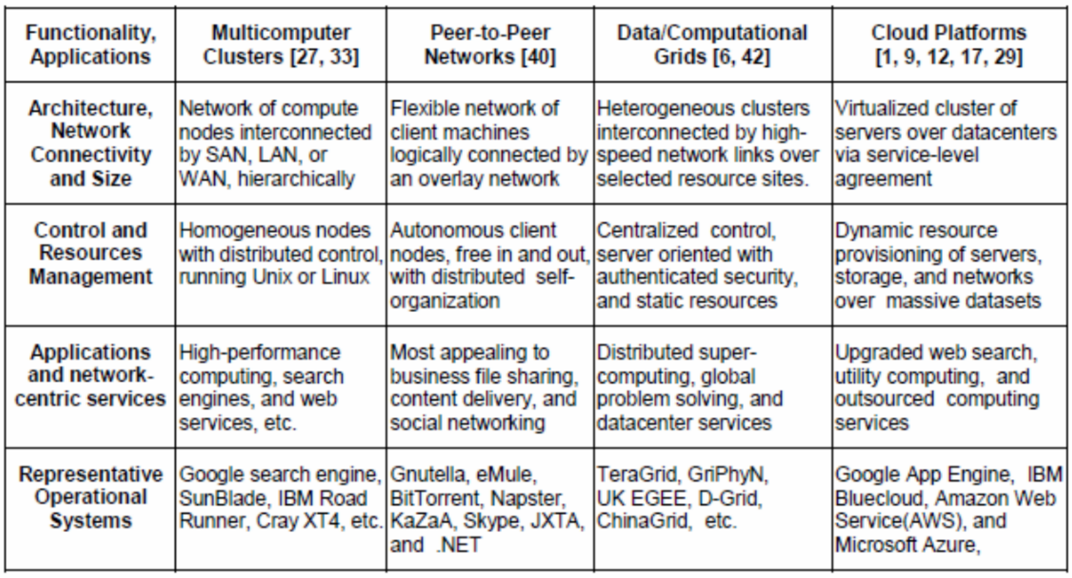
Clusters are based on the grid computing concept. Distant computers belonging to the same or different organization. Or sometimes as peer-to-peer, every node is client and server, without master-slave management, central coordination. With self-organized and distributed control.

2.1 Design principles of clusters

2.1.1 Scalability

Single-System Image (SSI). Cluster is used and seen as a single computer with multiply processes and resources. They are used with single interface and can be accessed from any node. Node is a participating computer (physical or virtual). The users aren’t aware of the location of physical devices, they agreed to shared all resources to cluster. We distinguish 3 types of nodes, from point of view of process P:

- Home node - node where the process P was created.



Functionality, Applications	Multicomputer Clusters [27, 33]	Peer-to-Peer Networks [40]	Data/Computational Grids [6, 42]	Cloud Platforms [1, 9, 12, 17, 29]
Architecture, Network Connectivity and Size	Network of compute nodes interconnected by SAN, LAN, or WAN, hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous clusters interconnected by high-speed network links over selected resource sites.	Virtualized cluster of servers over datacenters via service-level agreement
Control and Resources Management	Homogeneous nodes with distributed control, running Unix or Linux	Autonomous client nodes, free in and out, with distributed self-organization	Centralized control, server oriented with authenticated security, and static resources	Dynamic resource provisioning of servers, storage, and networks over massive datasets
Applications and network-centric services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed super-computing, global problem solving, and datacenter services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA, and .NET	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, Amazon Web Service(AWS), and Microsoft Azure,

Figure 2.1: Classification of Distributed Parallel Computing Systems [1]

- Local node - node on which process P is currently running, process was migrating.
- Remote nodes - remaining nodes of cluster.

Single-System Image is created from many parts which we will further explain:

- **Single file hierarchy**

  Can be imagined as "single huge" file system image, that integrates local and global disk of all other devices. The functionality of a single file hierarchy have already been partially provided by existing distributed file system in Figure 2.2. On this figure we use stable storage.

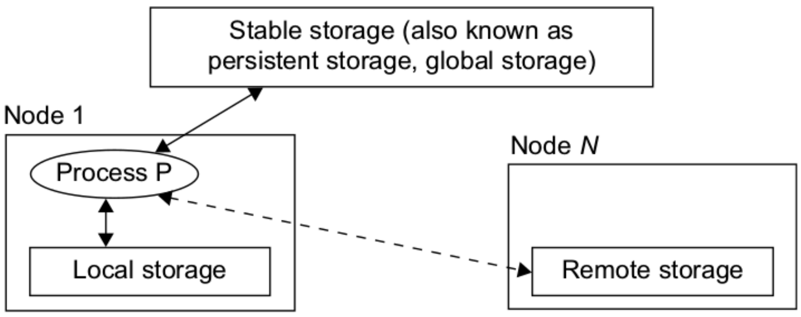

Stable storage (also known as persistent storage, global storage)
Node 1
Process P
Local storage
Node *N*
Remote storage

Figure 2.2: Three types of storage in a single file hierarchy. Solid lines show what process P can access and the dashed line shows what P may be able to access.

- **Single point of control**

  System administrator can configure, control, monitor and test all the clusters from single point through protocol SNMP.

- **Single entry point**

  Is very rich concept, consisting of single entry point, single file hierarchy, single I/O, etc. It enables users to log to a cluster as one virtual host, although the cluster may have multiply physical host nodes to serve the login session, for example through Telnet, HTTP, etc.

  In cluster as Figure 2.3 illustrates, is shown how DNS translates the symbolic name and return IP address of the least loaded node, which happen to be *Host1*. Only one user is shown, thousand of users can connect to the cluster in the same fashion.

