CLOUD COMPUTING

SAMRAT KRISHNA GADDAM
DR.T.S.RAVI KIRAN,DR.A.SRISAILA

DEDICATION

TO MY PARENTS AND BROTHER

Foreword

This book is a timely, comprehensive introduction to cloud computing. The phrase cloud computing, which was almost never used a decade ago, is now part of the standard vocabulary. Millions of people around the world use cloud services, and the numbers are growing rapidly. Even education is being transformed in radical ways by cloud computing in the form of massive open online courses (MOOCs). This book is particularly valuable at this time because the phrase cloud computing covers so many different types of computing services, and the many people participating in conversations about clouds need to be aware of the space that it spans. The introductory material in this book explains the key concepts of cloud computing and is accessible.

Preface

Cloud computing has recently emerged as one of the buzzwords in industry. Numerous IT vendors are promising to offer computation, storage, and application hosting services and to provide coverage in several continents, offering service-level agreements (SLA)-backed performance and uptime promises for their services. While these "clouds" are the natural evolution of traditional data centers, they are distinguished by exposing resources (computation, data/storage, and applications) as standards-based Web services and following a "utility" pricing model where customers are charged based on their utilization of computational resources, storage, and transfer of data. They offer subscription-based access to infrastructure, platforms, and applications that are popularly referred to as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). While these emerging services have increased interoperability and usability and reduced the cost of computation, application hosting, and content storage and delivery by several orders of magnitude, there is significant complexity involved in ensuring that applications

and services can scale as needed to achieve consistent and reliable operation under peak loads.

Currently, expert developers are required to implement cloud services. Cloud vendors, researchers, and practitioners alike are working to ensure that potential users are educated about the benefits of cloud computing and the best way .

Acknowledgements

First and foremost, we are grateful to all the contributing authors Dr.T.S.Ravi Kiran and Dr.A.Srisaila for their time, effort, and understanding during the preparation of the book. We thank editors of Notion Press book series on parallel and distributed computing, for his enthusiastic support and guidance during the preparation of book and enabling us to easily navigate through Notion Press publication process. All chapters were reviewed and authors have updated their chapters. We thank members for their time and effort in peer reviewing of chapters. Samrat Krishna would like to thank his family members, especially Finally, we would like to thank the staff at Notion Press(Chennai). They were wonderful to work with!

Prologue

An emerging internet based super computing model is represented by cloud computing. Cloud computing is the convergence and evolution of several concepts from virtualization, distributed storage, grid, and automation management to enable a more flexible approach for deploying and scaling applications. However, cloud computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. The concept of cloud computing on the basis of the various definitions available in the industry and the characteristics of cloud computing are being analyzed in this paper. The book also describes the main cloud service providers and their products followed by primary cloud computing operating systems.

CHAPTER ONE

Systems modeling, Clustering and virtualization

1.1 High Performance Computing (HPC) & High Throughput Computing (HTC)

High Performance Computing (HPC) is a somewhat ambiguous term. Supercomputer "…a computer at the frontline of contemporary processing capacity – particularly speed of calculation." We consider high performance computing to be when a large amount of high end resources are required to complete the task in a reasonable amount of time. By large, we mean dozens, hundreds, or thousands of CPU's. By high end resources, we mean the many cores per CPU with dozens or hundreds of gigabytes of RAM for each CPU. High performance computing tends to be performed by scientists and researchers in the biotech, geological, and astronomy spaces. Things like gene sequencing, oil discovery, and weather forecasting.

High Throughput Computing (HTC) is more ambiguous, but more simple, HTC describes "the use of many computing resources over long periods of time to accomplish a computational task." We'd say, high throughput computing is when a task is so big, it requires a large amount of resources, but those resources can be basic. You don't need high end CPU's with lots of RAM…you just need lots of computers. Dozens, hundreds, or thousands of computers. High throughput computing tends to be things like web crawling, where millions of different web sites need to be scrapped of their data. One computer wouldn't do this well, but 1,000 basic cloud systems would make quick work of the task. Another common example is video encoding– taking a video in one format and converting it into many others. Web sites that host video have to do this thousands of times each day. Its not a particularly complex compute task, they just have to do it a bunch of times.

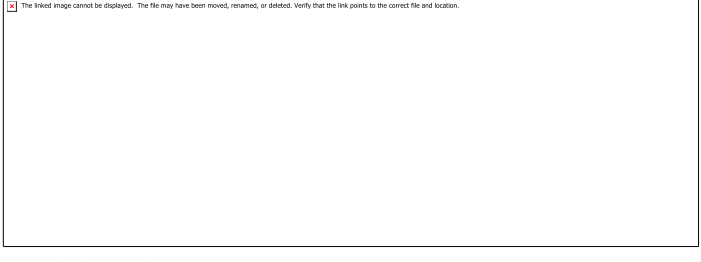


Figure1.1 HPC & HTC

When all is said and done, both HPC and HTC require the power of dozens, hundreds, and sometimes thousands of computers. Those computers might be physical

(your own datacenter) or virtual (in the cloud). Regardless, CPUsage can help you quickly and easily consume the power of dozens, hundreds, or thousands of computers for your high performance or high throughput computing tasks.

1.2 Performance Metrics and Scalability Analysis for virtual Machines.

VMWare provides the most comprehensive solution for server virtualization today. ManageEngine Applications Manager provides comprehensive performance metrics to monitor your VMware ESX/ESXi servers and their guest virtual machines, and helps you ensure they are performing well at all times. Applications Manager connects with VMware ESX/ESXi servers through APIs and determines the health status as well as the performance of host servers and their corresponding virtual machines.

With out-of-the-box reports, graphical views, alarms, thresholds and comprehensive fault management capabilities, administrators can maximize ESX server uptime and ensure that the guest virtual machines of the ESX/ESXi servers are running at peak performance.

Some of the virtual machine metrics provided by Applications Manager include:Availability, Health & CPU Usage

Monitor the current availability and health status of the virtual machines. Troubleshoot problems through the Root Cause Analysis (RCA) window. Know what portion of the ESX Server's CPU is used by each virtual machine and which virtual machine is consuming more CPU.

Also identify the hardware details of the ESX Server through hardware monitoring and minimize server downtimes caused due to a faulty hardware component.

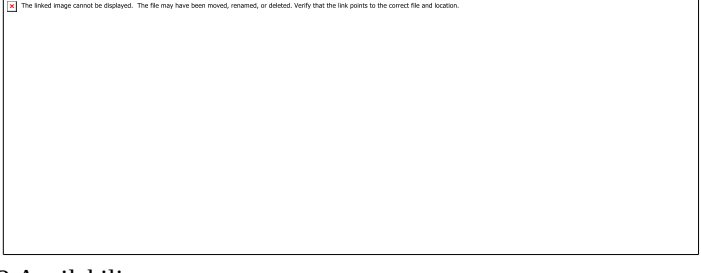


Figure1.2 Availability

Memory Utilization:

Avoid the problem of your virtual machines running out of memory. Get notified when the memory usage is high or memory becomes dangerously low. Metrics shown include consumed memory, active memory, overhead memory, shared memory, granted memory, reserved memory, etc.

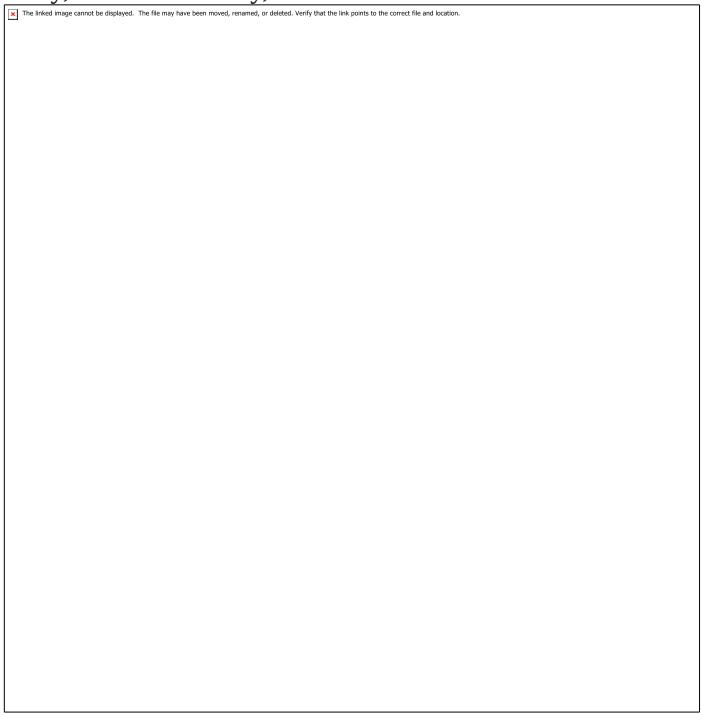


Figure1.3 Health & CPU Usage

ManageEngine Applications Manager also provides out-of-the-box reports that help analyze the overall performance of virtual machines and help in capacity planning.

- Gain insight into the performance of your VMware environment; troubleshoot and resolve problems before end users are affected.
- Plan capacity and make educated decisions about allocating virtual machines to host servers.
- Manage both physical and virtual components of your IT infrastructure using a single console.

- Agentless monitoring solution that is easy to set up and manage.
- Proven solution - Applications Manager was finalist in Best of InterOp 2010, under virtualization category

The VMware monitor is available as an add-on to Applications Manager and is priced $995/year for up to 25 servers or applications.

1.3 GPU Computing, Exascale & beyond

GPU-Accelerated Computing Goes Mainstream

GPU-accelerated computing has now grown into a mainstream movement supported by the latest operating systems from Apple (with OpenCL) and Microsoft (using DirectCompute). The reason for the wide and mainstream acceptance is that the GPU is a computational powerhouse, and its capabilities are growing faster than those of the x86 CPU.

In today's PC, the GPU can now take on many multimedia tasks, such as accelerating Adobe Flash video, transcoding (translating) video between different formats, image recognition, virus pattern matching and others. More and more, the really hard problems to solve are those that have an inherent parallel nature– video processing, image analysis, signal processing.The combination of a CPU with a GPU can deliver the best value of system performance, price, and power.

Exascale and Beyond:

As we bump up against the limits of processor speed, memory and energy consumption, we must rethink every aspect of scientific computing—from hardware and software, to algorithms, computer center efficiency and networking. The aim is reducing energy use, while producing more science per Watt.

Scientists and engineers in Berkeley Lab's Computational Research, National Energy Research Scientific Computing, Information Technology, and Environmental Energy Technologies divisions are working together to solve a significant problem faced by computing centers worldwide: how to engineer, build and operate power-efficient computers and data centers. Their research examines a wide range of issues, from creating new computer architectures using low-power processors to innovative building designs. ESnet researchers are also exploring how to improve the energy efficiency of national networks.

Since the advent of parallel computing in the early 1990s, supercomputer performance advanced by adding more processors running at higher speeds to the system. Once the performance of processors leveled off around 2006, systems designers turned to packing each chip with more cores. That was enough to achieve petaflop/s-level performance, but extending this approach to the next step—exascale computing—would be so energy intensive that no center could afford the 200 megawatts needed to power such a system each year.

To get beyond the current limitations of chip performance and energy demands, entirely new architectures will be needed and the Berkeley Lab Computing Sciences organization is looking for solutions. It may help to view the exascale problem as one of continuing to improve computing performance, rather than focusing solely on how to build the biggest supercomputer that we can. It's critical to start by creating the basic building blocks of such a system, but to create them in such a way that they can be assembled to scale to exaflops. This is no easy task, as the building blocks are systems of processors, memory system, storage systems, networking systems and more. Then there is the operating and software needed to make the system useful.

1.4 Massive Parallel Processors.

In the ever-changing landscape of information technology, data being collected and combed through for business intelligence purposes has reached excessive levels. Welcome to the big data revolution. In order for modern-day systems to keep up and process all this information in a timely manner, new technologies have been developed and old ones have been improved on. Massively parallel processing, or MPP, is such a technology. It deals with the coordinated processing of large datasets using multiple processors. This enables very fast rates of execution for intricate queries running against large data warehouses. The main reason this technology came about is because of the vast quantities of data that were inundating applications not designed for such massive volumes. The need to process this data for the purposes of analytics encouraged designers to develop ultra-fast processing techniques.

Without the techniques MPP employs, a query may take a very long time to complete, making modern business intelligence systems and data warehouses less than useful. Massively parallel processing is at the heart of many different types of big data solutions, and has made substantial inroads as an important technology. Amazon Redshift, the popular cloud-based data warehousing solution, uses MPP

architecture to achieve extremely fast query execution. MPP is one of the five key performance enablers of Redshift, along with columnar data storage, data compression, query optimization, and compiled code.

How MPP Works:

What MPP does is fairly simple in theory. It breaks up large, difficult to manage datasets into easily workable chunks, assigning each to a processor. There could be as many as hundreds or even thousands of processors working on chunks from the same dataset. Once all the data has been processed, results from the many different processors are combined for a final result set. How MPP actually achieves this is through a messaging interface the individual processors use to communicate with each other. Each processor has its own operating system and memory. This allows each one to work independently, on its assigned segment of the database, while loosely communicating with the other processors. It is for this reason that MPP systems are known as a "shared nothing" system. Since OS, memory, disk or anything else is not being shared among the individual nodes in a shared nothing architecture, possibilities and advantages for fine-tuning the system can be realized. The MPP system can be scaled to as many additional nodes as needed, dividing the work further and speeding up the system to optimum levels. Also, since the processor of each node operates independently of the others, there is no bottleneck to hinder performance.

Strengths and Weaknesses of MPP:

An MPP data warehouse has a multitude of advantages when considering the challenges that come along with the efficient extrapolation of big data analytics. However, it is not the right technology for every situation.

- **Strengths**: In addition to the advantages outlined above, data warehouses that employ MPP are equipped with an optimization mechanism that monitors the flow of data to the individual nodes. This makes them faster and more efficient than similar technologies. Also, MPP is cost effective, supports SQL-based business intelligence tools, and is generally easy to deploy, maintain, and use compared to competing technologies.
- **Weaknesses**: Even though MPP is a high-performance technology, it does have some drawbacks. One such weakness is the fact that unstructured data requires a bit of preprocessing. Unstructured data refers to data that does not have a predefined

organization, but its analysis is still considered to be of value in identifying trends and patterns for business intelligence purposes. However, this does not suggest that MPP cannot make use of unstructured data efficiently. It just uses a slightly different approach.

- **MPP vs. MapReduce:**Massively parallel processing is not the only technology available to facilitate the processing of large volumes of data. MapReduce, a part of the Apache Hadoop Project, is another technology that accomplishes the same things MPP does, but with some differences. As a matter of fact, you might even say MPP and MapReduce are distant cousins. At first glance, technological differences between the two may not seem too far apart, but depending on the needs of your data warehouse, choosing one over the other could have a huge impact.
- **Performance**: The optimization and distribution components of MPP allow it to manage the distribution of data among the separate nodes. This speeds up processing time considerably, making it a better performance choice over MapReduce. Also, MPP databases conform to the ACID compliance which stands for atomicity, consistency, isolation, and durability. ACID ensures database transactions are processed accurately and reliably. This is not something that is automatically enforced in Hadoop, giving MPP the overall performance edge over MapReduce.
- **Scaling**: The highly specialized hardware used by MPP systems make scalability a difficult and costly proposition. MapReduce and Hadoop, however, can be deployed to inexpensive commodity servers, allowing the clusters of nodes to grow as needed.
- **Deployment and Maintenance**: MPP is generally easy to deploy and maintain. Hadoop and MapReduce, on the other hand, can turn out to be a major implementation project requiring expensive and specialized expertise that may not be available in-house.
- **Data Restrictions**: Unlike MPP, Hadoop and MapReduce can handle unstructured data without the need to preprocess it. There is no need to massage the data before it can be used. The slight advantage here goes to MapReduce.
- **Language**: The language behind MapReduce's control mechanism is primarily Java. MPP uses SQL, making it easier to use and more cost effective. SQL is a well-known query language, generally used by database professionals, eliminating the need to hire costly Hadoop experts. Also, existing SQL-based business

intelligence tools are supported with MPP. This is not the case with MapReduce, and alternative solutions must be explored.

- **Is MPP better than MapReduce or vice versa?** That is a question that depends on what each organization wants to accomplish, for they are different tools that suit different situations. As a matter of fact, there are some organizations that use both MPP and MapReduce, affording them the luxury of the best of both worlds.

1.5 Degrees of Parallelisms.

Fifty years ago, when hardware was bulky and expensive, most computers were designed

in a bit-serial fashion. In this scenario, *bit-level parallelism (BLP)* converts bit-serial processing to word- level processing gradually. Over the years, users graduated from 4-bit microprocessors to 8-, 16-, 32-, and

64-bit CPUs. This led us to the next wave of improvement, known as *instruction-level parallelism (ILP)*, in which the processor executes multiple instructions simultaneously rather than only one instruction at a time.

