Sixth Semester B.E. Degree Examination, July/August 2022
Cloud Computing and Its Applications

Time: 3 hrs. Max. Marks: 100

Note: *Answer any FIVE full questions, choosing ONE full question from each module.*

ANSWER SCHEME

Module 1

1.

a. With neat diagram, explain cloud computing reference model. **(08 Marks)**

The Cloud Reference Model

Cloud computing supports any IT service that can be consumed as a utility and delivered through a network, most likely the Internet. Such characterization includes quite different aspects: infrastructure, development platforms, application and services.

Architecture

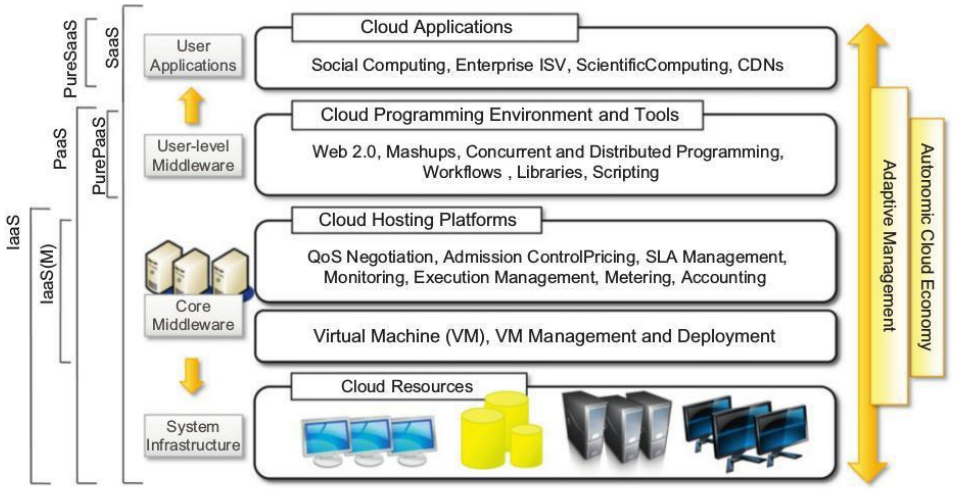


FIGURE 4.1
The cloud computing architecture.

It is possible to organize all the concrete realizations of cloud computing into a layered view covering the entire stack (see Figure 4.1), from hardware appliances to software systems. Cloud resources are harnessed to offer "computing horsepower" required for providing services. Cloud infrastructure can be heterogeneous in nature because a variety of resources, such as clusters and even networked PCs, can be used to build it.

The physical infrastructure is managed by the core middleware, the objectives of which are to provide an appropriate runtime environment for applications and to best utilize resources. At the bottom of the stack, virtualization technologies are used to guarantee runtime environment customization, application isolation, sandboxing, and quality of service. Hardware virtualization is most commonly used at this level. Hypervisors manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines. By using virtual machine technology it is possible to finely partition the hardware resources such as CPU and memory and to virtualize specific devices, thus meeting the requirements of users and applications. This solution is generally paired with storage and network virtualization strategies, which allow the infrastructure to be completely virtualized and controlled.

Infrastructure management is the key function of core middleware, which supports capabilities such as negotiation of the quality of service, admission control, execution management and monitoring, accounting, and billing.

The combination of cloud hosting platforms and resources is generally classified as an Infrastructure-as-a-Service (IaaS) solution. We can organize the different examples of IaaS into two categories: Some of them provide both the management layer and the physical infrastructure; others provide only the

management layer (IaaS (M)).

In this second case, the management layer is often integrated with other IaaS solutions that provide physical infrastructure and adds value to them.

IaaS solutions are suitable for designing the system infrastructure but provide limited services to build applications. Such service is provided by cloud programming environments and tools, which form a new layer for offering users a development platform for applications.

The range of tools include Web-based interfaces, command-line tools, and frameworks for concurrent and distributed programming. In this scenario, users develop their applications specifically for the cloud by using the API exposed at the user-level middleware. For this reason, this approach is also known as Platform-as-a-Service (PaaS) because the service offered to the user is a development platform rather than an infrastructure.

The top layer of the reference model depicted in Figure 4.1 contains services delivered at the application level. These are mostly referred to as Software-as -a-Service (SaaS). In most cases these are Web-based applications that rely on the cloud to provide service to end users. The horsepower of the cloud provided by IaaS and PaaS solutions allows independent software vendors to deliver their application services over the Internet.

Table 4.1 summarizes the characteristics of the three major categories used to classify cloud computing solutions. In the following section, we briefly discuss these characteristics along with some references to practical implementations.

b. List the characteristics and benefits of cloud computing. **(06 Marks)**

Characteristics and Benefits of Cloud Computing

1. On-demand self-service

AWS, Microsoft Azure, Google Cloud and other public cloud platforms make resources available to users at the click of a button or API call. With data centers all over the world, these vendors have vast amounts of compute and storage assets at the ready. This represents a radical departure for IT teams accustomed to an on-premises procurement process that can take months to complete.