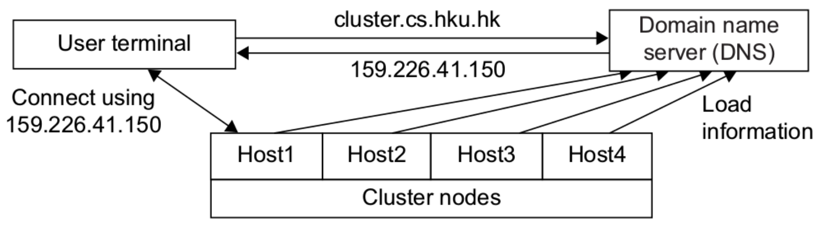


Figure 2.3: Realizing a single entry point in cluster using DNS

- **Single I/O space**

  Implemented over distributed RAID, shown on Figure 2.4. Four node linux cluster with three disks attached to SCSI bus of each node. All PCs can access both local and remote disks. The addressing scheme for all disk blocks is interleaved. **??**

- **Single networking**

  Any node of the cluster can use any network or I/O device as it were attached to the node which applied for it.

- **Single job management**

  All cluster jobs can be submitted from any node to a single job management system.

- **Single memory space**

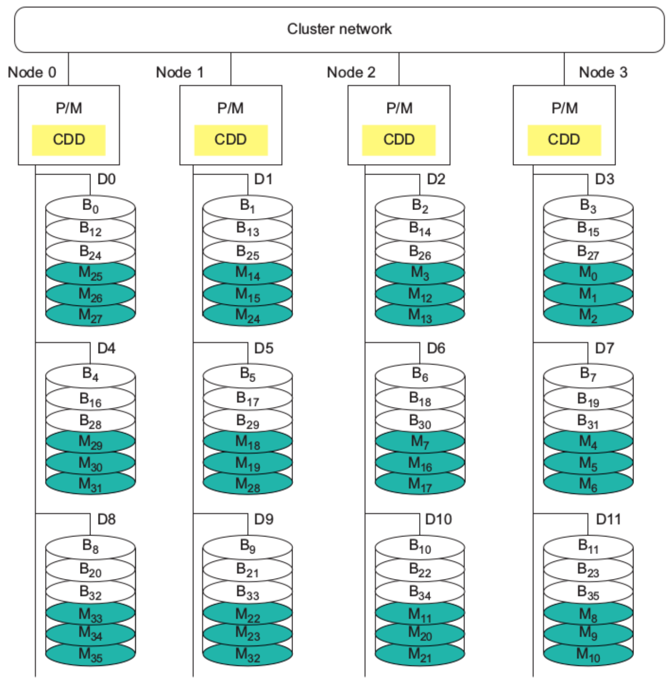


Figure 2.4: Distributed RAID architecture with a single I/O space over 12 distributed disks attached to 4 host computers in the cluster

As show on Figure 2.5, gives user illusion of a big, centralized main memory, which in reality may be clusters. A good way to test if a cluster has a single memory space is to run a sequential program that needs a memory space larger than any single node can provide.

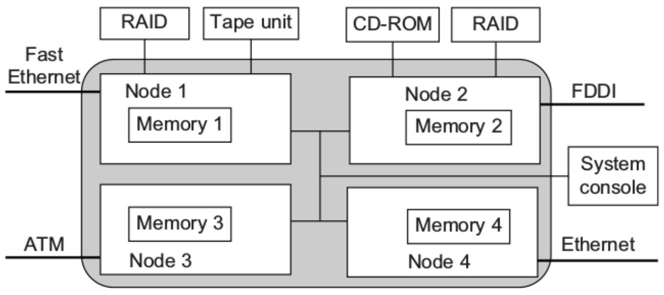


Figure 2.5: A cluster with single networking, single I/O space, single memory, and single point of control.

- **Single process space**

  All user processes created on various nodes form a single process space and share a uniform process identification scheme.

2.1.2 High-Availability (HA) capability

High-availability clusters also know as fail-over clusters try to utilized and support server applications with minimum amount of down-time, by using high availability software (we discussed availability in part 1.2).

As failure we understand event that prevents the system from normal operation, which can be unplanned or planned and temporarily permanent. Failure can be caused by hardware error, OS crash, bad upgrades, etc.

Possible fault-tolerant cluster configurations:

- **Hot standby server clusters**

  Only primary node doing useful work, other node are in standby mode and waiting if primary node fails to continue the work, then replace it from stand-by mode.

- **Active-takeover clusters**

  Two nodes are primary, and do useful work, when one fails other take over the work.

- **Failover clusters**

  When component fails, the reaming system take over the services of originally failed component.

After necessary fault, system have to recover. This recovery process take some time and is usually led by recovery schemes, which are planned before the possible fault, to prepare system and procedures for possible fault. Also checkpoints are used after significant running milestones to make fault recovery easier, by saving the state of the system at some significant points of computation.

2.2 Massively Parallel Processing (MPP)

Massively Parallel Processing is the use of many processors to perform a set of coordinated computations in parallel. Traditional MPP machines are distributed memory machines that use multiple processors.

As we can see in Top 500 [4], MPP machines are only potential opponent to Clusters in nowadays supercomputing. Most know MPP Clusters applications are IBM Blue Gene and Google search engine.