For the past 30 years, we have practiced ILP through pipelining, superscalar computing, *VLIW (very long instruction word)* architectures, and multithreading. ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently. *Data-level parallelism (DLP)* was made popular through *SIMD (single instruction, multiple data)* and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly. Ever since the introduction of multicore processors and *chip multiprocessors (CMPs)*, we have been exploring *task-level parallelism (TLP)*. A modern processor explores all of the aforementioned parallelism types. In fact, BLP, ILP, and DLP are well supported by advances in hardware and compilers. However, TLP is far from being very successful due to difficulty in programming and compilation of code for efficient execution on multicore CMPs. As we move from parallel processing to distributed processing, we will see an increase in computing granularity to *job-level parallelism (JLP)*. It is fair to say that coarsegrain parallelism is built on top of fine-grain parallelism.

1.6 Computing Paradigms

Deployment Models The Cloud model promotes four deployment models:

- Private Cloud: The Cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.
- Community Cloud: The Cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.
- Public Cloud: The Cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling Cloud services.
- Hybrid Cloud: The Cloud infrastructure is a composition of two or more Clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., Cloud bursting for load-balancing between Clouds).
- Service Models Cloud Computing is gaining popularity to the extent that the new XaaS service category introduced will gradually take the place of many types of computational and storage resources used today . Cloud Computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) . Infrastructure-as-a-Service Infrastructure-as-a-Service, also called Hardware-as-a-Service was coined possibly in 2006. As the result of rapid advances in hardware virtualization, IT automation and usage metering& pricing, users could buy IT hardware, or even an entire data center, as a pay-as-you-go subscription service.
- Infrastructure-as-a-Service Cloud Computing: Paradigms and Technologies 49 (IaaS) or Hardware-as-a-Service (HaaS) solutions deliver IT infrastructure based on virtual or physical resources as a commodity to customers. These resources meet the end user requirements in terms of memory, CPU type and power, storage, and, in most of the cases, operating system as well. Users are billed on a pay-peruse basis. They have to set up their applications on top of these resources that are hosted and managed in data centers owned by the vendor. Amazon is one of the major players in providing IaaS solutions.

- Amazon Elastic Compute Cloud (EC2) provides a large computing infrastructure and a service based on hardware virtualization. By using Amazon Web Services, users can create Amazon Machine Images (AMIs) and save them as templates from which multiple instances can be run. It is possible to run either Windows or Linux virtual machines, for which the user is charged per hour for each of the instances running. Amazon also provides storage services with the Amazon Simple Storage Service (S3), users can use Amazon S3 to host large amount of data accessible from anywhere .
- Platform-as-a-Service Platform-as-a-Service solutions provide an application or development platform in which users can create their own application that will run on the Cloud. More precisely, they provide an application framework and a set of API that can be used by developers to program or compose applications for the Cloud. PaaS solutions often integrate an IT infrastructure on top of which applications will be executed. This is the case of Google AppEngine and Microsoft Azure, while other solutions, such as Manjrasoft Aneka, are purely PaaS implementations. Cloud computing services classification Category Characteristics Product type Vendors and products SaaS Customers are provided with applications that are accessible anytime and from anywhere Web applications and services (Web 2.0)
- SalesForce.com (CRM) Google documents Clarizen.com (Project management) Google mail (automation) PaaS Customers are provided with a platform for developing applications hosted in the Cloud Programming APIs and frameworks; Deployment system Google AppEngine Microsoft Azure Manjrasoft Aneka IaaS/ HaaS Customers are provided with virtualized hardware and storage on top of which they can build their infrastructure Virtual machines management infrastructure, Storage management Amazon EC2 and S3; GoGrid; Nirvanix 50 A. Shawish and M. Salama Google AppEngine is a platform for developing scalable web applications that run on top of data centers maintained by Google.
- It defines an application model and provides a set of APIs that allow developers to take advantage of additional services such as Mail, Datastore, Memcache, and others. AppEngine manages the execution of applications and automatically scales them up/down as required. Google provides a free but limited service, while utilizes daily and per minute quotas to meter and price applications

requiring a professional service. Azure is a Cloud service operating system that serves as the development, runtime, and control environment for the Azure Services Platform

- By using the Microsoft Azure SDK, developers can create services that leverage the .NET Framework. These services have to be uploaded through the Microsoft Azure portal in order to be executed on top of Windows Azure. Additional services, such as workflow execution and management, web services orchestration, and access to SQL data stores, are provided to build enterprise applications. Aneka, commercialized by Manjrasoft, is a pure PaaS implementation and provides end users and developers with a platform for developing distributed applications for the Cloud by using .NET technology. The core value of Aneka is a service oriented runtime environment that is deployed on both physical and virtual infrastructures and allows the execution of applications developed by means of various programming models.
- Aneka provides a Software Development Kit (SDK) helping developers to create applications and a set of tools for setting up and deploying Clouds on Windows and Linux based systems. Aneka does not provide an IT hardware infrastructure to build computing Clouds, but system administrators can easily set up Aneka Clouds by deploying Aneka containers on clusters, data centers, desktop PCs, or even bundled within Amazon Machine Images.
- Software-as-a-Service Software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of software maintenance, and reduces the expense of software purchases. Software-as-a-Service solutions are at the top end of the Cloud Computing stack and they provide end users with an integrated service comprising hardware, development platforms, and applications. Users are not allowed to customize the service but get access to a specific application hosted in the Cloud.
- Examples of SaaS implementations are the services provided by Google for office automation, such as Google Mail, Google Documents, and Google Calendar, which are delivered for free to the Internet users and charged for professional quality services.
- Examples of commercial solutions are SalesForce.com and Clarizen.com, which provide online CRM (Customer Relationship Management) and project management services, respectively. Cloud Computing: Paradigms and

Technologies.Data-as-a-Service Data in various formats and from multiple sources could be accessed via services by users on the network. Users could, for example, manipulate the remote data just like operate on a local disk or access the data in a semantic way in the Internet.

- Amazon Simple Storage Service (S3) provides a simple Web services interface that can be used to store and retrieve, declared by Amazon, any amount of data, at any time, from anywhere on the Web. The DaaS could also be found at some popular IT services, e.g., Google Docs and Adobe Buzzword. ElasticDrive is a distributed remote storage application which allows users to mount a remote storage resource such as Amazon S3 as a local storage device.

1.7 System Models for Distributed and Cloud Computing

System models are classified into four groups: clusters, P2P networks, computing grids, and Internet clouds over huge data centers. Cluster Architecture A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource. Figure shows the architecture of a typical server cluster built around a low-latency, high bandwidth interconnection network. This network can be as simple as a SAN (e.g., Myrinet) or a LAN (e.g., Ethernet).

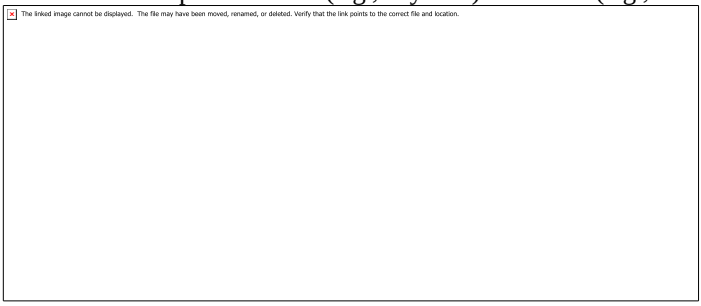


Figure1.4 Cluster

The gateway IP address locates the cluster. Most clusters have loosely coupled node computers. Single-System Image Ideal cluster should merge multiple system images into a single-system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes. An SSI is an illusion created by software or hardware that presents a collection of resources as one integrated,powerful resource.

A cluster with multiple system images is nothing but a collection of independent computers. Hardware, Software, and Middleware Support The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM or MPI, and a network interface card in each computer node. Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources. Many SSI features are expensive or difficult to achieve at various cluster operational levels. Instead of achieving SSI, many clusters are loosely coupled machines. Major Cluster Design Issues scalable performance efficient message passing, high system availability seamless fault tolerance cluster-wide job management Computational Grids.

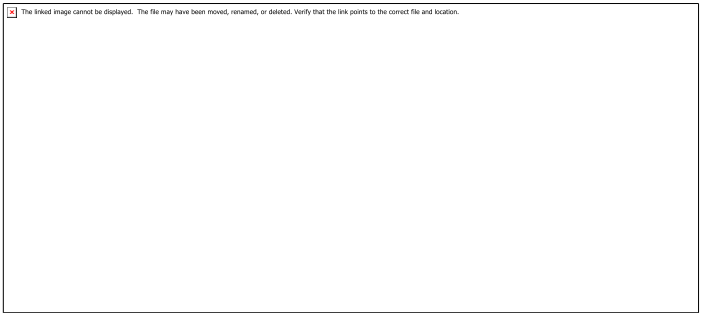


Figure1.5 Grid

A computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system. Figure shows an example computational grid built over multiple resource sites owned by different organizations. The grid is built across various IP broadband networks including LANs and WANs already used by enterprises or organizations over the Internet. The grid is presented to users as integrated resources pool as shown in the upper half of the figure.

The grid integrates the computing, communication, contents, and transactions as rented services. Peer-to-Peer Network The P2P architecture offers a distributed model of networked systems. A P2P network is client-oriented instead of server-oriented. In this system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet. All client machines act autonomously to join or leave the system freely.

This implies that no master-slave relationship exists among the peers. No central coordination or central database is needed. The architecture of a P2P network at two abstraction levels is shown in the figure. Initially, the peers are totally unrelated. Each peer machine joins or leaves the P2P network voluntarily. Only the participating peers form the physical network at any time.

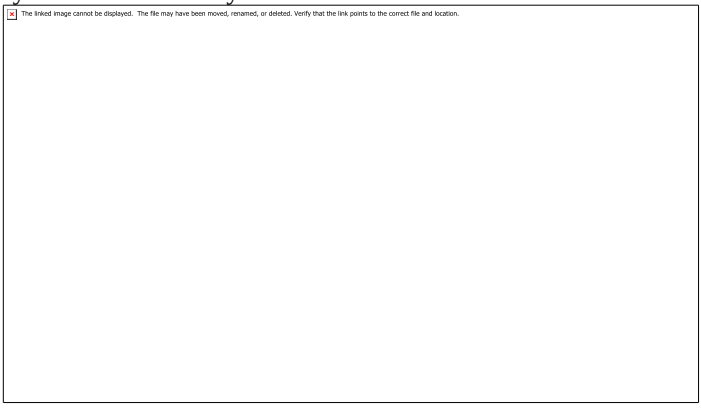


Figure1.6 P2P

Unlike the cluster or grid, a P2P network does not use a dedicated interconnection network. The physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols. Overlay Networks Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level. This overlay is a virtual network formed by mapping each physical machine with its ID, logically, through a virtual mapping as shown in Figure. Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level. There are two types of overlay networks: An unstructured overlay network is characterized by a random graph. There is no fixed route to send messages or files among the nodes.

Often, flooding is applied to send a query to all nodes in an unstructured overlay, thus resulting in heavy network traffic and nondeterministic search results. Structured overlay networks follow certain connectivity topology and rules for inserting and removing nodes (peer IDs) from the overlay graph.Routing mechanisms are developed to take advantage of the structured overlays. P2P networks are classified into four groups: The first family is for distributed file sharing of digital contents (music, videos, etc.) on the P2P network.

Collaboration P2P networks include MSN or Skype chatting, instant messaging, and collaborative design, among others. The third family is for distributed P2P computing in specific applications. Other P2P platforms, such as JXTA, .NET, and FightingAID@home, support naming, discovery, communication, security, and resource aggregation in some P2P applications. Computing challenges in P2P There are three types of heterogeneity problems in hardware, software, and network requirements. There are too many hardware models and architectures to select from; incompatibility exists between software and the OS; and different network connections and protocols make it too complex to apply in real applications. We need system scalability as the workload increases. System scaling is directly related to performance and bandwidth. P2P networks do have these properties.

Data location is also important to affect collective performance. Data locality, network proximity, and interoperability are three design objectives in distributed P2P applications. P2P performance is affected by routing efficiency and selforganization by participating peers. Fault tolerance, failure management, and load balancing are other important issues in using overlay networks. Lack of trust among peers poses another problem. The system is not centralized, managing it is difficult.

In addition, the system lacks security. Anyone can log on to the system and cause damage or abuse. In summary, P2P networks are reliable for a small number of peer nodes. They are only useful for applications that require a low level of security and have no concern for data sensitivity.

1.8 SOA with Applications

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification.

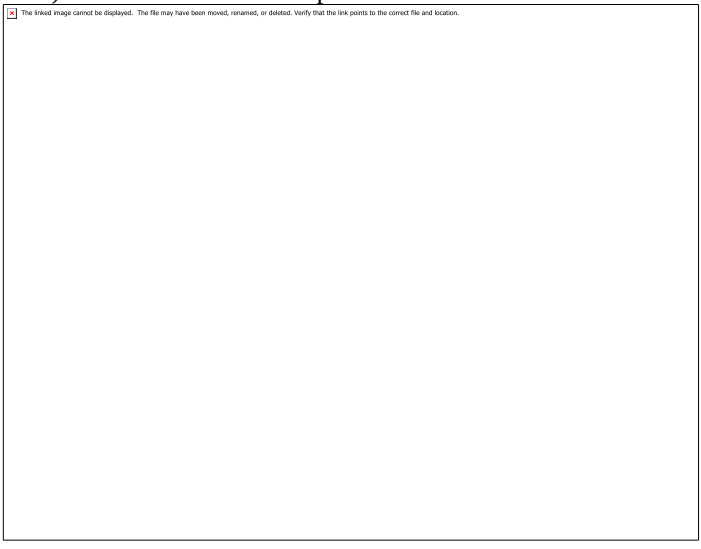


Figure1.7 Service Oriented Architecture

Services: If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

Connections: The technology of Web Services is the most likely connection technology of service-oriented architectures. The following figure illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. How those connections are defined is explained in Web Services Explained. A service provider can also be a service consumer.

Virtual Machines and Virtualization of Clusters and Data Centers

2.1Implementation Levels of Virtualization

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s [53]. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers. This virtualization technology has been revitalized as the demand for distributed and cloud computing increased sharply in recent years.

1. **Levels of Virtualization Implementation**

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure 2.1. After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure 2.2. This virtualization layer is known as hypervisor or virtual machine monitor (VMM). The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.

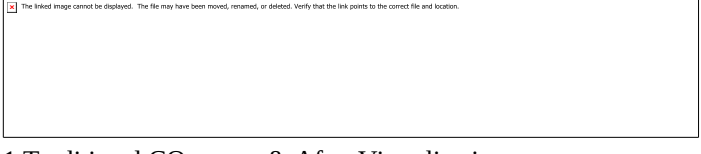


Figure 2.1 Traditional COmputer & After Virtualization

2.1.1 Instruction Set Architecture Level

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. With this approach, it is possible to run a large amount of legacy binary code writ-ten for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.

The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow. For better performance,dynamic binary translation is desired. This approach translates basic blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. Instruction set emulation requires binary translation and optimization. Avirtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

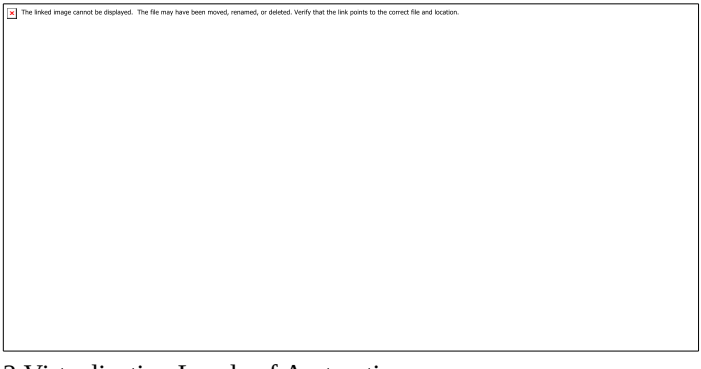


Figure 2.2 Virtualization Levels of Acstraction

1.

1. **Hardware Abstraction Level**

Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the

other hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer's resources, such as

its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently. The idea was implemented in the IBM VM/370 in the 1960s. More recently, the Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

1.
2. **Operating System Level**

This refers to an abstraction layer between traditional OS and user applications. OS-level virtualiza-tion creates isolated containers on a single physical server and the OS instances to utilize the hard-ware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

1.
3. **Library Support Level**

Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization. Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

1.
4. **User-Application Level**

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL)

VMs. In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.

Other forms of application-level virtualization are known as application isolation, applicationsandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations. An example is the LANDesk application virtuali-zation platform which deploys software applications as self-contained, executable files in an isolated environment without requiring installation, system modifications, or elevated security privileges.

1.

5. **Relative Merits of Different Approaches**

"Higher Performance" and "Application Flexibility" are self- explanatory. "Implementation Complexity" implies the cost to implement that particular vir-tualization level."Application Isolation" refers to the effort required to isolate resources committed to different VMs. Each row corresponds to a particular level of virtualization.