2. Resource Pooling

Public cloud providers rely on multi-tenant architectures to accommodate more users at the same time. Customers' workloads are abstracted from the hardware and underlying software, which serve multiple customers on the same host. Cloud providers increasingly rely on custom hardware and abstraction layers to improve security and speed users' access to resources.

3. Scalability and Rapid Elasticity

Resource pooling enables scalability for cloud providers and users, letting them add or remove compute, storage, networking and other assets as needed. This helps enterprise IT teams optimize their cloud-hosted workloads and avoid end-user bottlenecks. Clouds can scale vertically or horizontally, and service providers offer automation software to handle dynamic scaling for users.

4. Pay-per-use pricing

This cloud computing characteristic shifts IT spending from Capex to Opex as providers offer per-second billing. This model achieves economies of scale through reducing costs on a large scale and seeing an increase in efficiency. Though this can generally be seen as a positive, IT teams must be careful since their resource needs likely aren't static. VMs should be right-sized, turned off while not in use, or scaled down as conditions dictate. Otherwise, organizations waste money and can end up with sticker shock when the monthly bill arrives.

5. Measured service

Measuring cloud service usage is useful for both a cloud provider and its customers. The provider and the customer monitor and report on the use of resources and services, such as VMs, storage, processing and bandwidth. That data is used to calculate the customer's consumption of cloud resources and feeds into the pay-per-use model. The cloud provider, meanwhile, can better understand how customers utilize its resources and potentially improve the infrastructure and cloud computing services offered.

6. Resiliency and availability
Cloud providers use several techniques to guard against downtime, such as minimizing regional dependencies to avoid single points of failure. Users can also extend their workloads across availability zones, which have redundant networks connecting multiple data centers in relatively close proximity. Some higher-level services automatically distribute workloads across availability zones.

7. Security
While many enterprises balked at migrating workloads because of security fears, those concerns have largely subsided, partly due to the benefits of the above characteristics of cloud computing. Cloud vendors employ some of the best security experts in the world and are generally better equipped to handle threats than most in-house IT teams. In fact, some of the biggest financial firms in the world say the cloud is a security asset.

8. Broad network access
A big part of the cloud's utility is its ubiquity. Data can be uploaded and accessed from anywhere with an internet connection. Users can work from any location. The cloud is an attractive option for most enterprises that have a mix of operating systems, platforms and devices.

c. Write a note on challenges in cloud computing. **(06 Marks)**

Challenges in Cloud Computing

Availability of service; what happens when the service provider cannot deliver? Can a large company such as GM move its IT activities to the cloud and have assurances that its activity will not be negatively affected by cloud overload? A partial answer to this question is provided by Service Level Agreements (SLA)s. A temporary fix but with negative economical implications is overprovisioning, i.e., having enough resources to satisfy the largest projected demand.
Vendor lock-in; once a customer is hooked to one cloud service provider it is hard to move to another. The standardization efforts at NIST attempt to address this problem.
Data confidentiality and auditability; this is indeed a serious problem to be analyzed.
Data transfer bottlenecks critical for data-intensive applications. Transferring 1 TB of data on a 1 Mbps network takes 8 000 000 seconds or about 10 days; it is faster and cheaper to use courier service and send data recoded on some media than to send it over the network. Very high speed networks will alleviate this problem in the future, e.g., a 1 Gbps network would reduce this time to 8000 seconds, or slightly more than 2 hours.
Performance unpredictability; this is one of the consequences of resource sharing. Strategies for performance isolation need to studied in detail.
Elasticity, the ability to scale up and down quickly. New algorithms for controlling resource allocation and workload placement are necessary. Autonomic computing based on self-organization and self-management seems to be a promising avenue.

OR

2.

a. Explain in brief the services provided by following cloud
i) Amazon Web Service ii) Microsoft Azure iii) Hadoop. **(10 Marks)**

i) Amazon Web Service
a. AWS computing, storage, and communication services.
Amazon was the first provider of cloud computing; it announced a limited public beta release of its Elastic Computing platform called EC2 in August 2006. Elastic Compute Cloud (EC2) is a web service with a simple interface for launching instances of an application under several operating systems.
b. Elastic Block Store.
EBS provides persistent block level storage volumes for use with EC2 instances. A volume appears to an application as a raw, unformatted and reliable physical disk; the size of the storage volumes ranges from one gigabyte to one terabyte.

c. Simple DB.