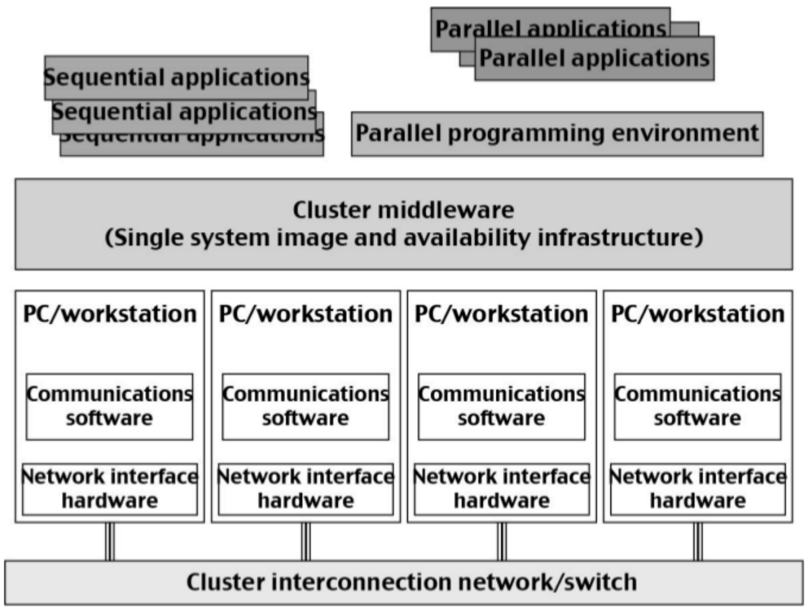


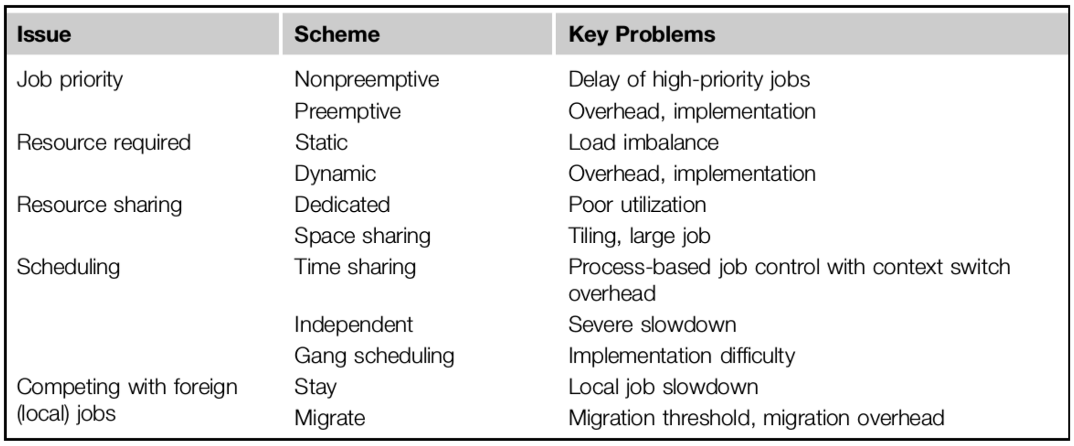
Figure 2.6: Example of MPP cluster

All these system are based on InfiniBand Architecture, point to point. Based on switches, connected through Internet by Host Channel Adapters (HCA) and Target Channel Adapters (TCA).

Cluster can be connected in three ways. Through most common LAN such as Ethernet network, shared disk or SCI bus. The shared-nothing architecture is used in most clusters, where the nodes are connected through the I/O bus. The shared-disk architecture is in favor of small-scale availability clusters in business applications. When one node fails, the other node takes over.

2.3 Job Scheduling Methods

Cluster jobs may be scheduled to run at a specific time (calendar scheduling) or when a particular event happens (event scheduling). Table 2.7 summarizes various schemes to resolve job scheduling issues on a cluster. Jobs are scheduled according to priorities based on submission time, resource nodes, execution time, memory, disk, job type, and user identity. With static priority, jobs are assigned priorities according to a predetermined, fixed scheme. The simplest scheme of job scheduling is to schedule jobs in a first-come, first-serve fashion.



Issue	Scheme	Key Problems
Job priority	Nonpreemptive	Delay of high-priority jobs
	Preemptive	Overhead, implementation
Resource required	Static	Load imbalance
	Dynamic	Overhead, implementation
Resource sharing	Dedicated	Poor utilization
	Space sharing	Tiling, large job
Scheduling	Time sharing	Process-based job control with context switch overhead
	Independent	Severe slowdown
	Gang scheduling	Implementation difficulty
Competing with foreign (local) jobs	Stay	Local job slowdown
	Migrate	Migration threshold, migration overhead

Figure 2.7: Job scheduling issues and schemes for cluster nodes

Chapter 3

Virtualization

3.1 Virtual Machines

Virtual Machine (VM) is an emulation of a computer system. They are based on computer architectures and provide functionality of a physical computer. Their implementation may involves special hardware, software or both.

Virtual Machines are divided to three types (as display in Figure 3.1):

- Native Virtual Machine - Virtual Machine Monitor (VMM) over the Hardware
- Hosted Virtual Machine - hosting operating system over the hardware
- Dual-mode Virtual Machine