The number of X's in the table cells reflects the advantage points of each implementation level. Five X's implies the best case and one X implies the worst case. Overall, hardware and OS support will yield the highest performance. However, the hardware and application levels are also the most expensive to implement. User isolation is the most difficult to achieve. ISA implementation offers the best application flexibility.

2.2VMM Design Requirements and Providers

Hardware-level virtualization inserts a layer between real hardware and tradi-tional operating systems. This layer is commonly called the Virtual Machine Monitor (VMM) and it manages the hardware resources of a computing system. Each time programs access the hardware the VMM captures the process. In this sense, the VMM acts as a traditional OS. One hardware compo-nent, such as the CPU, can be virtualized as several virtual copies. Therefore, several traditional operating systems which are the same or different can sit on the same set of hardware simultaneously.

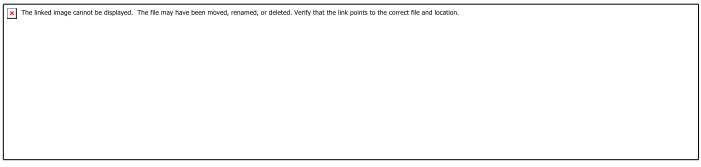


Figure 2.3 Merits of Virtualization at various levels

There are three requirements for a VMM. First, a VMM should provide an environment for pro-grams which is essentially identical to the original machine. Second, programs run in this environment should show, at worst, only minor decreases in speed. Third, a VMM should be in complete control of the system resources. Any program run under a VMM should exhibit a function identical to that which it runs on the original machine directly. Two possible exceptions in terms of differences are permitted with thisrequirement: differences caused by the availability of system resources and differences caused by timing dependencies. The former arises when more than one VM is running on the same machine.

The hardware resource requirements, such as memory, of each VM are reduced, but the sum of them is greater than that of the real machine installed. The latter qualification is required because of the intervening level of software and the effect of any other VMs concurrently existing on the same hardware. Obviously, these two differences pertain to performance, while the function a VMM pro-vides stays the same as that of a real machine. However, the identical environment requirement excludes the behavior of the usual time-sharing operating system from being classed as a VMM.

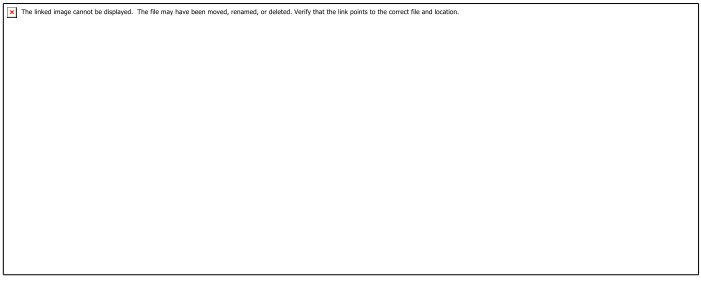


Figure 2.4 Comparision of four VMM and Hypervisor Software Packages

A VMM should demonstrate efficiency in using the VMs. Compared with a physical machine, no one prefers a VMM if its efficiency is too low. Traditional emulators and complete software interpreters (simulators) emulate each instruction by means of functions or macros. Such a method provides the most flexible solutions for VMMs. However, emulators or simulators are too slow to be used as real machines. To guarantee the efficiency of a VMM, a statistically dominant subset of the virtual processor's instructions needs to be executed directly by the real processor, with no software intervention by the VMM.

Complete control of these resources by a VMM includes the following aspects: (1) The VMM is responsible for allocating hardware resources for programs; (2) it is not possible for a program to access any resource not explicitly allocated to it; and (3) it is possible under certain circumstances for a VMM to regain control of resources already allocated. Not all processors satisfy these require-ments for a VMM. A VMM is tightly related to the architectures of processors.

It is difficult to implement a VMM for some types of processors, such as the x86. Specific limitations include the inability to trap on some privileged instructions. If a processor is not designed to support virtualization primarily, it is necessary to modify the hardware to satisfy the three requirements for a VMM. This is known as hardware-assisted virtualization.

2.3 XEN Architecture

Xen is a type 1 hypervisor that creates logical pools of system resources so that many virtual machines can share the same physical resources.

Xen is a hypervisor that runs directly on the system hardware. Xen inserts a virtualization layer between the system hardware and the virtual machines, turning

the system hardware into a pool of logical computing resources that Xen can dynamically allocate to any guest operating system. The operating systems running in virtual machines interact with the virtual resources as if they were physical resources.

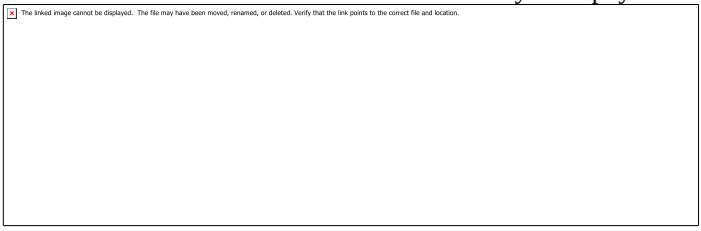


Figure 2.5 XEN Architecture

Xen is running three virtual machines. Each virtual machine is running a guest operating system and applications independent of other virtual machines while sharing the same physical resources.

Features:

The following are key concepts of the Xen architecture:

- Full virtualization.
- Xen can run multiple guest OS, each in its on VM.
- Instead of a driver, lots of great stuff happens in the Xen daemon, xend.

Full virtualization

Most hypervisors are based on *full virtualization* which means that they completely emulate all hardware devices to the virtual machines. Guest operating systems do not require any modification and behave as if they each have exclusive access to the entire system.

Full virtualization often includes performance drawbacks because complete emulation usually demands more processing resources (and more overhead) from the hypervisor. Xen is based on *paravirtualization*; it requires that the guest operating systems be modified to support the Xen operating environment. However, the user space applications and libraries do not require modification.

Operating system modifications are necessary for reasons like:

- So that Xen can replace the operating system as the most privileged software.
- So that Xen can use more efficient interfaces (such as virtual block devices and virtual network interfaces) to emulate devices — this increases performance.

Xen can run multiple guest OS each in its on VM

Xen can run several guest operating systems each running in its own virtual machine or domain. When Xen is first installed, it automatically creates the first domain, Domain 0 (or dom0).

Domain 0 is the management domain and is responsible for managing the system. It performs tasks like building additional domains (or virtual machines), managing the virtual devices for each virtual machine, suspending virtual machines, resuming virtual machines, and migrating virtual machines. Domain 0 runs a guest operating system and is responsible for the hardware devices.

Instead of a driver, lots of great stuff happens in the Xen daemon

The Xen daemon, xend, is a Python program that runs in dom0. It is the central point of control for managing virtual resources across all the virtual machines running on the Xen hypervisor. Most of the command parsing, validation, and sequencing happens in user space in xend and not in a driver.

IBM supports the SUSE Linux Enterprise Edition (SLES) 10 version of Xen which supports the following configuration:

- Four virtual machines per processor and up to 64 virtual machines per physical system.
- SLES 10 guest operating systems (paravirtualized only). Deploying virtualization.

2.4 Full Virtualization and Para Virtualization

Full virtualization:

Full virtualization is a common and cost-effective type of virtualization, which is basically a method by which computer service requests are separated from the physical hardware that facilitates them. With full virtualization, operating systems and their hosted software are run on top of virtual hardware. It differs from other forms of

virtualization (like paravirtualization and hardware-assisted virtualization) in its total isolation of guest operating systems from their hosts.

A private company called VMware developed a method to virtualize the x86 platform in 1998, which was previously believed to be impossible. The technology allowed multiple guest operating systems to run on a single host OS in full isolation using a combination of direct execution and binary translation. This was the first implementation of full virtualization, but certain inefficiencies have led to the development of other virtualization methods. These other methods include paravirtualization (which facilitates communication between the guest OS and the hypervisor in order to improve performance) and hardware-assisted virtualization (which gives virtual systems direct access to the hosting hardware, rather than merely its overlying software).

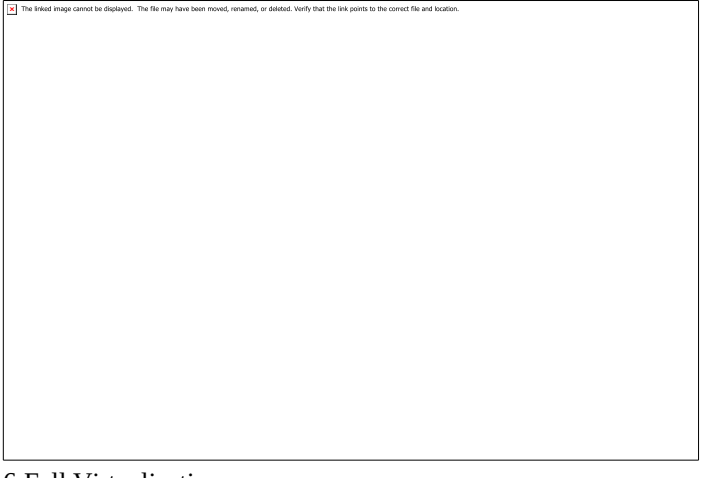


Figure 2.6 Full Virtualization

Para virtualization:

The modification of the source code of an operating system in order to run as a guest operating system in a specific virtual machine environment. Calls to the hardware from the guest OS are replaced with calls to the virtual machine monitor (VMM). For example, several operating systems, such as Linux, OpenBSD, FreeBSD and OpenSolaris, have been paravirtualized for the Xen virtual machine monitor.

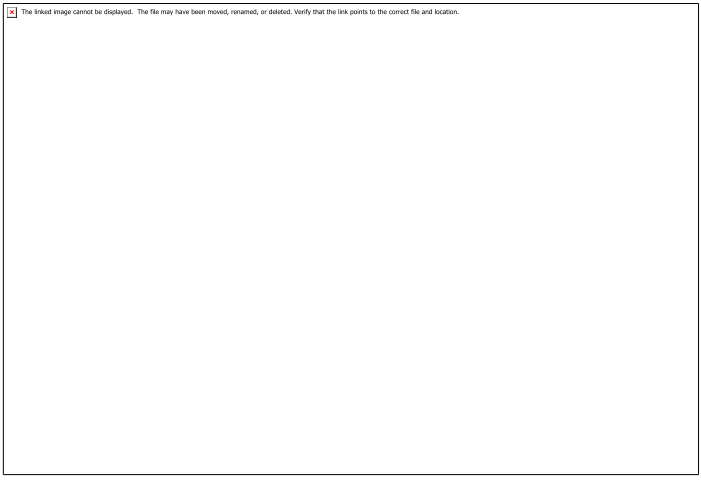


Figure 2.7 Para Virtualization

Paravirtualization Vs Emulation:

The guest OS can run as is without modification if the VMM emulates the hardware. In this case, the calls from the guest OS drivers to the hardware are intercepted and managed by the VMM, which redirects them to the real drivers. In addition, calls from the guest OS to the virtual memory page tables are intercepted and managed by the VMM. Emulation enables any guest OS to run intact, but emulation is slower than if the guest OS were paravirtualized.

Paravirtualization may be an option. If support for virtual machines is present in the CPU hardware, the guest OS may not need modification. For example, prior to the virtualization circuits built into x86 CPUs, the Xen VMM required the guest OS to be modified. If Xen runs in later Intel VT or AMD-V CPUs, a guest OS can run as is. See virtual machine monitor, virtual machine, hardware virtualization and Xen.

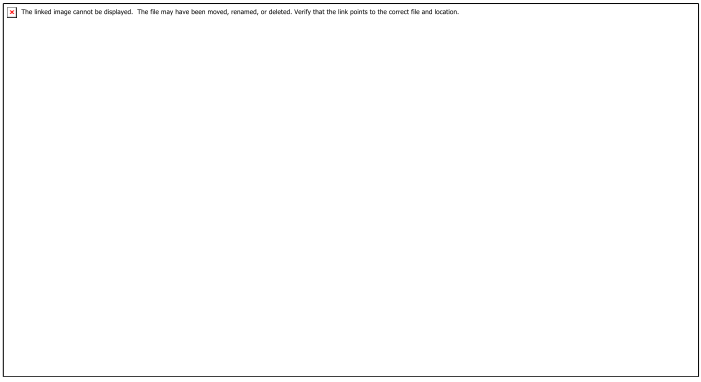


Figure 2.8 Architectural Comparision

Emulated Hardware:

Just like running in a non-virtualized computer, a non-paravirtualized guest OS communicates with the hardware as usual. The VMM presents a "device model" to the guest OS, which emulates the hardware. In these illustrations and the one following, the emphasis is on the device drivers. Paravirtualization also refers to modifying the calls to the virtual memory tables.

Paravirtualized Guests:

In a paravirtualized OS, the drivers are replaced with calls to the VMM interface. This example shows one VMM model.

2.5 Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance. 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. Furthermore, MMU virtualization should be supported, which is

transparent to the guest OS. 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.

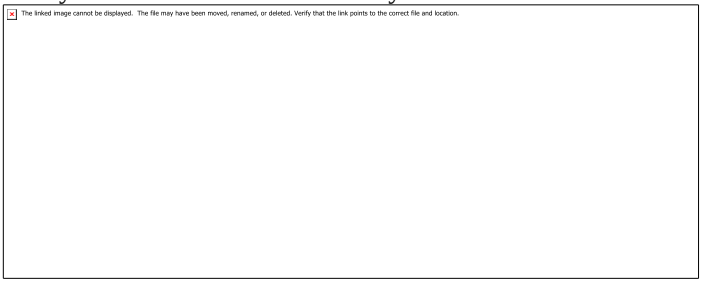


Figure 2.9 Two Level Memory Mapping Procedure

Since each page table of the guest OSes has a separate page table in the VMM corresponding to it, the VMM page table is called the shadow page table. Nested page tables add another layer of indirection to virtual memory. The MMU already handles virtual-to-physical translations as defined by the OS. Then the physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor. Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded. Consequently, the perfor-mance overhead and cost of memory will be very high.

VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation. Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to a physicalmemory mapping, the VMM updates the shadow page tables to enable a direct lookup. The AMD Barcelona processor has featured hardware-assisted memory virtualization since 2007. It provides hardware assistance to the two-stage address translation in a virtual execution environment by using a technology called nested paging.

Extended Page Table by Intel for Memory Virtualization:

Since the efficiency of the software shadow page table technique was too low, Intel developed a hardware-based EPT technique to improve it, as illustrated in Figure 3.13.

In addition, Intel offers a Virtual Processor ID (VPID) to improve use of the TLB. Therefore, the performance of memory virtualization is greatly improved.

When a virtual address needs to be translated, the CPU will first look for the L4 page table pointed to by Guest CR3. Since the address in Guest CR3 is a physical address in the guest OS, the CPU needs to convert the Guest CR3 GPA to the host physical address (HPA) using EPT. In this procedure, the CPU will check the EPT TLB to see if the translation is there. If there is no required translation in the EPT TLB, the CPU will look for it in the EPT. If the CPU cannot find the translation in the EPT, an EPT violation exception will be raised.

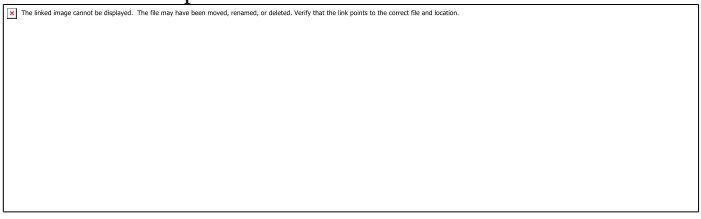


Figure 2.10 Two Level Memory Mapping Procedure

If the entry corresponding to the GVA in the L4 page table is a page fault, the CPU will generate a page fault interrupt and will let the guest OS kernel handle the interrupt. When the PGA of the L3 page table is obtained, the CPU will look for the EPT to get the HPA of the L3 page table, as described earlier. To get the HPA corresponding to a GVA, the CPU needs to look for the EPT five times, and each time, the memory needs to be accessed four times. There-fore, there are 20 memory accesses in the worst case, which is still very slow. To overcome this short-coming, Intel increased the size of the EPT TLB to decrease the number of memory accesses.

2.6 Implementation Levels of Virtualization

Implementation Levels of Virtualization Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. The idea is to separate the hardware from the software to yield better system efficiency.

A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure. After virtualization, different user

applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure. Virtualization can be implemented at various operational levels, as given below: Instruction set architecture (ISA) level Hardware level Operating system level Library support level

Application level Instruction Set Architecture Level At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired. This approach translates basic blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. A virtual instruction set architecture (V-ISA) thus requires adding a processor- specific software translation layer to the compiler.