Simple DB is a non-relational data store that allows developers to store and query data items via web services requests; it supports store and query functions traditionally provided only by relational databases. Simple DB creates multiple geographically distributed copies of each data item and supports high performance web applications; at the same time, it manages automatically the infrastructure provisioning, hardware and software maintenance, replication and indexing of data items, and performance tuning.

d. Simple Queue Service.

SQS is a hosted message queue. SQS is a system for supporting automated workflows; it allows multiple EC2 instances to coordinate their activities by sending and receiving SQS messages. Any computer connected to the Internet can add or read messages without any installed software or special firewall configurations.

ii) Microsoft Azure

Azure and Online Services are PaaS and, respectively, SaaS cloud platforms provided by Microsoft. Azure is an operating system, SQL Azure is a cloud-based version of the SQL Server, and Azure AppFabric (formerly .NET Services) is a collection of services for cloud applications.

Windows Azure has three core components: Compute which provides a computation environment, Storage for scalable storage, and Fabric Controller which deploys, manages, and monitors applications; it interconnects nodes consisting of servers, high-speed connections, and switches. The Content Delivery Network (CDN) maintains cache copies of data to speedup computations. The Connect subsystem supports IP connections between the users and their applications running on Windows Azure.

The API interface to Windows Azure is built on REST, HTTP and XML. The platform includes five services: Live Services, SQL Azure, AppFabric, SharePoint, and Dynamics CR. A client library and tools are also provided for developing cloud applications in Visual Studio.

The computations carried out by an application are implemented as one or more roles; an application typically runs multiple instances of a role. One distinguishes: (i) Web role instances used to create web applications; (ii) Worker role instances used to run Window-based code; and (iii) VM role instances running user-provided Windows Server 2008 R2 images.

iii) Hadoop

The solutions just listed are explicitly based on Hadoop. Cloud providers also offer other services, based on different technologies, for managing and analyzing large amounts of data. Some offer SQL-like query capabilities similar to Hive or Apache Impala, and others offer processing pipelines like Apache Oozie. It may be possible to use those services to augment Hadoop clusters, managed either directly or through the cloud provider's own prepackaged solution, depending on where and how data is stored.

Of course, these tools share the same disadvantages as the Hadoop-based solutions in terms of moving further away from the open source world and its interoperability benefits. Since they are not based on Hadoop, there is a separate learning curve for them, and the effort could be wasted if they are ever discarded in favor of something that works on Hadoop, or on a different cloud provider, or even on-prem. Their ready availability and ease of use, however, can be attractive.

b. Define virtualization and explain hardware level Virtualization with its pro's and con's of virtualization. **(10 Marks)**

Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure-based services. It encompasses a collection of solutions allowing the abstraction of some of the fundamental elements for computing, such as hardware, runtime environments, storage, and networking. Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are untrusted, without affecting other users' applications. The basis of this technology is the ability of a computer pro-gram—or a combination of software and hardware—to emulate an executing environment separate from the one that hosts such programs.

Hardware level virtualization

Hardware-assisted virtualization. This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation. This technique was originally introduced in the IBM System/370. At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel VT (formerly known as Vanderpool) and AMD V (formerly known as Pacifica). Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

Full virtualization. Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware. To make this possible, virtual machine managers are required to provide a complete emulation of the entire underlying hardware. The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform.

Paravirtualization. This is a not-transparent virtualization solution that allows implementing thin virtual machine managers. Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified. The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution.

Partial virtualization. Partial virtualization provides a partial emulation of the underlying hard-ware, thus not allowing the complete execution of the guest operating system in complete isolation. Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported.

Advantages of Virtualization

• Managed execution and isolation are perhaps the most important advantages of virtualization. In the case of techniques supporting the creation of virtualized execution environments, these two characteristics allow building secure and controllable computing environments.

• Portability is another advantage of virtualization, especially for execution virtualization techni-ques. Virtual machine instances are normally represented by one or more files that can be easily transported with respect to physical systems.

• Portability and self-containment also contribute to reducing the costs of maintenance, since the number of hosts is expected to be lower than the number of virtual machine instances. Since the guest program is executed in a virtual environment, there is very limited opportunity for the guest program to damage the underlying hardware.

• Finally, by means of virtualization it is possible to achieve a more efficient use of resources. Multiple systems can securely coexist and share the resources of the underlying host, without interfering with each other.

Disadvantages of Virtualization

Degradation

Performance is definitely one of the major concerns in using virtualization technology. Since virtualization interposes an abstraction layer between the guest and the host, the guest can experience increased latencies (delays).