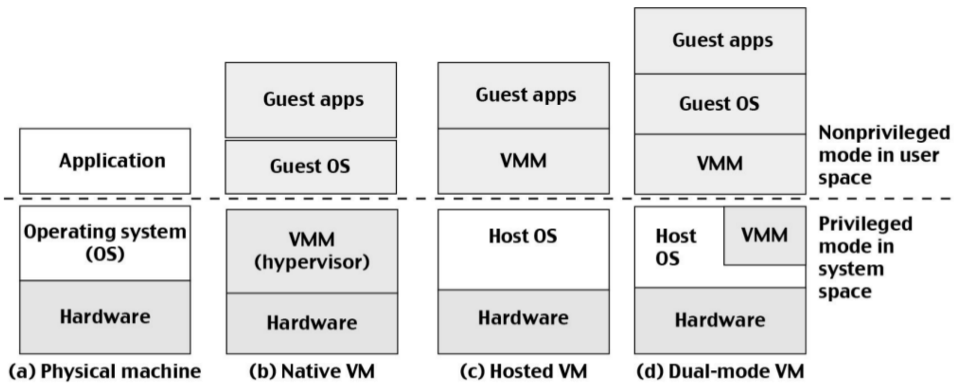


Figure 3.1: Native, Hosted and Dual-mode VM

3.1.1 Hypervisor Architecture

A hypervisor or Virtual Machine Monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines. A computer, on which a hypervisor runs one or more virtual machines is called a host machine. Hypervision provides hypercalls to guest-OSs and apps. We distinguish two VMM types: microkernel and monolithic.

Dynamic binary translation is the process of converting an executable binary, compiled for a particular ISA into an executable binary for a different ISA at runtime of the executable. This enables transparent execution of a binary compiled for a particular architecture on a different architecture, with no need to modify guest-OS. Done by instruction with two types: critical and non-critical.

Virtual Machine Monitor provides an abstraction identical to the physical machine, so the OS can just run over it as would run on physical hardware. VMM provides low level operations:

- multiplexing sever of VMs over the same hardware
- suspension of VM to storage
- re-suspend VM from hardware
- VM migration to different hardware

If hardware can be virtualized, than all the software environments can be also virtualized. For exmaple as Software as a Service (SaaS), which will be discussed in Chapter 4.

Abstraction levels of virtualization can be divided to 5 parts.

- **Instruction System Level (ISA) -** Allows to keep running legacy code in legacy hardware.
- **Hardware level -** On the top of the hardware we can generate virtual hardware. Software layer between hardware and OS. Done by Virtual Machine Monitor (VMM), which require complete control of the system resources. VMM is responsible for allocating all resources. VMM must be efficient, so programs suffer only minor decreasing in speed. Hardware virtualization may be slow, because it creates OS images from scratch.
- **Operating system level -** The same as in HW, can be done on top of OS. Also used for cloud based virtualization. OS virtualization insert a virtualization level inside the OS.

- **Library support level -** Emulator can run windows applications, on UNIX host by using the same API.
- **Application level -** For example Java virtual machine

Virtualization layer

Virtualization layer allows multiple customers to be housed on the same physical server with each client having a "private" area on the server to which critical resources are assigned. The performance of one customer's site does not impact the performance of other sites in a different partition. Also, the number of private areas on each server are limited, ensuring that resources are not over-leveraged.

Types of virtual architectures:

- Hypervision - between HW and guest-OS
- Para-virtualization - also between HW and guest-OS, but compiler help in speeding-up by bypassing the VMM
- Host-based virtualization - VMM on top of the host-OS

Para-virtualization

Para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Para-virtualization needs to modify the guest operating systems. Performance degradation is a critical issue of a virtualized system. The virtualization layer can be inserted at different positions in a machine soft-ware stack. Figure 3.2 illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized.

Hardware support for virtualization or hardware-assisted virtualization is a platform virtualization approach that enables efficient full virtualization using help from hardware capabilities, primarily from the host processors. Full virtualization is used to simulate a complete hardware environment, or virtual machine.

CPU Virtualization

CPU virtualization involves a single CPU acting as if it were multiple separate CPUs. The most common reason for doing this is to run multiple different operating systems on one machine. CPU virtualization emphasizes performance and runs directly on the available

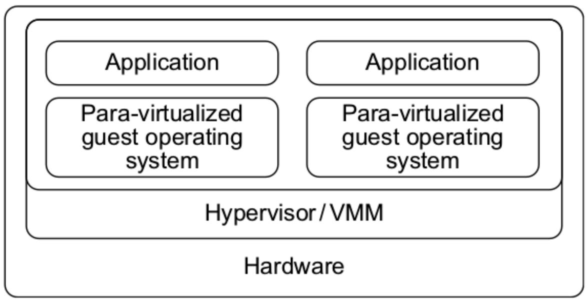

Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

Figure 3.2: Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization process

CPUs whenever possible. The underlying physical resources are used whenever possible and the virtualization layer runs instructions only as needed to make virtual machines operate as if they were running directly on a physical machine. VMM runs in supervisor mode. Hardware resources can help in CPU virtualization.

CPU virtualization is not the same thing as emulation. With emulation, all operations are run in software by an emulator. The emulator emulates the original computer's behavior by accepting the same data or inputs and achieving the same results.

Virtualization of memory

In a traditional execution environment, the operating system maintains mapping of virtual memory to machine memory with using page tables, which is a one-stage mapping from virtual memory to machine memory. However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs. That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory. The VMM is responsible for mapping the guest physical memory to the actual machine memory as in show in Figure 3.3.