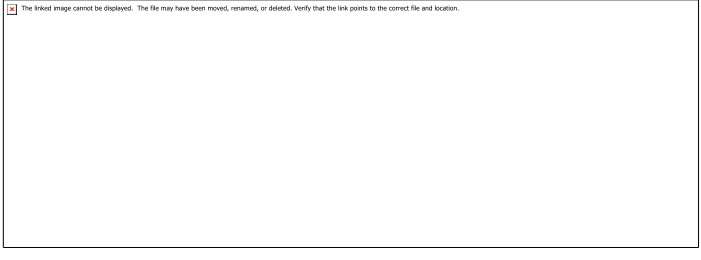


Figure 2.11 Memory Virtualization

Hardware Abstraction Level It is performed right on top of the bare hardware and generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices so as hardware utilization rate by multiple users concurrently may be upgraded Operating System Level OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers.

The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large

number of mutually distrusting users. Library Support Level Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration. User-Application Level On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL) VMs. In this scenario, the virtualization layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it.

The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM. Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations. Virtualization Support at the OS Level It is slow to initialize a hardware-level VM because each VM creates its own image from scratch and storage of such images are also slow. OS-level virtualization provides a feasible solution for these hardware-level virtualization issues. OS virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources. It enables multiple isolated VMs within a single operating system kernel.

This kind of VM is often called a virtual execution environment (VE). This VE has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings. Advantages: VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high Scalability It is possible for a VM and its host environment to synchronize state changes when necessary.

2.7 Live VM Migration Steps and Performance Effects

In a cluster built with mixed nodes of host and guest systems, the normal method of operation is to run everything on the physical machine. When a VM fails, its role could be replaced by another VM on a different node, as long as they both run with

the same guest OS. In other words, a physical node can fail over to a VM on another host. This is different from physical-to-physical failover in a tradi-tional physical cluster. The advantage is enhanced failover flexibility. The potential drawback is that a VM must stop playing its role if its residing host node fails. However, this problem can be mitigated with VM life migration.

There are four ways to manage a virtual cluster. First, you can use a guest-based manager, by which the cluster manager resides on a guest system. In this case, multiple VMs form a virtual cluster. For example, openMosix is an open source Linux cluster running different guest systems on top of the Xen hypervisor. Another example is Sun's cluster Oasis, an experimental Solaris cluster of VMs supported by a VMware VMM. Second, you can build a cluster manager on the host systems. The host- based manager supervises the guest systems and can restart the guest system on another physical machine. A good example is the VMware HA system that can restart a guest system after failure.

These two cluster management systems are either guest-only or host-only, but they do not mix. A third way to manage a virtual cluster is to use an independent cluster manager on both the host and guest systems. This will make infrastructure management more complex, however. Finally, you can use an integrated cluster on the guest and host systems. This means the manager must be designed to distinguish between virtualized resources and physical resources. Various cluster management schemes can be greatly enhanced when VM life migration is enabled with minimal overhead.

VMs can be live-migrated from one physical machine to another; in case of failure, one VM can be replaced by another VM. Virtual clusters can be applied in computational grids, cloud platforms, and high- performance computing (HPC) systems. The major attraction of this scenario is that virtual cluster-ing provides dynamic resources that can be quickly put together upon user demand or after a node failure. In particular, virtual clustering plays a key role in cloud computing. When a VM runs a live ser-vice, it is necessary to make a trade-off to ensure that the migration occurs in a manner that minimizes all three metrics. The motivation is to design a live VM migration scheme with negligible downtime, the lowest network bandwidth consumption possible, and a reasonable total migration time.

Furthermore, we should ensure that the migration will not disrupt other active services residing in the same host through resource contention (e.g., CPU, network bandwidth). A VM can be in one of the following four states. An inactive state is defined by the virtualization platform, under which the VM is not enabled. An active state refers to a VM that has been instantiated at the virtualization platform to perform a real task. A paused state corresponds to a VM that has been instantiated but disabled to process a task or paused in a waiting state. A VM enters the suspended state if its machine file and virtual resources are stored back to the disk.

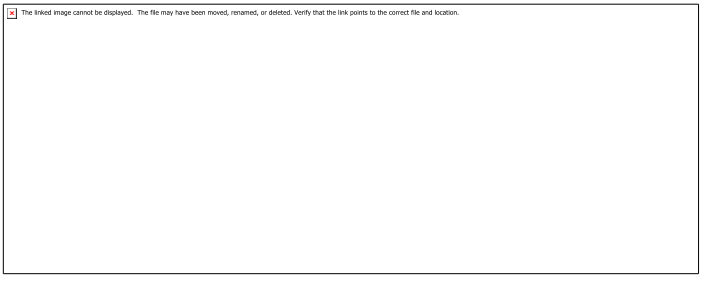


Figure 2.12 Live Migration of VM from host to other

Steps 0 and 1: Start migration. This step makes preparations for the migration, including determining the migrating VM and the destination host. Although users could manually make a VM migrate to an appointed host, in most circumstances, the migration is automatically started by strategies such as load balancing and server consolidation.

Steps 2: Transfer memory. Since the whole execution state of the VM is stored in memory, sending the VM's memory to the destination node ensures continuity of the service provided by the VM. All of the memory data is transferred in the first round, and then the migration controller recopies the memory data which is changed in the last round. These steps keep iterating until the dirty portion of the memory is small enough to handle the final copy. Although precopying memory is performed iteratively, the execution of programs is not obviously interrupted.

Step 3: Suspend the VM and copy the last portion of the data. The migrating VM's execution is suspended when the last round's memory data is transferred. Other nonmemory data such as CPU and network states should be sent as well. During this step, the VM is stopped and its applications will no longer run. This "service

unavailable" time is called the "downtime" of migration, which should be as short as possible so that it can be negligible to users.

Steps 4 and 5: Commit and activate the new host. After all the needed data is copied, on the destination host, the VM reloads the states and recovers the execution of programs in it, and the service provided by this VM continues. Then the network connection is redirected to the new VM and the dependency to the source host is cleared. The whole migration process finishes by removing the original VM from the source host.

CHAPTER THREE

Cloud Platform Architecture

3.1 Types of Cloud Computing Service Models

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. According to a recent survey, the percentage of companies using a public cloud is expected to rise to 51% in 2016. It is estimated that servers shipping to a public cloud will grow at 60% CAGR (Compound Growth Annual Rate), while on-site server spending will be reduced by 8.6% over the next two years.

Although cloud computing has evolved over the time it has been majorly divided into three broad service categories: **Infrastructure as a Service(IAAS)**, **Platform as a Service (PAAS)** and **Software as a Service (SAAS)** which are broadly discussed below:

Infrastructure as a Service (IAAS):

Infrastructure as a Service (IAAS) is a form of cloud computing that provides virtualized computing resources over the internet. In a IAAS model, a third party provider hosts hardware, software, servers, storage and other infrastructure components on the behalf of its users. IAAS providers also host users' applications and handle tasks including system maintenance backup and resiliency planning.

IAAS platforms offer highly scalable resources that can be adjusted on-demand which makes it a well- suited for workloads that are temporary, experimental or change unexpectedly. Other characteristics of IAAS environments include the

automation of administrative tasks, dynamic scaling, desktop virtualization and policy based services. Other characteristics of IAAS include the automation of administrative tasks, dynamic scaling, desktop virtualization and policy based services.

Technically, the IaaS market has a relatively low barrier of entry, but it may require substantial financial investment in order to build and support the cloud infrastructure. Mature open-source cloud management frameworks like OpenStack are available to everyone, and provide strong a software foundation for companies that want to build their private cloud or become a public cloud provider.

IAAS- Network:There are two major network services offered by public cloud service providers: load balancing and DNS (domain name systems). Load balancing provides a single point of access to multiple servers that run behind it. A load balancer is a network device that distributes network traffic among servers using specific load balancing algorithms. DNS is a hierarchical naming system for computers, or any other naming devices that use IP addressing for network identification – a DNS system associates domain names with IP addresses

Platform as a Service (PAAS):

Platform as a Service (PAAS) is a cloud computing model that delivers applications over the internet. In a PAAS model, a cloud provider delivers hardware and software tolls, usually those needed for application development, to its users as a service. A PAAS provider hosts the hardware and software on its own infrastructure. As a result, PAAS frees users from having to install in-house hardware and software to develop or run a new application.

PAAS doesn't replace a business' entire infrastructure but instead a business relies on PAAS providers for key services, such as Java development or application hosting. A PAAS provider, however, supports all the underlying computing and software; users only need to login and start using the platform-usually through a Web browser interface. PAAS providers then charge for that access on a per-use basis or on monthly basis.

Some of the main characteristics of PAAS are :

- Scalability and auto-provisioning of the underlying infrastructure.
- Security and redundancy.

- Build and deployment tools for rapid application management and deployment.
- Integration with other infrastructure components such as web services, databases, and LDAP.
- Multi-tenancy, platform service that can be used by many concurrent users.
- Logging, reporting, and code instrumentation.
- Management interfaces and/or API.

Software as a Service (SAAS):

Software as a Service(SAAS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. SAAS has become increasingly prevalent delivery model as underlying technologies that support Web services and service- oriented architecture (SOA) mature and new development approaches, such as Ajax, become popular. SAAS is closely related to the ASP (Application service provider) and on demand computing software delivery models. IDC identifies two slightly different delivery models for SAAS namely the hosted application model and the software development model.

Some of the core benefits of using SAAS model are:

- Easier administration.
- Automatic updates and patch management.
- Compatibility: all users will have the same version of software.
- Easier collaboration, for the same reason.
- Global accessibility.

Figure 3.1 Cloud Computing Service Models

Some of the other service categories which are more commonly classified as **Anything as a Service** (**XAAS**) are:

Storage as a Service (SAAS):

Storage as a Service is a business model in which a large company rents space in their storage infrastructure to a smaller company or individual. The economy of scale in the service provider's infrastructure theoretically allows them to provide storage much more cost effectively than most individuals or corporations can provide their own storage, when total cost of ownership is considered. Storage as a Service is generally seen as a good alternative for a small or mid-sized business that lacks the capital budget and/or technical personnel to implement and maintain their own storage infrastructure.

Communications as a Service (CAAS):

Communications as a Service (CAAS) is an outsourced enterprise communications solution that can be leased from a single vendor. Such communications can include voice over IP (VoIP or Internet telephony), instant messaging (IM), collaboration and

video conference applications using fixed and mobile devices. The CAAS vendor is responsible for all hardware and software management and offers guaranteed Quality of Service (QoS). CAAS allows businesses to selectively deploy communications devices and modes on a pay-as-you-go, as-needed basis.

Network as a Service (NAAS):

Network as a Service (NAAS), a framework that integrates current cloud computing offerings with direct, yet secure, client access to the network infrastructure. NAAS is a new cloud computing model in which the clients have access to additional computing resources collocated with switches and routers. NAAS can include flexible and extended Virtual Private Network (VPN), bandwidth on demand, custom routing, multicast protocols, security firewall, intrusion detection and prevention, Wide Area Network (WAN), content monitoring and filtering, and antivirus.

Monitoring as a Service (MAAS):

Monitoring-as-a-service (MAAS) is a framework that facilitates the deployment of monitoring functionalities for various other services and applications within the cloud. The most common application for MAAS is online state monitoring, which continuously tracks certain states of applications, networks, systems, instances or any element that may be deployable within the cloud. MAAS makes it easier for users to deploy state monitoring at different levels of Cloud services.

3.2 Types of Clouds

Cloud computing is usually described in one of two ways. Either based on the cloud location, or on the service that the cloud is offering.

Based on a cloud location, we can classify cloud as:

- Public
- Private
- Hybrid
- Community cloud

Based on a service that the cloud is offering, we are speaking of either:

- IaaS (Infrastructure-as-a-Service)
- PaaS (Platform-as-a-Service)

- SaaS (Software-as-a-Service)
- Storage, Database, Information, Process, Application, Integration, Security, Management, Testing- as-a-service

Previously, we have explained how cloud works. Basically, programs that are needed to run a certain application are now more popularly located on a remote machine, owned by another company. This is done in order not to lose on the quality performance due to processing power of your own computer, to save money on IT support, and yet remain advantageous on the market. These computers that run the applications, store the data, and use a server system, are basically what we call "the cloud".

Where Do I Pull the Switch: Cloud Location

When we talk about **public cloud**, we mean that the whole computing infrastructure is located on the premises of a cloud computing company that offers the cloud service. The location remains, thus, separate from the customer and he has no physical control over the infrastructure.

As public clouds use shared resources, they do excel mostly in performance, but are also most vulnerable to various attacks.

Private cloud means using a cloud infrastructure (network) solely by one customer/organization. It is not shared with others, yet it is remotely located. If the cloud is externally hosted. The companies have an option of choosing an on-premise private cloud as well, which is more expensive, but they do have a physical control over the infrastructure.

The security and control level is highest while using a private network. Yet, the cost reduction can be minimal, if the company needs to invest in an on-premise cloud infrastructure.

Hybrid cloud, of course, means, using both private and public clouds, depending on their purpose.

For example, public cloud can be used to interact with customers, while keeping their data secured through a private cloud.

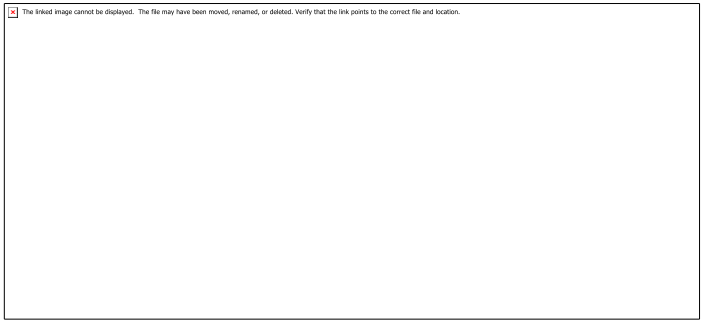


Figure 3.2 Public and Private Clouds

Community cloud implies an infrastructure that is shared between organizations, usually with the shared data and data management concerns. For example, a community cloud can belong to a government of a single country. Community clouds can be located both on and off the premises.

What Can I Do With It: Cloud Service

The most popular services of the cloud are that of either **infrastructure**, **platform**, **software**, or **storage**. As explained before, the most common cloud service is that one offering data storage disks and virtual servers, i.e. infrastructure. Examples of Infrastructure-as-a-Service (IaaS) companies are Amazon, Rackspace, Flexiscale.

If the cloud offers a development platform, and this includes operating system, programming language execution environment, database, and web server, the model is known as Platform-as-a-Service (PaaS), examples of which are Google App Engine, Microsoft Azure, Salesforce. Operating system can be frequently upgraded and developed with PaaS, services can be obtained from diverse sources, and programming can be worked in teams (geographically distributed).

Software-as-a-Service (SaaS), finally, means that users can access various software applications on a pay- per-use basis. As opposed to buying licensed programs, often very expensive. Examples of such services include widely used GMail, or Google Docs.

3.3 Google cloud platform

This overview covers the following types of services:

- Computing and Computing and hosting services.

- Storage Services
- Networking Services
- Big Data Services

Computing and hosting services:

Cloud Platform gives you options for computing and hosting. You can choose to work with a managed application platform, leverage container technologies to gain lots of flexibility, or build your own cloud- based infrastructure to have the most control and flexibility. You can imagine a spectrum where, at one end, you have most of the responsibilities for resource management and, at the other end, Google has most of those responsibilities:

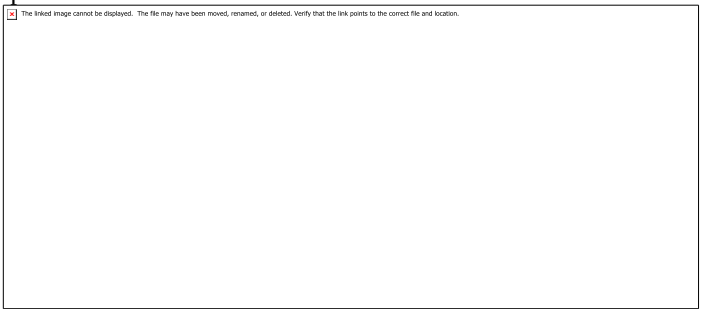


Figure 3.3 Google Cloud Platform

Application platform:

Google App Engine is Cloud Platform's *platform as a service* (PaaS). With App Engine, Google handles most of the management of the resources for you. For example, if your application requires more computing resources because traffic to your website increases, Google automatically scales the system to provide those resources. If the system software needs a security update, that's handled for you, too.

When you build your app on App Engine, you can:

- Build your app on top of the App Engine standard environment runtimes in the languages that the standard environment supports, including: Pyhton 2.7,Java 7,PHP.

- Build your app on top of the App Engine flexible environment runtimes in the languages that App Engine flexible supports, including: Python 2.7/3.4, Java 8, Go, Node.js, and Ruby. Or use custom runtimes to use an alternative implementation of a supported language or any other language.
- Let Google manage app hosting, scaling, monitoring and infrastructure for you.
- Use the App Engine SDK to develop and test on your local machine in an environment that simulates App Engine on Cloud Platform.
- Easily use the storage technologies that App Engine is designed to support in the standard and flexible environments.