For instance, in the case of hardware virtualization, where the intermediate emulates a bare machine on top of which an entire system can be installed, the causes of performance degradation can be traced back to the overhead introduced by the following activities:

• Maintaining the status of virtual processors
• Support of privileged instructions (trap and simulate privileged instructions)
• Support of paging within VM
• Console functions

Inefficiency and Degraded User Experience

Virtualization can sometime lead to an inefficient use of the host. In particular, some of the specific features of the host cannot be exposed by the abstraction layer and then become inaccessible. In the case of hardware virtualization, this could happen for device drivers: The virtual machine can sometime simply provide a default graphic card that maps only a subset

of the features available in the host. In the case of programming-level virtual machines, some of the features of the underlying operating systems may become inaccessible unless specific libraries are used.

Security Holes and New Threats

Virtualization opens the door to a new and unexpected form of phishing. The capability of emulating a host in a completely transparent manner led the way to malicious programs that are designed to extract sensitive information from the guest.

The same considerations can be made for programming-level virtual machines: Modified versions of the runtime environment can access sensitive information or monitor the memory locations utilized by guest applications while these are executed.

Module 2

3.

a. Explain the following in detail
i) Hardware as service ii) Platform as service iii) Software as service. **(10 Marks)**

i) Hardware As A Service

Infrastructure- and Hardware-as-a-Service (IaaS/HaaS) solutions are the most popular and developed market segment of cloud computing. They deliver customizable infrastructure on demand. The available options within the IaaS offering umbrella range from single servers to entire infrastructures, including network devices, load balancers, and database and Web servers. The main technology used to deliver and implement these solutions is hardware virtualization: one or more virtual machines opportunely configured and interconnected define the distributed system on top of which applications are installed and deployed. Virtual machines also constitute the atomic components that are deployed and priced according to the specific features of the virtual hardware: memory, number of processors, and disk storage. From the perspective of the customer it reduces the administration and maintenance cost as well as the capital costs allocated to purchase hardware.

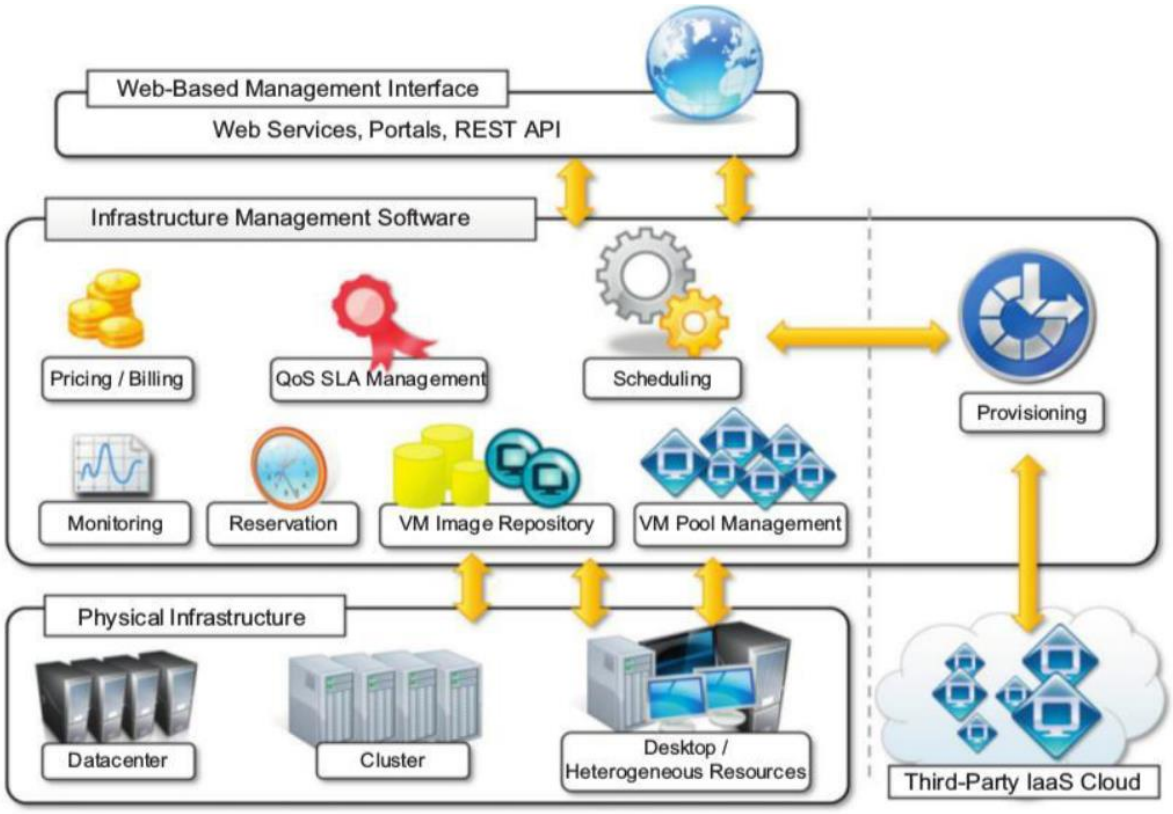


FIGURE 4.2

Infrastructure-as-a-Service reference implementation.