3.1.2 Virtualization of Input/Output (I/O)

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. A single hardware device can be shared by multiple VMs

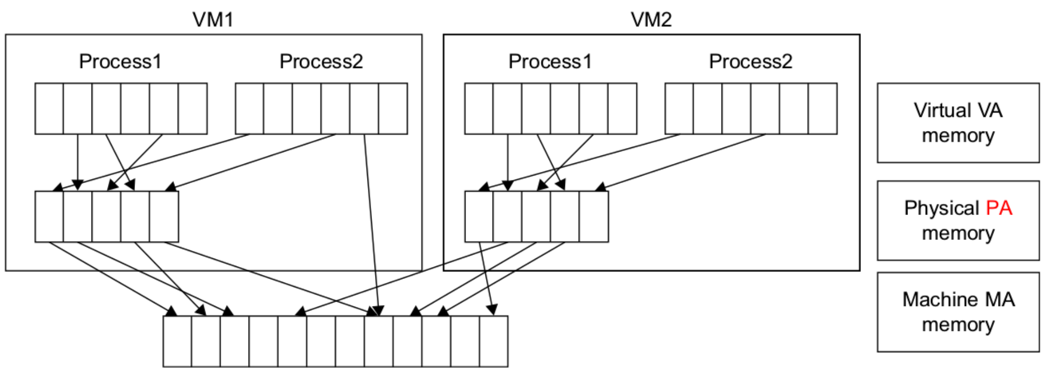


Figure 3.3

that run concurrently.

- **I/O Para-virtualization -** Each driver is divided to two parts, front-end which manages requests from guest-OS and back-end which manage the real I/O devices.
- **Direct I/O -** VMs use directly the I/O device, which work the best with networking devices. That requires hardware support.

Physical vs. virtual cluster

Virtual clusters in contrast with physical clusters are formed with physical machines or a VM hosted by multiple physical clusters. Provisioning VMs to a virtual cluster is done dynamically:

- Multiple VMs with different OSs can be deployed to the same physical node.
- Guest-OS can be different from host-OS where VM is implemented.
- VMs can be replicated in multiple physical servers.
- Number of nodes of virtual clusters may dynamically vary.
- Failure of a VM will not put down the host system.

Virtual machines fast deployment and scheduling

Deployment is construct of software stacks as OS, libraries and applications. Quickly switching between virtual clusters to use of resources at highest efficiency. Automatic scale-up/scale-down based on load balancing are big advantage of virtualization and increase resource utilization and reduces response time. High performance virtual storage is necessary for efficient VM deployment. Process of deployment of VM in cluster:

- Prepare disk image.
- Configure VM.
- Choose destination nodes.
- Execute VM deployment command on every host.

3.1.3 Live VM migration

Process of moving a running virtual machine or application between different physical machines without disconnecting the client or application is called live VM migration. Memory, storage and network connectivity is also transferred from original machine to the destination.

Main purpose of migration is enhance fail-over flexibility. When a VM fails, new VM replace it in the same or different host. In case of host failure, VM live migration allows the service to survive.

Virtual machine can be situated in 4 states: *inactive* (not enabled), *active* (performing a real task), *paused* (waiting), *suspended* (resources stored back to the disk).

Methods of virtual clusters migration:

- Guest-based manager - manager resides on a guest-OS.
- Host-based manager - manager is situated in actual physical node, in the host, when is supervising the VMs.
- Independent manager - uses both, guest and host.
- Integrated cluster - distinguishes between virtualized and physical resources.

Memory, files and network resources can be also migrated. Memory migration must be done efficiently from small to big sizes. Files system migration can be done in two solutions. As one virtual disk for each VM, contains file system or as global file system across all VMs. Network migration must maintain all open network connections, assign and keep virtual IP. In the manner we have to also need to have in mind, as in classic network UDP connection, that some packets may be lost.

Data centers

Server consolidation in data center is an approach to the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. The practice developed in response to the problem of server sprawl, a situation in which multiple, under-utilized servers take up more space and consume more resources than can be justified by their workload.

Virtual storage management means, that each of the running programs can assume, that it has access to all of the storage, defined by the architecture's addressing scheme. The only limit is the number of bits in a storage address. This ability to use a large number of storage locations is important because a program may be long and complex, and both the program's code and the data it requires must be in central storage for the processor to access them.

Data centers must be virtualized to serve as a cloud providers. For this task we use Virtual Infrastructure (VI). VI create VM and aggregate it into virtual cluster as a elastic resource. For this purpose we usually use OpenStack. OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds.

Because of VMM can change the computer architecture, means a SW layer between HW and VMs. VM encapsulates the state of the guest OS running on it, this is a security problem. VMM provides the security isolation needed by VMs, but once a attacker successfully enters a VMM, they have whole system of VM. By danger of this we create VM-based intrusion detection. By intrusion is mean unauthorized access to computer, OSs build on characterizing disreputable actions, etc.

Chapter 4

Clouds

Cloud architectures are built with commodity hardware and network (WSC). Design emphasize performance/price ratio instead of only performance. Clouds promises much more than data center model. Three types of clouds:

- **Public cloud**

  A public cloud is basically the internet. Service providers use the internet to make resources, such as applications (also known as Software-as-a-service) and storage, available to the general public, or on a public cloud. Owned by the service providers, which are responsible for creating and managing of VM within their infrastructure.

- **Private cloud**

  Private clouds are data center architectures owned by a single company that provides flexibility, scalability, provisioning, automation and monitoring. The goal of a private cloud is not sell "as-a-service" offerings to external customers, but instead to gain the benefits of cloud architecture without giving up the control of maintaining your own data center.

- **Hybrid cloud**

  By using a Hybrid approach, companies can maintain control of an internally managed private cloud while relying on the public cloud as needed. For instance during peak periods individual applications, or portions of applications can be migrated to the Public Cloud. This will also be beneficial during predictable outages: hurricane warnings, scheduled maintenance windows, rolling brown/blackouts.