Google Cloud SQL is your SQL database, supporting either MySQL or PostgreSQL. App Engine Datastore is your schemaless, NoSQL datastore. Google Cloud Storage provides space for your large files.

In the standard environment, you can also choose from a variety of third-party databases to use with your applications such as Redis, MongoDB, Cassandra, and Hadoop.

In the flexible environment, you can easily use any third-party database supported by your language, if the database is accessible from the Google App Engine instance.

In either environment, these third-party databases can be hosted on Compute Engine, hosted on another cloud provider, hosted on-premises, or managed by a third-party vendor.

- Use Cloud Endpoints in the standard environment to generate APIs and client libraries that you can use to simplify data access from other applications. Endpoints makes it easier to create a web backend for web clients and mobile clients, such as Android or iOS.
- Use built-in, managed services for activities such as email and user management.
- Use Cloud Security Scanner to identify security vulnerabilities as a complement to your existing secure design and development processes.
- Deploy your app by using the App Engine launcher GUI application on Mac OS X or Microsoft Windows or by using the command line.

- For the standard environment, run your app from the Central US or Western Europe regions. For a complete list and description of App Engine's features, see the App Engine documentation.

Containers:

With container-based computing, you can focus on your application code, instead of on deployments and integration into hosting environments. Google Kubernetes Engine is built on the open source Kubernetes system, which gives you the flexibility of on-premises or hybrid clouds, in addition to Cloud Platform's public cloud infrastructure.

When you build with Kubernetes Engine, you can:

- Create and manage groups of Compute Engine instances running Kubernetes, called *clusters*. Kubernetes Engine uses Compute Engine instances as *nodes* in a cluster. Each node runs the Docker runtime, a Kubeletagent that monitors the health of the node, and a simple network proxy.
- Declare the requirements for your Docker containers by creating a simple JSON configuration file.
- Use Google Container Registry for secure, private storage of Docker images. You can push images to your registry and then you can pull images to any Compute Engine instance or your own hardware by using an HTTP endpoint.
- Create single- and multi-container *pods*. Each pod represents a logical host that can contain one or more containers. Containers in a pod work together by sharing resources, such as networking resources. Together, a set of pods might comprise an entire application, a micro-service, or one layer in a multi-tier application.
- Create and manage *replication controllers*, which manage the creation and deletion of pod replicas based on a template. Replication controllers help to ensure that your application has the resources it needs to run reliably and scale appropriately.
- Create and manage *services*. Services create an abstraction layer that decouples frontend clients from pods that provide backend functions. In this way, clients can work without concerns about which pods are being created and deleted at any given moment.
- Create an external network load balancer.

Virtual machines:

Cloud Platform's unmanaged compute service is Google Compute Engine. You can think of Compute Engine as providing an *infrastructure as a service* (IaaS), because the system provides a robust computing infrastructure, but you must choose and configure the platform components that you want to use. With

Compute Engine, it's your responsibility to configure, administer, and monitor the systems. Google will ensure that resources are available, reliable, and ready for you to use, but it's up to you to provision and manage them. The advantage, here, is that you have complete control of the systems and unlimited flexibility.

When you build on Compute Engine, you can:

- Use virtual machines (VMs), called *instances*, to build your application, much like you would if you had your own hardware infrastructure. You can choose from a variety of instance types to customize your configuration to meet your needs and your budget.
- Choose which global regions and zones to deploy your resources in, giving you control over where your data is stored and used.
- Choose which operating systems, development stacks, languages, frameworks, services, and other software technologies you prefer.
- Create instances from public or private images .
- Use Cloud Platform storage technologies or any third-party technologies you prefer.
- Use Google Cloud Launcher to quickly deploy pre-configured software packages. For example, you can deploy a LAMP or MEAN stack with just a few clicks.
- Create instance groups to more easily manage multiple instances together.
- Use autoscaling with an instance group to automatically add and remove capacity.
- Attach and detach disks as needed.
- Use SSH to connect directly to your instances.

Combining computing and hosting options:

You don't have to stick with just one type of computing service. For example, you can combine App Engine and Compute Engine to take advantage of the features and benefits of each.

Storage services:

Whatever your application, you'll probably need to store some data. Cloud Platform provides a variety of storage services, including:

- A SQL database in Cloud SQL, which provides either MySQL or PostgreSQL databases.
- A fully managed, mission-critical, relational database service in Cloud Spanner that offers transactional consistency at global scale, schemas, SQL querying, and automatic, synchronous replication for high availability.
- Two options for NoSQL data storage: Cloud Datastore and Cloud Bigtable.
- Consistent, scalable, large-capacity data storage in Cloud Storage. Cloud Storage comes in several flavors:
- Multi-Regional provides maximum availability and geo-redundancy.
- Regional provides high availability and a localized storage location.
- Nearline provides low-cost archival storage ideal for data accessed less than once a month.
- Coldline provides the lowest-cost archival storage for backup and disaster recovery.
- Persistent disks on Compute Engine, for use as primary storage for your instances. Compute Engine offers both hard-disk-based persistent disks, called *standard persistent disks*, and solid-state persistent disks (SSD).

Networking services:

While App Engine manages networking for you, and Kubernetes Engine uses the Kubernetes model, Compute Engine provides a set of networking services. These services help you to load-balance traffic across resources, create DNS records, and connect your existing network to Google's network.

Networks, firewalls, and routes:

Compute Engine provides a set of networking services that your VM instances use. Each instance can be attached to only one network. Every Compute Engine project

has a *default network*. You can create additional networks in your project, but networks cannot be shared between projects.

Firewall rules govern traffic coming into instances on a network. The default network has a default set of firewall rules, and you can create custom rules, too.

A *route* lets you implement more advanced networking functions in your instances, such as creating VPNs. A route specifies how packets leaving an instance should be directed. For example, a route might specify that packets destined for a particular network range should be handled by a gateway virtual machine instance that you configure and operate.

Load balancing:

If your website or application is running on Compute Engine, the time might come when you're ready to distribute the workload across multiple instances. Compute Engine's server-side load balancing features provide you with the following options:

- Network load balancing lets you distribute traffic among server instances in the same region based on incoming IP protocol data, such as address, port, and protocol. Network load balancing is a great solution if, for example, you want to meet the demands of increasing traffic to your website.

- HTTP/HTTPS load balancing enables you to distribute traffic across regions so you can ensure that requests are routed to the closest region or, in the event of a failure or over-capacity limitations, to a healthy instance in the next closest region. You can also use HTTP/HTTPS load balancing to distribute traffic based on content type. For example, you might set up your servers to deliver static content, such as images and CSS, from one server and dynamic content, such as PHP pages, from a different server. The load balancer can direct each request to the server that provides each content type.

Cloud DNS:

You can publish and maintain Domain Name System (DNS) records by using the same infrastructure that Google uses. You can use the Google Cloud Platform Console, the command line, or a REST API to work with managed zones and DNS records.

Advanced connectivity:

If you have an existing network that you want to connect to Cloud Platform resources, Google Cloud Interconnect offers three options for advanced connectivity:

- Carrier Interconnect enables you to connect your infrastructure to Google's network edge through highly available, lower-latency connections by using service providers. You can also extend your private network into your private Compute Engine network over Carrier Interconnect links by using a VPN tunnel between the networks.
- You can establish a direct peering connection between your business network and Google's. With this connection, you can exchange Internet traffic between your network and Google's at one of Google's broad-reaching Edge network locations. Visit Google's peering site to find out more information about edge locations.
- Cloud VPN lets you connect your existing network to your Compute Engine network via an IPsec connection. You can use VPN to connect two Compute Engine VPN gateways to each other.

Big data services:

Big data services enable you to process and query big data in the cloud to get fast answers to complicated questions.

Data analysis:

BigQuery provides data analysis services. With BigQuery, you can:

- Create custom schemas that organize your data into datasets and tables.
- Load data from a variety of sources, including streaming data.
- Use SQL-like commands to query massive datasets very quickly. BigQuery is designed and optimized for speed.
- Use the web UI, command-line interface, or API.
- Load, query, export, and copy data by using jobs.
- Manage data and protect it by using permissions.

Batch and streaming data processing:

Cloud Dataflow provides a managed service and set of SDKs that you can use to perform batch and streaming data processing tasks. Dataflow works well for high-volume computation, especially when the processing tasks can clearly and easily be divided into parallel workloads. Dataflow is also great for extract- transform-load (ETL) tasks, which are useful for moving data between different storage media, transforming data into a more desirable format, or loading data onto a new storage system.

3.4 Amazon cloud computing infrastructure

Data storage in clouds is very popular and widely used in the modern world. The Amazon is one of the companies which provide this service. Amazon Web Services offers the inexpensive and reliable cloud computing services, that's why many large companies prefer the Amazon Cloud for storage and operating their data. It is convenient to draw various AWS diagrams explaining the use of amazon cloud with help of tools of AWS Architecture Diagrams Solution from the Computer and Networks Area of ConceptDraw.

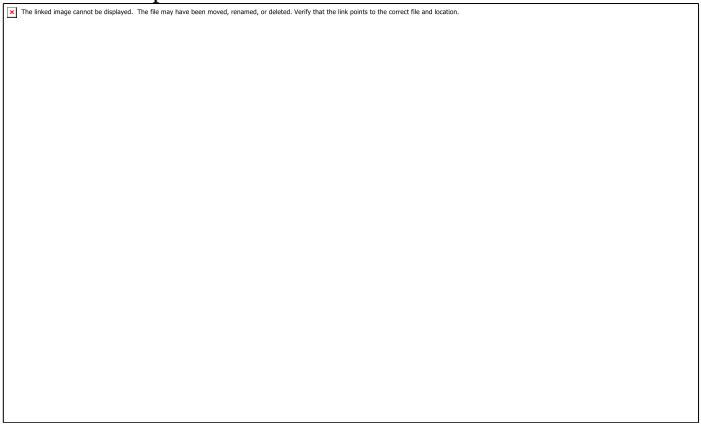


Figure 3.4 Amazon Web Services(AWS)

AWS:

The AWS diagrams are convenient way for explaining the work of Amazon Web Services. ConceptDraw PRO diagramming and vector drawing software offers the AWS Architecture Diagrams Solution from the Computer and Networks Area for fast and easy creating the AWS diagrams of any complexity.

In 2006, Amazon Web Services (AWS) started to offer IT services to the market in the form of web services, which is nowadays known as cloud computing. With this cloud, we need not plan for servers and other IT infrastructure which takes up much of time in advance. Instead, these services can instantly spin up hundreds or thousands of servers in minutes and deliver results faster. We pay only for what we use with no up-front expenses and no long-term commitments, which makes AWS cost efficient.

Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers multitude of businesses in 190 countries around the world.

What is Cloud Computing?

Cloud computing is an internet-based computing service in which large groups of remote servers are networked to allow centralized data storage, and online access to computer services or resources.

Using cloud computing, organizations can use shared computing and storage resources rather than building, operating, and improving infrastructure on their own.

Cloud computing is a model that enables the following features.

Users can provision and release resources on-demand.

Resources can be scaled up or down automatically, depending on the load.

Resources are accessible over a network with proper security.

Cloud service providers can enable a pay-as-you-go model, where customers are charged based on the type of resources and per usage.

Types of Clouds:

There are three types of clouds − Public, Private, and Hybrid cloud.

Public Cloud:

In public cloud, the third-party service providers make resources and services available to their customers via Internet. Customer's data and related security is with the service providers' owned infrastructure.

Private Cloud:

A private cloud also provides almost similar features as public cloud, but the data and services are managed by the organization or by the third party only for the customer's organization. In this type of cloud, major control is over the infrastructure so security related issues are minimized.

Hybrid Cloud:

A hybrid cloud is the combination of both private and public cloud. The decision to run on private or public cloud usually depends on various parameters like sensitivity of data and applications, industry certifications and required standards, regulations, etc.

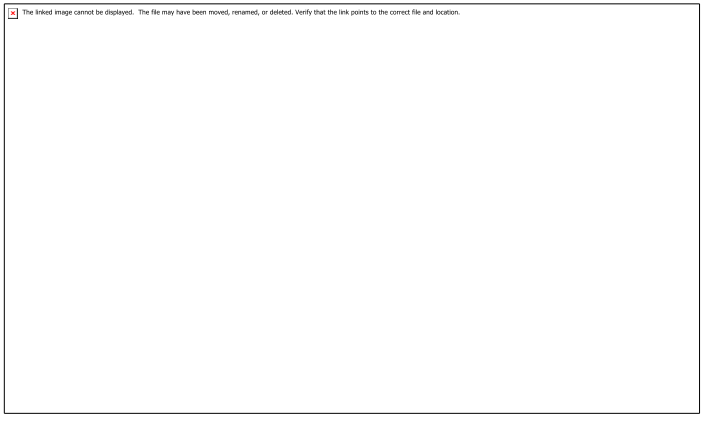


Figure 3.5 Amazon Web Services

Cloud Service Models:

There are three types of service models in cloud − IaaS, PaaS, and SaaS.

IaaS:

IaaS stands for Infrastructure as a Service. It provides users with the capability to provision processing, storage, and network connectivity on demand. Using this service model, the customers can develop their own applications on these resources.

PaaS:

PaaS stands for Platform as a Service. Here, the service provider provides various services like databases, queues, workflow engines, e-mails, etc. to their customers. The customer can then use these components for building their own applications. The services, availability of resources and data backup are handled by the service provider that helps the customers to focus more on their application's functionality.

SaaS:

SaaS stands for Software as a Service. As the name suggests, here the third-party providers provide end-user applications to their customers with some administrative capability at the application level, such as the ability to create and manage their users.

Also some level of customizability is possible such as the customers can use their own corporate logos, colors, etc.

Advantages of Cloud Computing:

Here is a list of some of the most important advantages that Cloud Computing has to offer −

Cost-Efficient − Building our own servers and tools is time-consuming as well as expensive as we need to order, pay for, install, and configure expensive hardware, long before we need it. However, using cloud computing, we only pay for the amount we use and when we use the computing resources. In this manner, cloud computing is cost efficient.

Reliability − A cloud computing platform provides much more managed, reliable and consistent service than an in-house IT infrastructure. It guarantees 24x7 and 365 days of service. If any of the server fails, then hosted applications and services can easily be transited to any of the available servers.

Unlimited Storage − Cloud computing provides almost unlimited storage capacity, i.e., we need not worry about running out of storage space or increasing our current storage space availability. We can access as much or as little as we need.

Backup & Recovery − Storing data in the cloud, backing it up and restoring the same is relatively easier than storing it on a physical device. The cloud service providers also have enough technology to recover our data, so there is the convenience of recovering our data anytime.

Easy Access to Information − Once you register yourself in cloud, you can access your account from anywhere in the world provided there is internet connection at that point. There are various storage and security facilities that vary with the account type chosen.

Disadvantages of Cloud Computing:

Although Cloud Computing provides a wonderful set of advantages, it has some drawbacks as well that often raise questions about its efficiency.

Security issues:

Security is the major issue in cloud computing. The cloud service providers implement the best security standards and industry certifications, however, storing data and important files on external service providers always bears a risk.

AWS cloud infrastructure is designed to be the most flexible and secured cloud network. It provides scalable and highly reliable platform that enables customers to deploy applications and data quickly and securely.

Technical issues:

As cloud service providers offer services to number of clients each day, sometimes the system can have some serious issues leading to business processes temporarily being suspended. Additionally, if the internet connection is offline then we will not be able to access any of the applications, server, or data from the cloud.

Not easy to switch service providers:

Cloud service providers promises vendors that the cloud will be flexible to use and integrate, however switching cloud services is not easy. Most organizations may find it difficult to host and integrate current cloud applications on another platform. Interoperability and support issues may arise such as applications developed on Linux platform may not work properly on Microsoft Development Framework (.Net).

3.5 Microsoft Azure

Microsoft Azure, formerly known as Windows Azure, is Microsoft's public cloud computing platform. It provides a range of cloud services, including those for compute, analytics, storage and networking. Users can pick and choose from these services to develop and scale new applications, or run existing applications, in the public cloud.

Microsoft Azure, formerly known as Windows Azure, is Microsoft's public cloud computing platform. It provides a range of cloud services, including those for compute, analytics, storage and networking. Users can pick and choose from these services to develop and scale new applications, or run existing applications, in the public cloud.

Microsoft Azure is widely considered both a Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) offering.

Microsoft categorizes Azure services into 11 main product types:

1. **Compute** – these services provide virtual machines, containers, batch processing and remote application access.

2. **Web and mobile** – these services support the development and deployment of web and mobile applications, and also offer features for API management, notification and reporting.

3. **Data storage** – this category includes Database as a Service offerings for SQL and NoSQL, as well as unstructured and cached cloud storage.
4. **Analytics** – these services provide distributed analytics and storage, as well as real-time analytics, big data analytics, data lakes, machine learning and data warehousing.
5. **Networking** – this group includes virtual networks, dedicated connections and gateways, as well as services for traffic management, load balancing and domain name system (DNS) hosting.