ii) Platform As A Service

Platform-as-a-Service (PaaS) solutions provide a development and deployment platform for running applications in the cloud. They constitute the middleware on top of which applications are built. A general overview of the features characterizing the PaaS approach is given in Figure 4.3. Application management is the core functionality of the middleware. PaaS implementations provide applications with a runtime environment and do not expose any service for managing the underlying infrastructure. They automate the process of deploying applications to the infrastructure, configuring application components, provisioning and configuring supporting technologies such as load balancers and databases, and managing system change based on policies set by the user. The core middleware is in charge of managing the resources and scaling applications on demand or automatically, according to the commitments made with users. From a user point of view, the core middleware exposes interfaces that allow programming and deploying applications on the cloud. These can be in the form of a Web-based interface or in the form of programming APIs and libraries.

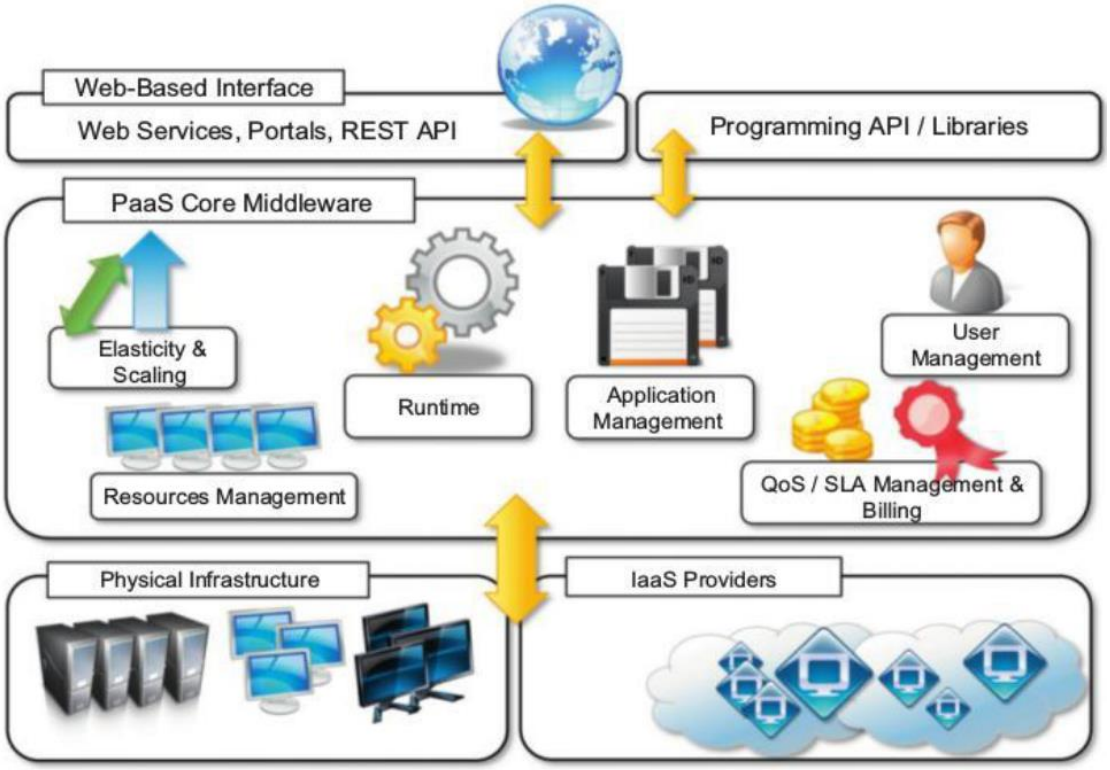


FIGURE 4.3

The Platform-as-a-Service reference model.

iii) Software As A Service

Software-as -a-Service (SaaS) is a software delivery model that provides access to applications through the Internet as a Web-based service. It provides a means to free users from complex hardware and software management by offloading such tasks to third parties, which build applications accessible to multiple users through a Web browser. In this scenario, customers neither need install anything on their premises nor have to pay considerable up-front costs to purchase the software and the required licenses. The SaaS model is appealing for applications serving a wide range of users and that can be adapted to specific needs with little further customization. This requirement characterizes SaaS as a "one-to-many" software delivery model, whereby an application is shared across multiple users. This is the case of CRM 3 and ERP 4 applications that constitute common needs for almost all enterprises, from small to medium-sized and large business. Every enterprise will have the same requirements for the basic features concerning CRM and ERP; different needs can be satisfied with further customization.

ASPs (application service providers) has some of the core characteristics of SaaS:

- The product sold to customer is application access.
- The application is centrally managed.
- The service delivered is one-to-many.