Clouds are based on the paradigm that user pay-per-use of service. Based on subscription and quality of service. That is the principal cost reducer of cloud services. The mainly design

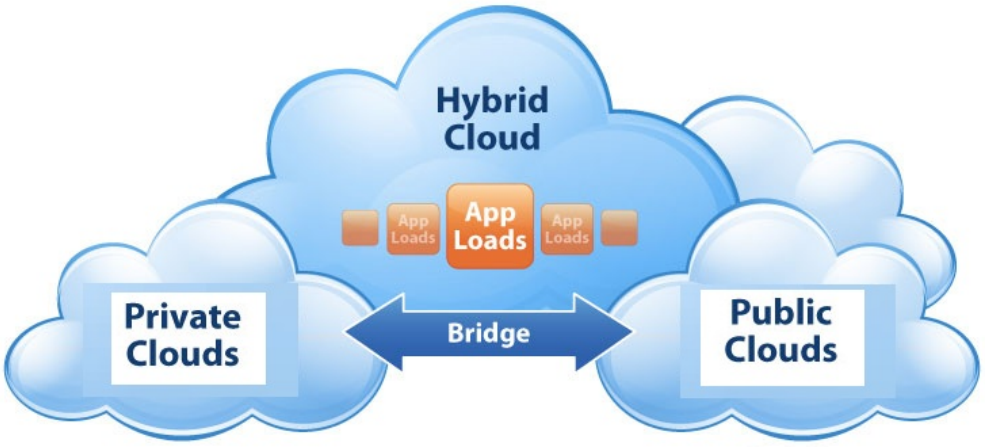


Figure 4.1: Three types of cloud, private, public and hybrid between them.

objectives are shifting computing from desktops to data centers, service provisioning, scalability in performance, data privacy protection, hight quality of cloud service, new standards and interfaces.

Three types of cloud application services:

- **IaaS**

  Infrastructure as a Service are self-service models for accessing, monitoring, and managing remote datacenter infrastructures, such as compute (virtualized or bare metal), storage, networking, and networking services (e.g. firewalls). Instead of having to purchase hardware outright, users can purchase IaaS based on consumption, similar to electricity or other utility billing. Users can control over OS, deploy and run apps as virtualizing resources.

- **PaaS**

  Platform as a Service are used for applications, and other development, while providing cloud components to software. Built on the top of IaaS. What developers gain with PaaS is a framework they can build upon to develop or customize applications. PaaS makes the development, testing, and deployment of applications quick, simple, and cost effective. With this technology, enterprise operations, or a third-party provider, can manage OS's, virtualization, servers, storage, networking, and the PaaS software itself. Developers, however, manage the applications. User can use programing languages and software tools provide by cloud.

- **SaaS**

  Software as a Service represent the largest cloud market and are still growing quickly. Built on the top of PaaS. SaaS uses the web to deliver applications that are managed by a third-party vendor and whose interface is accessed on the clients side. Most SaaS applications can be run directly from a web browser without any downloads or installations required, although some require plugins. User don't have to invest in server or software providers, just in stored data. Providers can keep very low costs.

4.1 Warehouse-Scale Data-Center design

As computation continues to move into the cloud, the computing platform of interest no longer resembles a pizza box or a refrigerator, but a warehouse full of computers. These new large datacenters are quite different from traditional hosting facilities of earlier times and cannot be viewed simply as a collection of co-located servers. Large portions of the hardware and software resources in these facilities must work in concert to efficiently deliver good levels of Internet service performance, something that can only be achieved by a holistic approach to their design and deployment. In other words, we must treat the datacenter itself as one massive warehouse-scale computer (WSC).

Clouds work on massive data centers up to one million servers. By the size of data-center we can lower the cost of unit, but with size, we still must ensure cooling, sufficient network bandwidth and also deal with hardware/software failures.

Very import part of making a good working data center is ensure good work of data center interconnection network, which can be sum up to five requirements:

- low latency
- high bandwidth
- low cost
- message-passing interface
- fault-tolerance

The network topology should support all message-parsing interface (MPI) communication patterns. Both point-to-point and collective MPI communications must be supported. The network should have high bisection bandwidth to meet this requirement.

The interconnection network should be expandable. With thousands or even hundreds of thousands of server nodes, the cluster network interconnection should be allowed to expand once more servers are added to the data center.

The interconnection network should provide some mechanism to tolerate link or switch failures. In addition, multiple paths should be established between any two server nodes in a data center. Fault tolerance of servers is achieved by replicating data and computing among redundant servers.

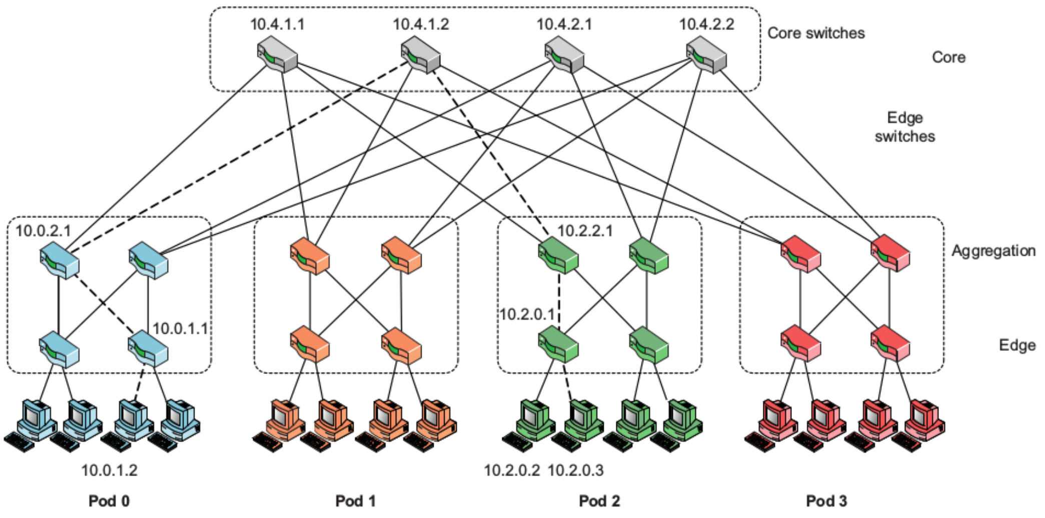


Figure 4.2: A fat-tree interconnection topology for scalable data-center construction.