6. **Media and content delivery network (CDN)** – these services include on-demand streaming, encoding and media playback and indexing.
7. **Hybrid integration** – these are services for server backup, site recovery and connecting private and public clouds.
8. **Identity and access management (IAM)** – these offerings ensure only authorized users can employ Azure services, and help protect encryption keys and other confidential information.
9. **Internet of Things (IoT)** – these services help users capture, monitor and analyze IoTdata from sensors and other devices.
10. **Development** – these services help application developers share code, test applications and track potential issues. Azure support a range of application programming languages, including JavaScript, Python, .NET and Node.js.
11. **Management and security** – these products help cloud administrators manage their Azure deployment, schedule and run jobs, and create automation. This product group also includes capabilities for identifying and responding to cloud security threats.

The full list of Azure services is constantly subject to change. Users should check the Microsoft Azure website for updates.Just as they can with other public cloud platforms, some organizations use Azure for data backup and disaster recovery. In

addition, some organizations use Azure as an alternative to their own data center. Rather than investing in local servers and storage, these organizations choose to run some, or all, of their business applications in Azure.

Microsoft introduced Azure in October 2008. The cloud platform was originally called Windows Azure, but was rebranded to Microsoft Azure in April 2014. Azure competes with other public cloud platforms, including Amazon Web Services (AWS) and Google Cloud Platform.

To ensure availability, Microsoft has Azure data centers located around the world. As of January 2016, Microsoft said Azure services are available in 22 regions across the globe, including in the United States, Europe, Asia, Australia and Brazil.As with other public cloud providers, Azure primarily uses a pay-as-you- go pricing modelthat charges based on usage. However, a single application may use multiple Azure services, so users should review and manage usage to minimize costs.

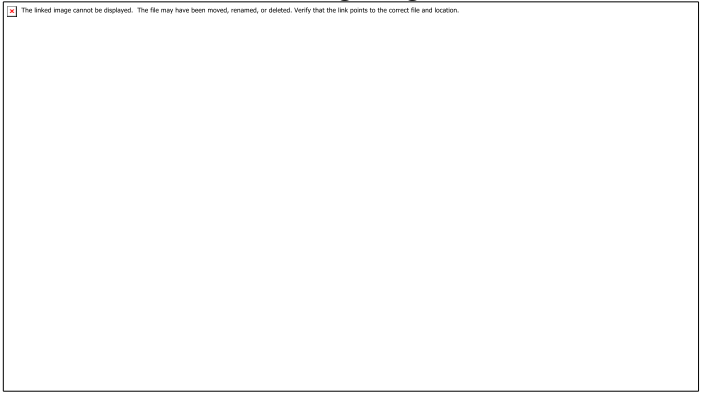


Figure 3.6 Microsoft Azure

Azure is productive for developers:

Get your apps to market faster. Azure integrated tools, from mobile DevOps to serverless computing support your productivity. Build the way you want to, using the tools and open source technologies you already know. Azure supports a range of operating systems, programming languages, frameworks, databases and devices.

- Continuously innovate and deliver high-quality apps.

- Provide cross-device experiences with support for all major mobile platforms.
- Run any stack, Linux-based or Windows-based and use advanced capabilities such as Kubernetes cluster in Azure Container Service.

Azure is the only consistent hybrid cloud:

Build and deploy wherever you want with Azure, the only consistent hybrid cloud on the market. Connect data and apps in the cloud and on-premises—for maximum portability and value from your existing investments. Azure offers hybrid consistency in application development, management and security, identity management and across the data platform.

- Extend Azure on-premises and build innovative, hybrid apps with Azure Stack.
- Connect on-premises data and apps to overcome complexity and optimise your existing assets.
- Distribute and analyse data seamlessly across cloud and on-premises. Azure is the cloud for building intelligent apps

Use Azure to create data-driven, intelligent apps. From image recognition to bot services, take advantage of Azure data services and artificial intelligence to create new experiences—that scale—and support deep learning, HPC simulations and real-time analytics on any shape and size of data.

- Develop breakthrough apps with built-in AI.
- Build and deploy custom AI models at scale, on any data.
- Combine the best of Microsoft and open source data and AI innovations.

CHAPTER FOUR

Cloud Programming and Software Environments

4.1 Features of Cloud and Grid Platforms

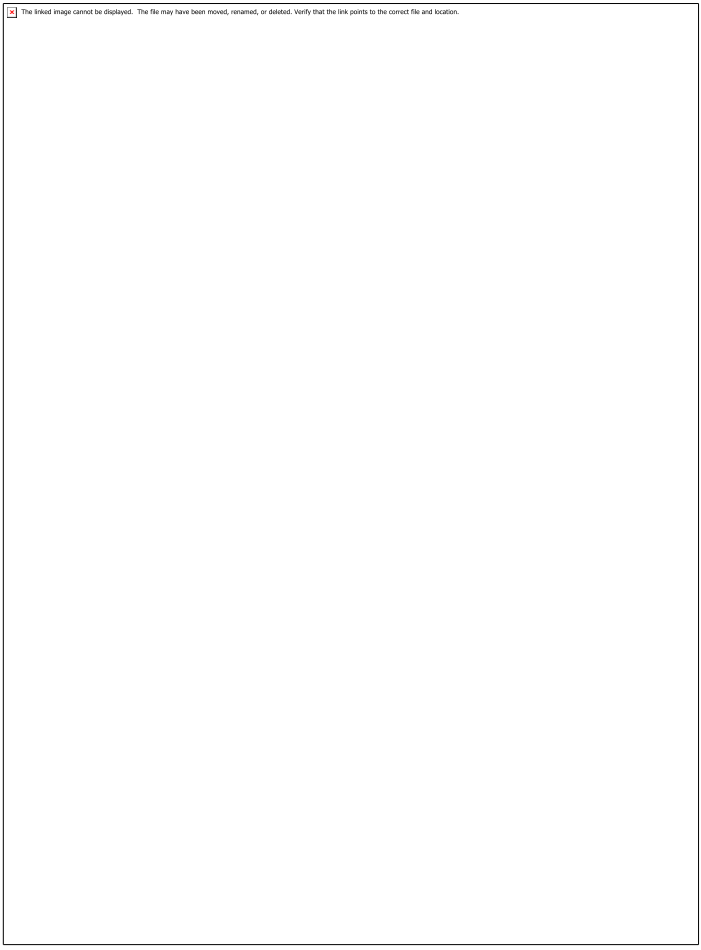
In this section, we summarize important features in real cloud and grid platforms. In four tables, we cover the capabilities, traditional features, data features, and features for programmers and runtime systems to use. The entries in these tables are source references for anyone who wants to program the cloud efficiently.

Cloud Capabilities and Platform Features:

The capabilities offer cost-effective utility computing with the elasticity to scale up and down in power. However, as well as this key distinguishing feature, commercial clouds offer a growing number of additional capabilities commonly termed "Platform as a Service" (PaaS). For Azure, current platform features include Azure Table, queues, blobs, Database SQL, and web and Worker roles. Amazon is often viewed as offering "just" Infrastructure as a Service (IaaS), but it continues to add platform features including SimpleDB (similar to Azure Table), queues, notification, monitoring, content delivery network, relational database, and MapReduce (Hadoop). Google does not currently offer a broad-based cloud service, but the Google App Engine (GAE) offers a powerful web application development environment.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

Important Cloud Platform Capabilities



Infrastructure Cloud Features

Grid Computing Vs Cloud Computing:

i. Types and the Division

After the evolution, the cloud computing deployment has been changed into public clouds, private clouds, community clouds, and hybrid clouds.

However, grid computing has a distributed computing system, a distributed information system, and distributed pervasive systems.

ii. Main Focus and Motto

The main motto of cloud computing is to provide the service at a lower rate and increase returns. It also provides flexibility and scalability so that the user can easily use cloud computing with increased availability and security.

However, grid computing basically focuses on networks to solve some complex problems and has a large-scale goal. Grid computing also delivers a computer as a utility.

iii. Use and Security

There is a large amount of data stored in the cloud so it provides security according to it. Data store in the cloud is secured and can access with the help of credentials only.

Grid computing deals with Idol energy in computers and mostly use for something sensible.

iv. Basis of Dependency

Cloud Computing is totally dependent on the internet through the data center. The cloud provides maximum security along with maximum performance.

Grid computing works even if a computer stops or failure occurs. The other computer will pick up the work making the system more reliable and efficient.

v. Space and Storage

It is easy in the cloud to backup and restores the data as it has fast data processors. The new updates in cloud computing are efficient and automatic.

In a grid, computing space is saved and access to additional resources can be done.

vi. Difference and Similarity

Cloud computing and grid computing are network-based technologies which have the same characteristics. They are different from each other in a few terms such as architecture, business model, and interoperability.

vii. Remote Usage

In cloud computing, the computing resources manage within a single location which locates at a different place.However, in grid computing, there is a distributed system where the resources are distributed at different locations and can be located from different sites.

viii. Resource Requirement

Grid computing involves more amount of resources and other than computers in networks. Cloud computing doesn't access resources directly, it gets from the internet.

ix. Problem Solving Techniques

Grid computing uses all kinds of computing resources for job scheduling. We divide a big task into multiple tasks which we can solve by multiple computers as the work assigns to a particular computer.

Cloud Computing has resources which are pooling through grouping resources and requires the basis from the cluster of servers.

x. Services and Capabilities

Cloud computing is nothing but whole internet-based computing. There are lots of services provided by the cloud such as management of data, data security, job queries, etc. It eliminates the cost of buying new hardware and software which are necessary to build applications.

Whereas, grid computing generally use by academic researchers and is capable to handle large sets of limited job which are complex and involves a large volume of data.

xi. Terminology

Cloud computing and grid computing share similar characteristics such as resource pooling. They both are network computing technologies which are different in terms of architecture, business model, etc.

Grid computing is nothing but a collection of resources from various locations to resolve a single task. However, cloud computing is a form of computer-based virtualizes resources which are situated at different locations in the cluster.

xii. Research

In grid computing, the resources are provided as the utility with the grid as a computing platform. These will group together in the virtual organization with many user communities which will be able to resolve problems over the internet.

On the other hand, Cloud Computing involves a common group of system administrator which will be able to perform the complete management.

xiii. Interoperability

Grid computing can handle interoperability easily whereas cloud computing does not support interoperability and can result in vendor lock-in, which makes it difficult to transfer from one cloud service provider to another.

4.2 ***Programming Support of Google App Engine***

GAE programming model for two supported languages: Java and Python. A client environment includes an Eclipse plug-in for Java allows you to debug your GAE on your local machine. Google Web Toolkit is available for Java web application developers. Python is used with frameworks such as Django and CherryPy, but Google also has webapp Python environment.

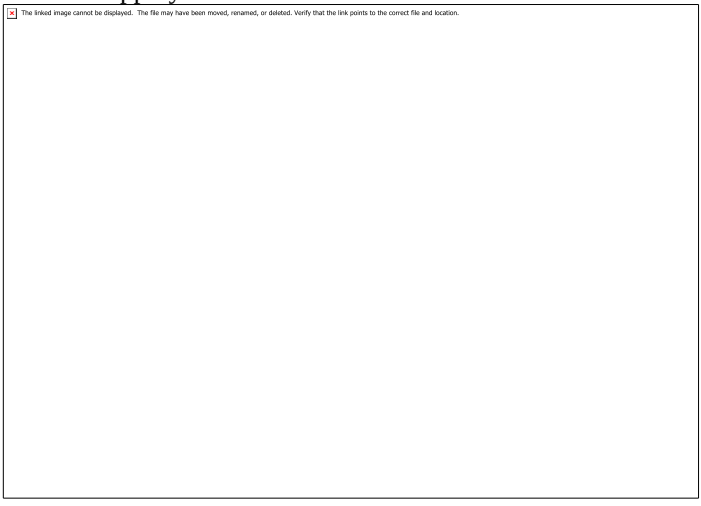


Figure 4.1 Google App Engine

There are several powerful constructs for storing and accessing data. The data store is a NOSQL data management system for entities. Java offers Java Data Object (JDO) and Java Persistence API (JPA) interfaces implemented by the Data Nucleus Access platform, while Python has a SQL-like query language called GQL. The performance of the data store can be enhanced by in-memory caching using the memcache, which can also be used independently of the data store.

Recently, Google added the blobstore which is suitable for large files as its size limit is 2 GB. There are several mechanisms for incorporating external resources. The Google SDC Secure Data Connection can tunnel through the Internet and link your intranet to an external GAE application. The URL Fetch operation provides the ability for applications to fetch resources and communicate with other hosts over the Internet using HTTP and HTTPS requests.

An application can use Google Accounts for user authentication. Google Accounts handles user account creation and sign-in, and a user that already has a Google account (such as a Gmail account) can use that account with your app. GAE provides the ability to manipulate image data using a dedicated Images service which can resize, rotate, flip, crop, and enhance images. A GAE application is configured to consume resources up to certain limits or quotas. With quotas, GAE ensures that your application won't exceed your budget, and that other applications running on GAE won't impact the performance of your app. In particular, GAE use is free up to certain quotas.

Google File System (GFS)

GFS is a fundamental storage service for Google's search engine. GFS was designed for Google applications, and Google applications were built for GFS. There are several concerns in GFS. rate). As servers are composed of inexpensive commodity components, it is the norm rather than the exception that concurrent failures will occur all the time. Another concerns the file size in GFS. GFS typically will hold a large number of huge files, each 100 MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64 MB instead of the 4 KB in typical traditional file systems. The I/O pattern in the Google application is also special. Files are typically written once, and the write operations are often the appending data blocks to the end of files. Multiple appending operations might be concurrent. The customized API can simplify the problem and focus on Google applications.

Figure shows the GFS architecture. It is quite obvious that there is a single master in the whole cluster. Other nodes act as the chunk servers for storing data, while the single master stores the metadata. The file system namespace and locking facilities are managed by the master. The master periodically communicates with the chunk servers to collect management information as well as give instructions to the chunk servers to do work such as load balancing or fail recovery.

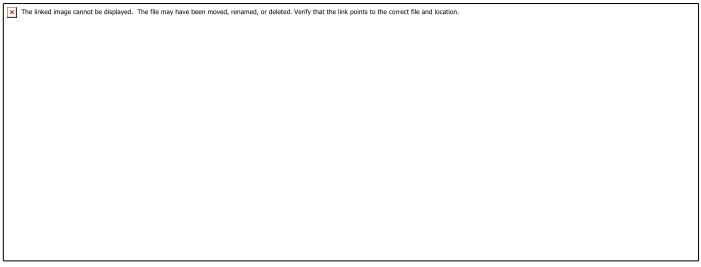


Figure 4.2 Searching record in GFS

The master has enough information to keep the whole cluster in a healthy state. Google uses a shadow master to replicate all the data on the master, and the design guarantees that all the data operations are performed directly between the client and the chunk server. The control messages are transferred between the master and the clients and they can be cached for future use. With the current qualityof commodity servers, the single master can handle a cluster of more than 1,000 nodes.

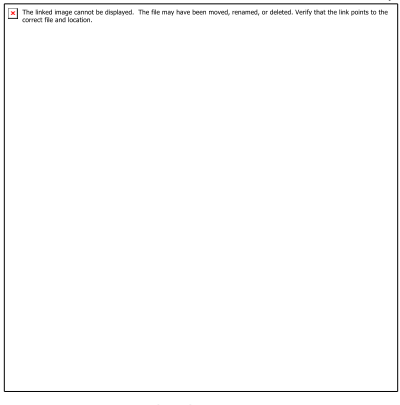


Figure 4.3 Read Write operations in GFS

The goal is to minimize involvement of the master. The mutation takes the following steps:

1. The client asks the master which chunk server holds the current lease for the chunk and the locations of the other replicas. If no one has a lease, the master grants one to a replica it chooses (not shown).

2. The master replies with the identity of the primary and the locations of the other (secondary) replicas. The client caches this data for future mutations. It needs to contact the master again only when the primary becomes unreachable or replies that it no longer holds a lease.

3. The client pushes the data to all the replicas. Each chunk server will store the data in an internal LRU buffer cache until the data is used or aged out. By decoupling the data flow from the control flow, we can improve performance by scheduling the expensive data flow based on the network topology regardless of which chunk server is the primary.

4. Once all the replicas have acknowledged receiving the data, the client sends a write request to the primary. The request identifies the data pushed earlier to all the replicas. The primary assigns consecutive serial numbers to all the mutations it receives, possibly from multiple clients, which provides the necessary serialization. It applies the mutation to its own local state in serial order.

5. The primary forwards the write request to all secondary replicas. Each secondary replica applies mutations in the same serial number order assigned by the primary.

6. The secondaries all reply to the primary indicating that they have completed the operation.

7. The primary replies to the client. Any errors encountered at any replicas are reported to the client. In case of errors, the write corrects at the primary and an arbitrary subset of the secondary replicas. The client request is considered to have failed, and the modified region is left in an inconsistent state. Our client code handles such errors by retrying the failed mutation. It will make a few attempts at steps 3 through 7 before falling back to a retry from the beginning of the write.