• The service delivered is an integrated solution delivered on the contract, which means provided as promised.

ASPs provide access to packaged software solutions that addressed the needs of a variety of customers. The SaaS approach introduces a more flexible way of delivering application services that are fully customizable by the user by integrating new services, injecting their own components, and designing the application and information workflows. The benefits delivered are the following:

1. Software cost reduction and total cost of ownership (TCO) were paramount
2. Service-level improvements
3. Rapid implementation
4. Standalone and configurable applications
5. Rudimentary application and data integration
6. Subscription and pay-as-you-go (PAYG) pricing

b. Explain the open challenges faced in cloud computing in detail. **(10 Marks)**

Cloud computing presents many challenges for industry and academia. There is a significant amount of work in academia focused on defining the challenges brought by this phenomenon.

In this section, we highlight the most important ones.

- Cloud definition
- Cloud interoperability and standards
- Scalability and fault tolerance
- Security, trust, and privacy
- Organizational aspects

4.4.1 Cloud definition

There have been several attempts made to define cloud computing and to provide a classification of all the services and technologies identified as such. NSIT characterizes cloud computing as on-demand self-service, broad network access, resource-pooling, rapid elasticity, and measured service; classifies services as SaaS, PaaS, and IaaS; and categorizes deployment models as public, private, community, and hybrid clouds. Alternative taxonomies for cloud services. David Linthicum, founder of Blue Mountains Labs, provides a more detailed classification, which comprehends 10 different classes and better suits the vision of cloud computing within the enterprise. These characterizations and taxonomies reflect what is meant by cloud computing at the present time, but being in its infancy the phenomenon is constantly evolving, and the same will happen to the attempts to capture the real nature of cloud computing.

4.4.2 Cloud interoperability and standards

To fully realize this goal, introducing standards and allowing interoperability between solutions offered by different vendors are objectives of fundamental importance. Vendor lock-in constitutes one of the major strategic barriers against the seamless adoption of cloud computing at all stages. The presence of standards that are actually implemented and adopted in the cloud computing community could give room for interoperability and then lessen the risks resulting from vendor lock-in. The first steps toward a standardization process have been made, and a few organizations, such as the Cloud Computing Interoperability Forum (CCIF), the Open Cloud Consortium, and the DMTF Cloud Standards Incubator, are leading the path.

Another interesting initiative is the Open Cloud Manifesto, which embodies the point of view of various stakeholders on the benefits of open standards in the field. The Open Virtualization Format (OVF) is an attempt to provide a common format for storing the information and metadata describing a virtual machine image. Even though the OVF provides a full specification for packaging and distributing virtual machine images in completely platform-independent fashion, it is supported by few vendors that use it to import static virtual machine images.

4.4.3 Scalability and fault tolerance

The ability to scale on demand constitutes one of the most attractive features of cloud computing. Clouds allow scaling beyond the limits of the existing in-house IT resources,

whether they are infrastructure (compute and storage) or applications services. To implement such a capability, the cloud middleware has to be designed with the principle of scalability along different dimensions in mind— for example, performance, size, and load.

The cloud middleware manages a huge number of resource and users, which rely on the cloud to obtain the horsepower. In this scenario, the ability to tolerate failure becomes fundamental, sometimes even more important than providing an extremely efficient and optimized system. Hence, the challenge in this case is designing highly scalable and fault-tolerant systems that are easy to manage and at the same time provide competitive performance.

4.4.4 Security, trust, and privacy

Security, trust, and privacy issues are major obstacles for massive adoption of cloud computing. The traditional cryptographic technologies are used to prevent data tampering and access to sensitive information. The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable. Information can be stored within a cloud storage facility using the most advanced technology in cryptography to protect data and then be considered safe from any attempt to access it without the required permissions. The lack of control over data and processes also poses severe problems for the trust we give to the cloud service provider and the level of privacy we want to have for our data.

4.4.5 Organizational aspects

More precisely, storage, compute power, network infrastructure, and applications are delivered as metered services over the Internet. This introduces a billing model that is new within typical enterprise IT departments, which requires a certain level of cultural and organizational process maturity.

In particular, the following questions have to be considered:

- What is the new role of the IT department in an enterprise that completely or significantly relies on the cloud?
- How will the compliance department perform its activity when there is a considerable lack of control over application workflows?
- What are the implications (political, legal, etc.) for organizations that lose control over some aspects of their services?
- What will be the perception of the end users of such services?

From an organizational point of view, the lack of control over the management of data and processes poses not only security threats but also new problems that previously did not exist.