4.2 Layered cloud architectural development

Cloud development architecture is divided to three layers:

- **Infrastructure layer**

  The basic layer of Cloud is the infrastructure IaaS. This layer is basically hardware and network. What distinguishes this from a regular server or hosting company are mainly two things, scalability and virtualization.

- **Platform layer**

The second layer of Cloud is the platform – the PaaS. The platform layer provides resources to actually build applications. In combination with IaaS, PaaS provides the ability to develop, test, run and host applications. The platform layer opens up for 3. parties to add their software (or integrations) to a Cloud service.

- **Service layer**

  The third Cloud layer is the actual Software – the SaaS. SaaS has been used for many years, but in a Cloud setting it is the layer in which the user consumes the offering from the service provider. The SaaS layer must be web based and hence accessible from everywhere and preferably on any device.

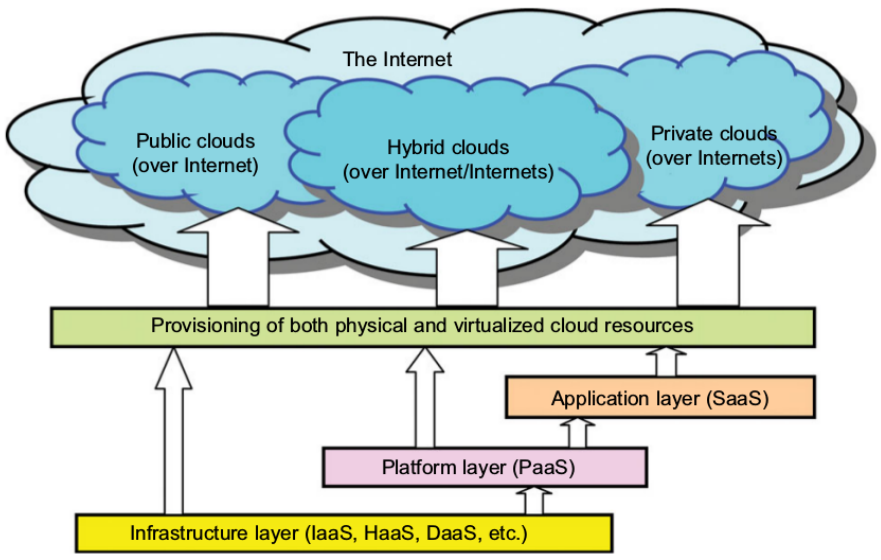

The Internet
Public clouds (over Internet)
Hybrid clouds (over Internet/Internets)
Private clouds (over Internets)
Provisioning of both physical and virtualized cloud resources
Application layer (SaaS)
Platform layer (PaaS)
Infrastructure layer (IaaS, HaaS, DaaS, etc.)

Figure 4.3: Layered architectural development of the cloud platform for IaaS, PaaS, and SaaS applications over the Internet.

Virtualization support for the Cloud

One very distinguishing feature of cloud computing infrastructure is the use of system virtualization and the modification to provisioning tools. Virtualization of servers on a shared cluster can consolidate web services. As the VMs are the containers of cloud services, the provisioning tools will first find the corresponding physical machines and then deploy the VMs to those nodes before scheduling the service to run on the virtual nodes.

In many cloud computing systems, virtualization software is used to virtualize the hardware. System virtualization software is a special kind of software which simulates the execution of hardware and runs even unmodified operating systems. Cloud computing systems use virtualization software as the running environment for legacy software such as old operating systems and unusual applications. It's give extremely flexibility to users, as they have fill privileges on their systems and also they have complete isolation.

4.2.1 Challenges of architectural design

Some of promising challenges in cloud design:

- **Service availability**

  Single company can cause single point of failure even in many of their data centers at once.

- **Data privacy and security**

  Public clouds are exposed to attack, as every user have access. They are vulnerable to classic network attacks as DoS, malware, worm etc. But also to cloud attacks as man-in-the-middle, hyper-vision malware, etc. This can be secure by encrypted storage, virtual LAN, network security as firewall etc.

- **Unpredictable performance**

  Mostly cause by I/O, by sharing them, which can cause conflicting between other applications or VM's. Solution is to virtualize disks, introduce interrupts etc.

- **Software bugs**

  Debugging large scale distributed bugs is difficult, because it cannot be reproduced. Virtualization can capture information very important for debugging.

- **Scalability, interoperability, standardization**

  Pay-as-you-go for computing is difficult.