GFS was designed for high fault tolerance and adopted some methods to achieve this goal. Master and chunk servers can be restarted in a few seconds, and with such a fast recovery capability, the window of time in which the data is unavailable can be greatly reduced. As we mentioned before, each chunk is replicated in at least three places and can tolerate at least two data crashes for a single chunk of data. The shadow master handles the failure of the GFS master **Big Table**

BigTable was designed to provide a service for storing and retrieving structured and semistructured data. BigTable applications include storage of web pages, per-user data, and geographic locations. The database needs to support very high read/write rates and the scale might be millions of operations per second. Also, the database needs to support efficient scans over all or interesting subsets of data, as well as efficient joins of large one-to-one and one-to-many data sets. The application may need to examine data changes over time. The BigTable system is scalable, which means the system has thousands of servers, terabytes of in-memory data, petabytes of disk-based data, millions of reads/writes per second, and efficient scans. BigTable is used in many projects, including Google Search, Orkut, and Google Maps/Google Earth, among others.

The BigTable system is built on top of an existing Google cloud infrastructure. BigTable uses the following building blocks:

1. GFS: stores persistent state
2. Scheduler: schedules jobs involved in BigTable serving
3. Lock service: master election, location bootstrapping
4. MapReduce: often used to read/write BigTable data

Web Table stores the data about a web page. Each web page can be accessed by the URL. The URL is considered the row index. The column provides different data related to the corresponding URL

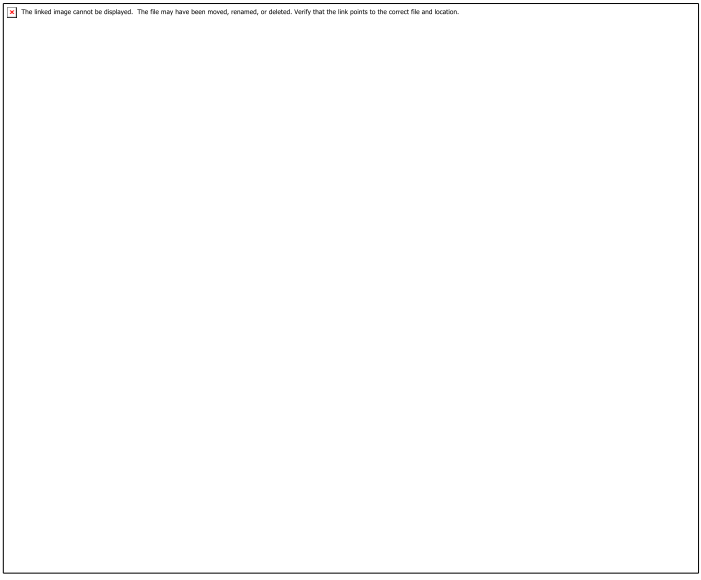


Figure 4.4 Web Table

The map is indexed by row key, column key, and timestamp—that is, (row:string, column: string, time:int64) maps to string (cell contents). Rows are ordered in lexicographic order by row key. The row range for a table is dynamically partitioned and each row range is called “Tablet”.

Syntax for columns is shown as a (family:qualifier) pair. Cells can store multiple versions of data with timestamps. A BigTable master manages and stores the metadata of the BigTable system. BigTable clients use the BigTable client programming library to communicate with the BigTable master and tablet servers. BigTable relies on a highly available and persistent distributed lock service called Chubby.

4.3 Programming on Amazon AWS

AWS platform has many features and offers many services

Features:

Relational Database Service (RDS) with a messaging interface

Elastic MapReduce capability

NOSQL support in SimpleDB

Capabilities:

Auto-scaling enables you to automatically scale your Amazon EC2 capacity up or down according to conditions.

Elastic load balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.

CloudWatch is a web service that provides monitoring for AWS cloud resources, operational performance, andoverall demand patterns—including metrics such as CPU utilization, disk reads and writes, and network traffic.

Amazon provides several types of preinstalled VMs. Instances are often called Amazon Machine Images (AMIs) which are preconfigured with operating systems based on Linux or Windows, and additional software. Figure 6.24 shows an execution environment.A MIs are the templates for instances, which are running VMs. The AMIs are formed from the virtualized compute, storage, and server resource.

Private AMI: Images created by you, which are private by default. You can grant access to other users to launch your private images.

Public AMI: Images created by users and released to the AWS community, so anyone can launch instances based on them

Figure 4.5 Amazon Machine Image(AMI)

Paid QAMI: You can create images providing specific functions that can be launched by anyone willing to pay you per each hour of usage.

Amazon Simple Storage Service (S3):

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. S3 provides the object- oriented storage service for users. Users can access their objects through Simple Object Access Protocol (SOAP) with either browsers or other client programs which support SOAP. SQS is responsible for ensuring a reliable message service between two processes.

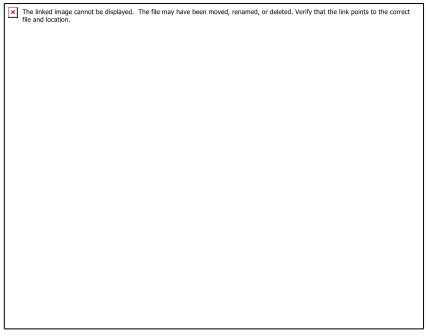


Figure 4.6 AWS S3 Service

The fundamental operation unit of S3 is called an object. Each object is stored in a bucket and retrieved via a unique, developer-assigned key. In other words, the bucket is the container of the object. Besides unique key attributes, the object has other attributes such as values, metadata, and access control information. Through the key-value programming interface, users can write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. There are two types of web service interface for the user to access the data stored in Amazon clouds. One is a REST (web 2.0) interface, and the other is a SOAP interface. Here are some key features of S3:

Redundant through geographic dispersion.

Designed to provide 99.99% durability and 99.99 %availability of objects over a given year with cheaper reduced redundancy storage (RRS).

Authentication mechanisms to ensure that data is kept secure from unauthorized access.

Objects can be made private or public, and rights can be granted to specific users.

Per-object URLs and ACLs (access control lists). Default download protocol of HTTP

Amazon Elastic Block Store (EBS) and SimpleDB:

The Elastic Block Store (EBS) provides the volume block interface for saving and restoring the virtual images of EC2 instances. The status of EC2 can now be saved in the EBS system after the machine is shut down.Users can use EBS to save persistent data and mount to the running instances of EC2. S3 is "Storage as a Service" with a messaging interface. Multiple volumes can be mounted to the same instance. These storage volumes behave like raw, unformatted block devices, with user-supplied device names and a block device interface.

Amazon SimpleDB Service:

SimpleDB provides a simplified data model based on the relational database data model. Structured data from users must be organized into domains. Each domain can be considered a table. The items are the rows in the table. A cell in the table is recognized as the value for a specific attribute (column name) of the corresponding row. it is possible to assign multiple values to a single cell in the table. This is not permitted in a traditional relational database. SimpleDB, like Azure Table, could be called "LittleTable" as they are aimed at managing small amounts of information stored in a distributed table.

4.4 Emerging Cloud Software Environments

Developments in the cloud computing industry move at a pace that can be maddening to follow and impossible to predict.But some big-picture trends that will characterize the market for the next year are coming into focus, even if the technologies that ultimately enable them and vendors that drive them seem constantly in flux and vulnerable to disruption.

Many of those emerging cloud computing trends stem from the industry entering a phase of standardization and increased compatibility—a sign of maturity in any tech sector.Cloud infrastructure—public, hosted private and on-premises—is increasingly less siloed, allowing workloads to be more portable and data streams more mobile.

That standardization, largely thanks to the open-source movement, is allowing a shift in focus up the stack, with new channel roles emerging to support application-level processes, from enabling artificial intelligence and high-performance computing, to delivering novel SaaSOps and application development services.

Multi-Cloud Becomes Omni-Cloud:

In 2019, it became banal to say we are headed into a multi-cloud world as enterprises started routinely deploying workloads across multiple Infrastructure-as-a-Service

providers.But as applications become even more portable, compute cycles easier to procure in real time, data integration platforms streamline connectivity, and vendors form cross-platform alliances, that multi-cloud trend might start looking more like an omni-cloud one in the near future.

As a general rule, the largest enterprises may soon be customers of all the hyperscalers and some niche providers to boot, allowing them to take advantage of increasingly differentiated services, specific deals and avoid lock-in.The Hearst Corp., which has more than 360 separate businesses, provides a good example of things to come.The New York-based media, information and services company recently engaged its digital transformation across Amazon Web Services (AWS), Microsoft Azure and Google Cloud. That omni-cloud approach gives Hearst developers and divisions the best competitive posture in all their relevant markets.

Kubernetes Breaks And Blurs Cloud Barriers:

Enterprises select the Kubernetes platform best meeting their unique operational needs and capabilities. That could be a prescriptive solution along the Red Hat OpenShift model, an under-the-covers implementation from Pivotal, independent distributions of the likes offered by Docker or Rancher Labs, or native provider services like Google GKE, Microsoft AKS and AWS EKS.

The container orchestrator often then becomes the fabric enabling them to extend applications across disparate cloud infrastructure—delivering on the multi-cloud promise.As such, Kubernetes isn't just bringing a wrecking ball to cloud barriers, but it's also creating a strange market dynamic.

The cloud infrastructure software vendor increasingly being decoupled from the provider that owns the buildings that house the server racks is leading to some offerings that would have been unimaginable a few years back.Consider Google's Anthos service, which can run as easily on Amazon Web Services or Microsoft Azure as it can on Google Cloud Platform. Or the coming VMware Tanzu, that leaps off-premises to span all those hyper-scalers as well.

The multi-cloud world appears to be one where not only customer workloads span clouds, but the cloud providers themselves routinely extend into rival territory.

Kubernetes Companions Create New Silos:

Kubernetes has won the day, but the era of cross-cloud unity the container orchestrator ushers in will be challenged by the ancillary services developing around

it.The Cloud Native Computing Foundation, which keeps tight control over the core Kubernetes project, has been incubating companion technologies to round out the stack. CNCF promotes those projects through its CNCF roadmap, a document detailing a path to comprehensive container adoption based on enterprise use cases.

One technology on that roadmap that's essential for enterprises deploying hybrid applications is the Istio service mesh. Other open source projects are increasingly important components, from Prometheus, to monitor and instrument containerized applications; the ELK Stack for logging and trouble-shooting (which constitutes Elasticsearch, Logstash and Kibana); Harbor, a container registry needed for high availability in production; and Jaeger, emerging as a standard technology for tracing application logs across microservices.

While the cloud giants aren't straying far from upstream Kubernetes functionality, increasingly they look to be promoting competitive technologies to other projects on the CNCF roadmap. For example, AWS is advising customers to use its native CloudWatch service, rather than Prometheus, for monitoring. Google does the same with Stackdriver.

Because balkanization of Kubernetes companion technology can undermine progress toward cross-cloud unity, tensions may flare as more enterprises adopt cloud-native infrastructure.

More Kubernetes Consolidation:

Every cloud infrastructure company—public services provider or on-prem hardware and software vendor—must have a strong Kubernetes proposition to stay competitive in the current market.Again and again, acquisitions have proven an effective means for legacy giants to pour fuel on their nascent Kubernetes divisions, or create new ones from whole cloth.

Most notably in that category, IBM's $34 billion mega-deal for Red Hat, which made a strong Kubernetes play with OpenShift and its earlier acquisition of container pioneer CoreOS.Microsoft expanded its Kubernetes toolkit in 2017 by buying Deis after fully committing to the project. And NetApp jumped into the market with the acquisition of StackPointCloud.

VMware may have gotten the best deal of them all when it bought Heptio, a startup that laid a foundation for development of Tanzu, the platform that will power Dell Technologies cross-cloud vision in the coming years.But there are more container-

focused startups out there than ever before (Docker being one of them), so expect major cloud vendors to aggressively make M&A moves on them in the coming year.

Security Acquisitions:

Enterprises deciding on a cloud provider often want access to a full array of platform-native security tools rather than third-party solutions.Companies that can't develop in-house all the capabilities of a modern security stack have to look to buy them. That's why security acquisitions have been a major theme over the last year.

VMware complemented its organic security development efforts with a deal for Carbon Black, an endpoint protection specialist, and then looked to raise its application security game by buying Intrinsic.Microsoft upped its data security and governance game when it bought Blue Talon earlier this year. Google brought into its fold Chronicle, a sister company in Alphabet (though some published reports suggest the merger is not a hit.)

Broadcom's just completed its deal for Symantec. Last year Cisco bought Duo Security and AT&T bought AlienVault to launch a cyber-security division.

But security is such a complex proposition these days that it seems there are always more gaps to fill. The M&A trend will almost surely continue, if not accelerate, in 2020.There's no shortage of startups that can help cloud providers of all stripes establish a full-court portfolio of native security features.

Private Cloud Repatriation Gets Real:

The inevitable march of workloads into the public cloud is starting to look more like a two-way street.

Containers and other technologies that facilitate application portability are making it easier for repatriation to private infrastructure. Many companies are taking advantage as they become more familiar with the nuanced benefits of different environments.

That doesn't mean the pace of migrations to public cloud will slow, or even decelerate. But traffic will flow more freely in both directions as some customers realize that some workloads actually see cost savings, and performance and security benefits, when running in software-defined data centers.

And private cloud repatriation will further stimulate the red-hot market enabling hybrid cloud environments.

All SaaS Becomes Intelligent SaaS:

Every Software-as-a-Service, IT Ops, analytics and BI product is currently being infused, in various ways and to varying degrees, with machine learning—whether it needs it or not.Beyond the very real advantages artificial intelligence delivers in automation and insight, the term is a huge selling point, one it never hurts to throw into a product pitch.

And from a chatbot to an inference engine to predictive analytics, it's easy for AI to find its way into just about any cloud-software product.Some machine-learning based features are genuinely useful; others just capitalize off the buzz word. But by next year, it will be hard to find a product that's not billed as intelligent.

Ramping SaaS Ops:

As Software-as-a-Service proliferates, more specialized platforms are emerging to manage migrations, operations and spend of those cloud-based apps.To keep up, SaaSOps is increasingly becoming established as a new role for specialized IT experts, often in concert with products from an emerging crop of Cloud Access Security Brokers.

Vendors like BetterCloud, CloudManager and Blissfully are enabling comprehensive management of solution suites like Microsoft Office 365 and Google G Suite, as well as Salesforce and other leading SaaS vendors."The term [SaaSOps] is nebulous now, but can apply to security, license management, spend management, discovery/mitigation of shadow IT, and on-boarding/off-boarding management across platforms and business areas responsible for their own apps," Allen Falcon, CEO of Cumulus Global, explained.

Focus Shifts To App Delivery:

Kubernetes in many ways has brought order to a turbulent cloud infrastructure market. The container orchestrator is now something close to standard tech for deploying infrastructure to support cloud-native workloads.That makes it likely competition will move up the stack to focus on improving application delivery, Dan Kohn, executive director of the Cloud Native Computing Foundation, told CRN.

Major open source projects will emerge to support 'app dev' practices, delivering services facilitating the packaging and deployment of applications.To make good on that promise, CNCF has launched a new Application Delivery Special Interest Group.

HPC Falls For Cloud:

High Performance Computing workloads are typically run sporadically and in batches—a use case where the unique elasticity of the public cloud seems like a major value proposition.

And yet, the HPC market has largely stayed on-premises, despite the obvious benefits of being able to scale up and down quickly with a hyper-scale provider. Only in recent years have HPC users even begun routinely bursting into the cloud to ease bottlenecks in accessing resources.The root of cloud skepticism has a lot to do with the kind of work done on HPC systems—they often aid development of new product designs, proprietary data modeling, and advanced simulations. Scientists, engineers and product developers have been wary to allow those crown jewels off-site.

That dam is breaking. Pretty soon the expensive, multi-core systems needed to do advanced computations will shift to public providers.The impetus to not upgrade capital-intensive data centers, and be able to guarantee resources on-demand, will prove unstoppable, even if HPC costs in the cloud can be tricky to predict.

CHAPTER FIVE

Storage Systems

5.1 Storage models of file systems and data base

Storage models of file systems:

1. **Disk File Systems:**

A Disk File System is probably what most people – at least those who would think about these kinds of things to begin with – have in mind when they think about what a file system does. It manages block device storage, stores “files” in “directories”, maintains a file system hierarchy, controls file metadata, and allows for simple I/O via a few basic system calls. The canonical file system for the Unix family of operating systems is, of course, the Unix File System (UFS), also known as the Berkeley Fast File System (FFS).

2. **Distributed File Systems**

Unlike a disk file system, which provides access to the data it holds only to the processes running in the operating system the disk is attached to, a distributed file system may allow different client systems to access a centralized storage resource simultaneously, thus forming the basis for any NAS solutions.

3. "Other" File Systems

In the Unix world, there exists an old mantra: "Everything is a file." That is, the simple API defining file I/O has proven so useful that a number of non-file "things" have come to implement this interface as well: not only can regular files be accessed using the open(2), read(2), write(2) and close(2) system calls17, but so can network sockets and other interprocess communication endpoints, character and block devices, so-called pseudo-devices (such as /dev/null), and various other "special files".