OR

4.

a. Exp1ain the following deployment mode platform for building Aneka cloud
i) Private cloud, ii) Public cloud, and iii) Hybrid cloud. **(10 Marks)**

i) Private cloud

A private deployment mode is mostly constituted by local physical resources and infrastructure management software providing access to a local pool of nodes, which might be virtualized.

Figure 5.5 shows a common deployment for a private Aneka Cloud. This deployment is acceptable for a scenario in which the workload of the system is predictable and a local virtual machine manager can easily address excess capacity demand. Most of the Aneka nodes are constituted of physical nodes with a long lifetime and a static configuration and generally do not need to be reconfigured often. The different nature of the machines harnessed in a private environment allows for specific policies on resource management and usage that can be accomplished by means of the Reservation Service. For example, desktop machines that are used during the day for office automation can be exploited outside the standard working hours to execute distributed applications. Workstation clusters might have some specific legacy software that is required for supporting the execution of applications and should be executed with special requirements.

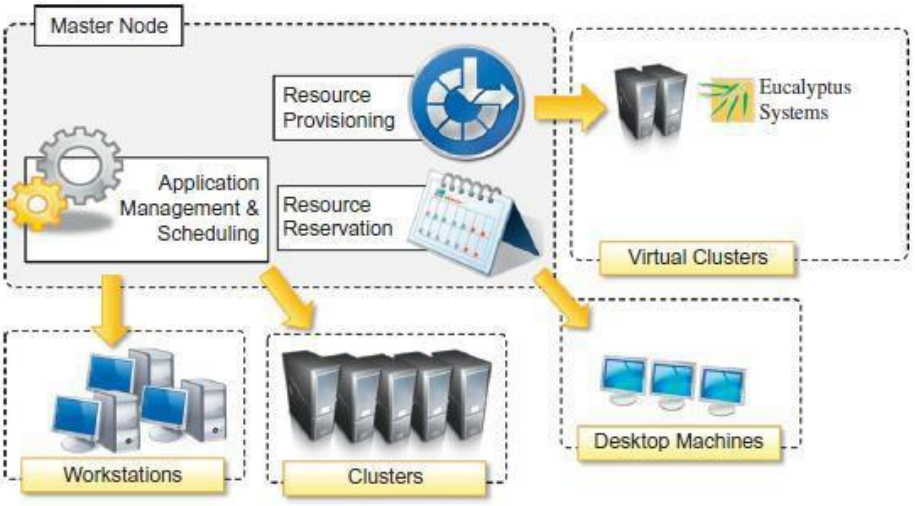


FIGURE 5.5

Private cloud deployment.

ii) Public cloud

Public Cloud deployment mode features the installation of Aneka master and worker nodes over a completely virtualized infrastructure that is hosted on the infrastructure of one or more resource providers such as Amazon EC2 or GoGrid.

Figure 5.6 provides an overview of this scenario. The deployment is generally contained within the infrastructure boundaries of a single IaaS pro- vider. The reasons for this are to minimize the data transfer between different providers, which is generally priced at a higher cost, and to have better network performance. In this scenario it is pos- sible to deploy an Aneka Cloud composed of only one node and to completely leverage dynamic provisioning to elastically scale the infrastructure on demand. A fundamental role is played by the Resource Provisioning Service, which can be configured with different images and templates to instantiate. Other important services that have to be included in the master node are the Accounting and Reporting Services. These provide details about resource utilization by users and applications and are fundamental in a multitenant Cloud where users are billed according to their consumption of Cloud capabilities.

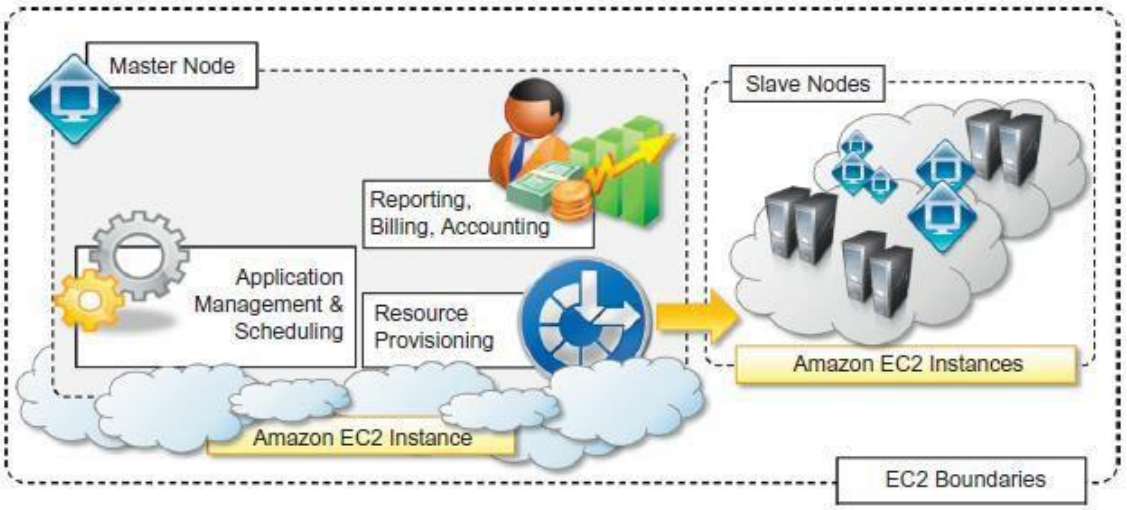


FIGURE 5.6

Public Aneka cloud deployment.