- **Software licensing**

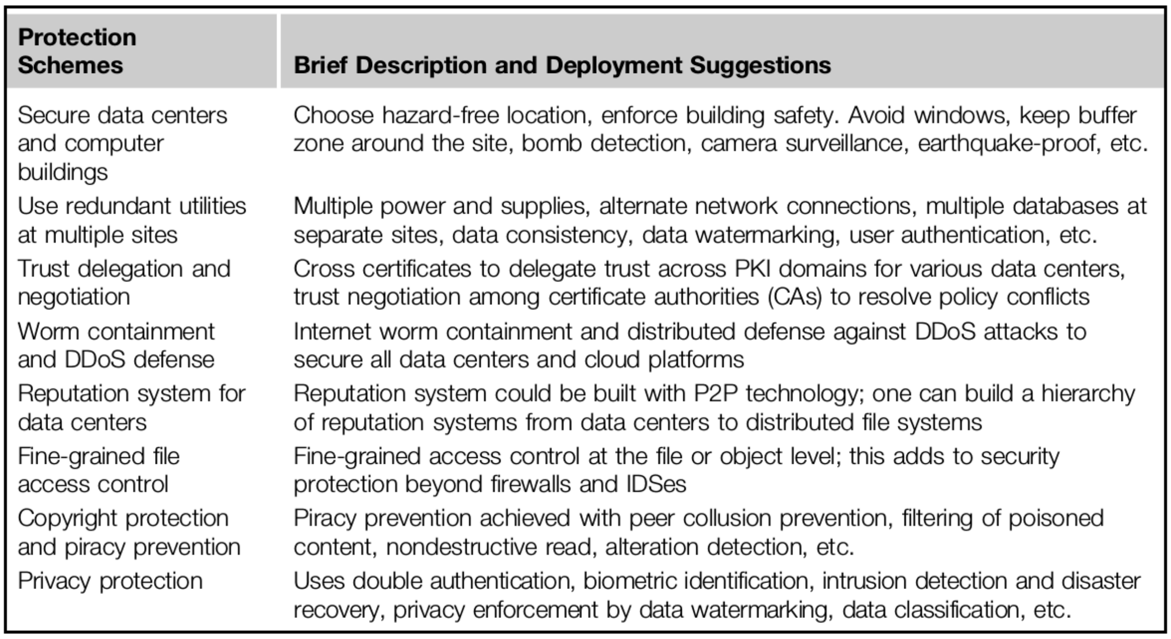
  Open source software is used because licensing is much better. Traditional software supplier must adapt to cloud model.

4.2.2 Cloud security

Trust and security are key to gain the confidence of providers and users. A healthy cloud ecosystem is desired to free users from abuses, violence, cheating, hacking, viruses, rumors, pornography, spam, and privacy and copyright violations. Three basic cloud security enforcements are expected:

- Facility security - Security of building which store cloud servers. Access restrictions, alarms, man traps, etc.
- Network security - Fault tolerant external firewalls, vulnerability assessment, etc.
- Platform security -SSL, data encryption, password policies, etc.

Virtualization enhances cloud security. But VMs add an additional layer of software that could become single point of failure. Security attacks in one VM are isolated and contained from affecting the other VMs. The Figure 4.4 lsit eight protection schemes to secure clouds and data centers.



Protection Schemes	Brief Description and Deployment Suggestions
Secure data centers and computer buildings	Choose hazard-free location, enforce building safety. Avoid windows, keep buffer zone around the site, bomb detection, camera surveillance, earthquake-proof, etc.
Use redundant utilities at multiple sites	Multiple power and supplies, alternate network connections, multiple databases at separate sites, data consistency, data watermarking, user authentication, etc.
Trust delegation and negotiation	Cross certificates to delegate trust across PKI domains for various data centers, trust negotiation among certificate authorities (CAs) to resolve policy conflicts
Worm containment and DDoS defense	Internet worm containment and distributed defense against DDoS attacks to secure all data centers and cloud platforms
Reputation system for data centers	Reputation system could be built with P2P technology; one can build a hierarchy of reputation systems from data centers to distributed file systems
Fine-grained file access control	Fine-grained access control at the file or object level; this adds to security protection beyond firewalls and IDSes
Copyright protection and piracy prevention	Piracy prevention achieved with peer collusion prevention, filtering of poisoned content, nondestructive read, alteration detection, etc.
Privacy protection	Uses double authentication, biometric identification, intrusion detection and disaster recovery, privacy enforcement by data watermarking, data classification, etc.

Figure 4.4: Physical and Cyber Security Protection at Cloud/Data Centers.

Distributed intrusion/anomaly detection

Data security is the weakest link in all cloud models. We need new cloud security standards to apply common API tools to cope with the data lock-in problem and network attacks or abuses. Security threats may be aimed at VMs, guest OSes, and software running on top of the cloud. IDSes attempt to stop these attacks before they take effect.

- **Distributed Defense against DDoS Flooding Attacks**

  A DDoS defense system must be designed to cover multiple network domains spanned by a given cloud platform. These network domains cover the edge networks where cloud resources are connected. DDoS attacks come with widespread worms.

- **Man in the middle attack**

  Migration from host A to host B via security vulnerable network. The man in the middle can view the VM contents being migrated, steal data and modify VM contents.

Data and software protection techniques

- **Data integrity and privacy protection**

  Users desire a software environment that provides many useful tools to build cloud applications over large data sets. In addition to application software, users need some security and privacy protection software for using the cloud. Such software should offer special APIs, fine-grained access control, protected shared data sets, personal firewalls, etc.

- **Data coloring and cloud watermarking**

  With shared files and data sets, privacy, security, and copyright information could be compromised in a cloud computing environment. Users desire to work in a trusted software environment that provides useful tools to build cloud applications over protected data sets. The user identification is also colored to be matched with the data colors. This color matching process can be applied to implement different trust management events.

Bibliography

[1] Kai Hwang, Jack Dongarra, Geoffrey C. Fox. *Morgan Kauthann.* (English) [*Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*]. Annalen der Physik, ISBN-13: 978-0123858801.

[2] Computing with HTCondor™,
`https://research.cs.wisc.edu/htcondor/htc.html`

[3] Main website of NVidia™,
`https://www.nvidia.com/object/what-is-gpu-computing.html`

[4] TOP500 Supercomputer Sites,
`https://www.top500.org/lists/2017/11/`