Database storage models:

1. Hierarchical model

In a hierarchical model, data is organized into a tree-like structure, implying a single parent for each record. A sort field keeps sibling records in a particular order. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System.

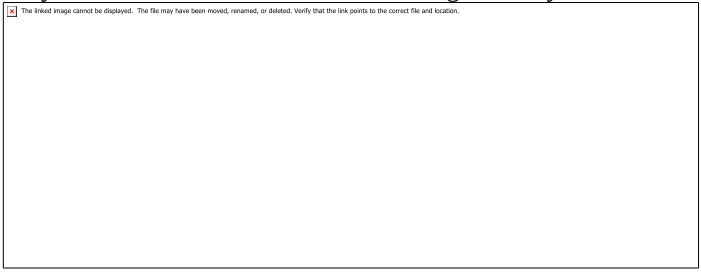


Figure 5.1 Hierarchial Model

2.Network Model

The network model expands upon the hierarchical structure, allowing many-to-many relationships in a tree-like structure that allows multiple parents. It was most

popular before being replaced by the relational model, and is defined by the CODASYL specification.

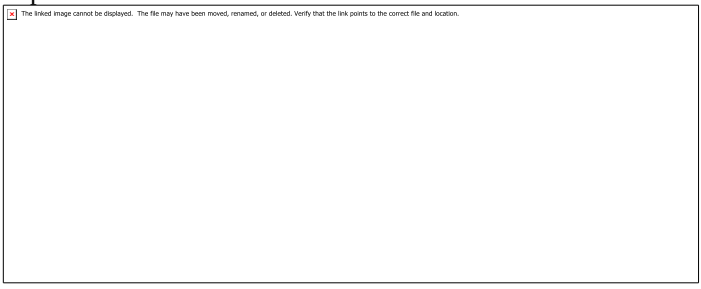


Figure 5.2 Network Model

1. **Inverted file model**

In an *inverted file* or *inverted index*, the contents of the data are used as keys in a lookup table, and the values in the table are pointers to the location of each instance of a given content item. This is also the logical structure of contemporary database indexes, which might only use the contents from a particular columns in the lookup table. The *inverted file data model* can put indexes in a second set of files next to existing flat database files, in order to efficiently directly access needed records in these files.

2. **Relational model**

The relational model was introduced by E.F. Codd in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory, and systems implementing it have been used by mainframe, midrange and microcomputer systems.

5.2 Amazon S3

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers.Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable,

reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

1. Working with Amazon S3 Buckets

Amazon S3 is cloud storage for the Internet. To upload your data (photos, videos, documents etc.), you first create a bucket in one of the AWS Regions. You can then upload any number of objects to the bucket.

In terms of implementation, buckets and objects are resources, and Amazon S3 provides APIs for you to manage them. For example, you can create a bucket and upload objects using the Amazon S3 API. You can also use the Amazon S3 console to perform these operations. The console internally uses the Amazon S3 APIs to send requests to Amazon S3.

Amazon S3 bucket names are globally unique, regardless of the AWS Region in which you create the bucket. You specify the name at the time you create the bucket.

Amazon S3 creates buckets in a region you specify. You can choose any AWS Region that is geographically close to you to optimize latency, minimize costs, or address regulatory requirements. For example, if you reside in Europe, you might find it advantageous to create buckets in the EU (Ireland) or EU (Frankfurt) regions

Figure 5.3 Amazon S3

2. **Working with Amazon S3 Objects**

Amazon S3 is a simple key, value store designed to store as many objects as you want. You store these objects in one or more buckets. An object consists of the following:

-
- Key – The name that you assign to an object. You use the object key to retrieve the object.
- Version ID – Within a bucket, a key and version ID uniquely identify an object.

The version ID is a string that Amazon S3 generates when you add an object to a bucket. Value – The content that you are storing.

An object value can be any sequence of bytes. Objects can range in size from zero to 5 TB. Metadata – A set of name-value pairs with which you can store information regarding the object. You can assign metadata, referred to as user-defined metadata, to your objects in Amazon S3. Amazon S3 also assigns system-metadata to these objects, which it uses for managing objects Subresources – Amazon S3 uses the subresource mechanism to store object-specific additional information.

Because subresources are subordinates to objects, they are always associated with some other entity such as an object or a bucket.

Access Control Information – You can control access to the objects you store in Amazon S3. Amazon S3 supports both the resource-based access control, such as an Access Control List (ACL) and bucket policies, and user-based access control.

3. **Managing Access Permissions to Your Amazon S3 Resources**

By default, all Amazon S3 resources—buckets, objects, and related subresources (for example,lifecycle configuration and website configuration)—are private: only the

resource owner, an AWS account that created it, can access the resource. The resource owner can optionally grant access permissions to others by writing an access policy.

Amazon S3 offers access policy options broadly categorized as resource-based policies and user policies. Access policies you attach to your resources (buckets and objects) are referred to as resource-based policies. For example, bucket policies and access control lists (ACLs) are resource-based policies. You can also attach access policies to users in your account. These are called user policies. You may choose to use resource-based policies, user policies, or some combination of these to manage permissions to your Amazon S3 resources. The introductory topics provide general guidelines for managing permissions.

5.3 Mega Store Architecture

Megastore handles over 3 billion writes and 20 billion reads daily on almost 8 PB of primary data across many global data centers.

The mission Support Internet apps such as Google's AppEngine.

- Scale to millions of users
- Responsive despite Internet latencies to impatient users
- Easy for developers
- Fault resilience from drive failures to data center loss and everything in between
- Low-latency synchronous replication to distant sites Availability and scale

To achieve availability and global scale the designers implemented two key architectural features:

- For availability, an asynchronous log replicator optimized for long-distance
- For scale, data partitioned into small databases each with its own replicated log
- Entities

An e-mail account is a natural entity. But defining other entities is more complex.

- Geographic data lacks natural granularity. For example, the globe is divided into non-overlapping entities. Changes across these geographic entities use (expensive) two-phase commits.

Replication:

Megastore uses Paxos to manage synchronous replication. But in order to make Paxos practical despite high latencies the team developed some optimizations:

- Fast reads. Current reads are usually from local replicas since most writes succeed on all replicas.
- Fast writes. Since most apps repeatedly write from the same region, the initial writer is granted priority for further replica writes. Using local replicas and reducing write contention for distant replicas minimizes latency.
- Replica types. In addition to full replicas Megastore has 2 other replica types:

witness replicas. Witnesses vote in Paxos rounds and store the write-ahead log but do not store entity data or indexes to keep storage costs low. They are also tiebreakers when isn't a quorum.

Read-only replicas are the inverse: nonvoting replicas that contain full snapshots of the data. Their data may be slightly stale but they help disseminate the data over a wide area without slowing writes.

API

The insight driving the API is that the big win is scalable performance rather than a rich query language. Thus a focus on controlling physical locality and hierarchical layouts.

Architecture:

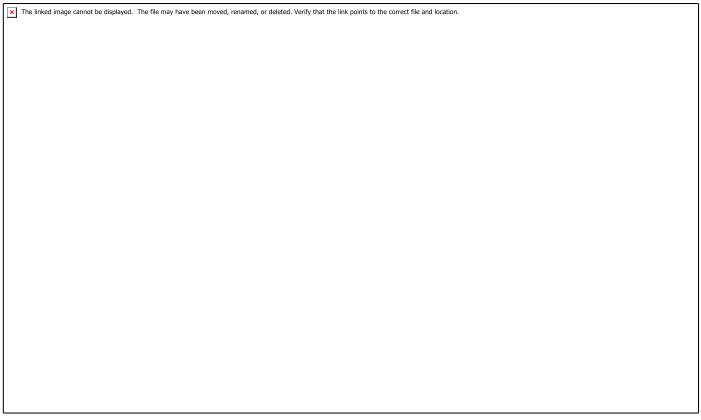


Figure 5.4 Mega Store Architecture

Availability:

As coordinator servers do most local reads their availability is critical to maintaining Megastore's performance. The coordinators use an out-of-band protocol to track other coordinators and use Google's Chubby distributed lock service to obtain remote locks. If the coordinator loses a majority of its locks it will consider all entities in its purview to be out of date until the locks are regained and the coordinator is current.

Performance:

Because Megastore is geographically distributed, application servers in different locations may initiate writes to the same end entity group simultaneously. Only one of them will succeed and the other writers will have to retry.

TheStorageMojo:

As Brewer's CAP theorem showed, a distributed system can't provide consistency, availability and partition tolerance to all nodes at the same time. But this paper shows that by making smart choices we can get darn close as far as human users are concerned.

5.4 Google's Big Table

Bigtable is a compressed, high performance, and proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies. On May 6, 2015, a public version of Bigtable was made available as a service. Bigtable also underlies Google Cloud Datastore, which is available as a part of the Google Cloud Platform.

Bigtable development began in 2004 and is now used by a number of Google applications, such as web indexing,MapReduce, which is often used for generating and modifying data stored in Bigtable,Google MapsGoogle Book Search, "My Search History", Google Earth, Blogger.com.

Code hosting, YouTube,and Gmail.Google's reasons for developing its own database include scalability and better control of performance characteristics

Google's Spanner RDBMS is layered on an implementation of Bigtable with a Paxos group for two-phase commits to each table. Google F1 was built using Spanner to replace an implementation based on MySQL.

Bigtable maps two arbitrary string values (row key and column key) and timestamp (hence three- dimensional mapping) into an associated arbitrary byte array. It is not a relational database and can be better defined as a sparse, distributed multi-dimensional sorted map Bigtable is designed to scale into the petabyte range across "hundreds or thousands of machines, and to make it easy to add more machines the system and automatically start taking advantage of those resources without any reconfiguration".

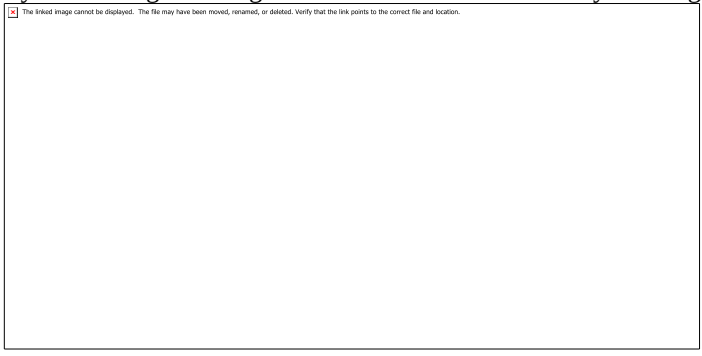


Figure 5.5 Big Table Architecture

Google's thorough description of Bigtable's inner workings has allowed other organizations and open source development teams to create Bigtable derivatives, including the Apache HBase database, which is built to run on top of the Hadoop Distributed File System (HDFS). Other examples include Cassandra, which originated at Facebook Inc., and Hypertable, an open source technology that is marketed in a commercial version as an alternative to HBase.

5.5 General Parallel File System

The General Parallel File System (GPFS) is a high-performance clustered file system developed by IBM. It can be deployed in shared-disk or shared-nothing distributed parallel modes. It is used by many of the world's largest commercial companies, as well as some of the supercomputers on the Top 500 List.For example, GPFS was the filesystem of the ASC Purple Supercomputer which was composed of more than 12,000 processors and has 2 petabytes of total disk storage spanning more than 11,000 disks.GPFS began as the Tiger Shark file system, a research project at IBM's Almaden Research Center as early as 1993. Shark was initially designed to support high throughput multimedia applications. This design turned out to be well suited to scientific computing.

GPFS provides high performance by allowing data to be accessed over multiple computers at once. Most existing file systems are designed for a single server environment, and adding more file servers does not improve performance. GPFS provides higher input/output performance by striping blocks of data from individual files over multiple disks, and reading and writing these blocks in parallel. Other features provided by GPFS include high availability, support for heterogeneous clusters, disaster recovery, security, DMAPI, HSMand ILM.

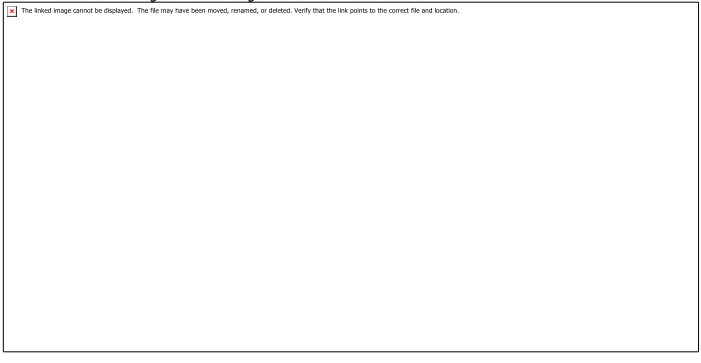


Figure 5.6 General Parallel File System

Other features of the filesystem include

- Distributed metadata, including the directory tree. There is no single "directory controller" or "index server" in charge of the filesystem.

- Efficient indexing of directory entries for very large directories. Many filesystems are limited to a small number of files in a single directory (often, 65536 or a similar small binary number). GPFS does not have such limits.
- Distributed locking. This allows for full Posix filesystem semantics, including locking for exclusive file access.
- Partition Aware. A failure of the network may partition the filesystem into two or more groups of nodes that can only see the nodes in their group. This can be detected through a heartbeat protocol, and when a partition occurs, the filesystem remains live for the largest partition formed. This offers a graceful degradation of the filesystem — some machines will remain working.
- Filesystem maintenance can be performed online. Most of the filesystem maintenance chores (adding new disks, rebalancing data across disks) can be performed while the filesystem is live. This ensures the filesystem is available more often, so keeps the supercomputer cluster itself available for longer.

5.5 Architecture of GFS clustering

GFS is enhanced for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that must be retained.

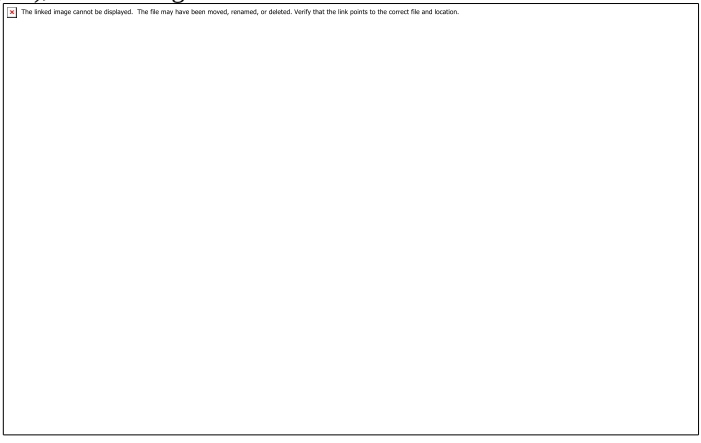


Figure 5.7 Google File System

Google effort, "BigFiles", developed by Larry Page and Sergey Brin in the early days

of Google, while it was still located in Stanford. Files are divided into fixed-size *chunks*of 64 megabytes, similar to clusters or sectors in regular file systems, which are only extremely rarely overwritten, or shrunk; files are usually appended to or read. It is also designed and optimized to run on Google's computing clusters, dense nodes which consist of cheap "commodity" computers, which means precautions must be taken against the high failure rate of individual nodes and the subsequent data loss. Other design decisions select for high data throughputs, even when it comes at the cost of latency.

A GFS cluster consists of multiple nodes. These nodes are divided into two types: one *Master* node and a large number of *Chunkservers*. Each file is divided into fixed-size chunks. Chunk servers store these chunks. Each chunk is assigned a unique 64-bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network, with the minimum being three, but even more for files that have high end-in demand or need more redundancy.

The Master server does not usually store the actual chunks, but rather all the metadata associated with the chunks, such as the tables mapping the 64-bit labels to chunk locations and the files they make up, the locations of the copies of the chunks, what processes are reading or writing to a particular chunk, or taking a "snapshot" of the chunk pursuant to replicate it (usually at the instigation of the Master server, when, due to node failures, the number of copies of a chunk has fallen beneath the set number). All this metadata is kept current by the Master server periodically receiving updates from each chunk server ("Heart-beat messages").

Permissions for modifications are handled by a system of time-limited, expiring "leases", where the Master server grants permission to a process for a finite period of time during which no other process will be granted permission by the Master server to modify the chunk. The modifying chunkserver, which is always the primary chunk holder, then propagates the changes to the chunkservers with the backup copies. The changes are not saved until all chunkservers acknowledge, thus guaranteeing the completion and atomicity of the operation.

Programs access the chunks by first querying the Master server for the locations of the desired chunks; if the chunks are not being operated on (i.e. no outstanding leases

exist), the Master replies with the locations, and the program then contacts and receives the data from the chunkserver directly (similar to Kazaa and its supernodes).

Unlike most other file systems, GFS is not implemented in the kernel of an operating system, but is instead provided as a userspace library.