iii) Hybrid cloud

The hybrid deployment model constitutes the most common deployment of Aneka. In many cases, there is an existing computing infrastructure that can be leveraged to address the computing needs of applications. This infrastructure will constitute the static deployment of Aneka that can be elastically scaled on demand when additional resources are required.

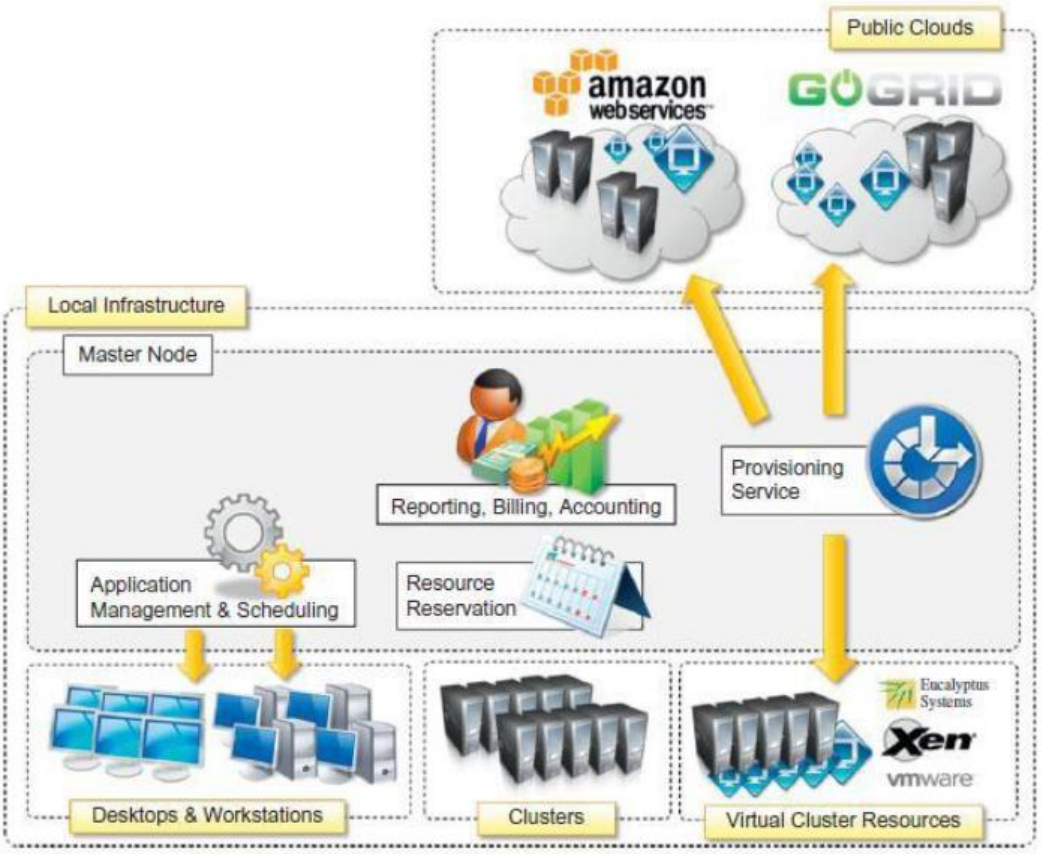


FIGURE 5.7

Hybrid cloud deployment.

An overview of this deployment is presented in Figure 5.7. This scenario constitutes the most complete deployment for Aneka that is able to leverage all the capabilities of the framework:

•Dynamic Resource Provisioning
•Resource Reservation
•Workload Partitioning (Scheduling)
•Accounting, Monitoring, and Reporting

In a hybrid scenario, heterogeneous resources can be used for different purposes. As we discussed in the case of a private cloud deployment, desktop machines can be reserved for low priority work- load outside the common working hours. The majority of the applications will be executed on work- stations and clusters, which are the nodes that are constantly connected to the Aneka Cloud. Any additional computing capability demand can be primarily addressed by the local virtualization facilities, and if more computing power is required, it is possible to leverage external IaaS providers.

b. Explain Aneka SDK in detail. **(10 Marks)**

Aneka provides APIs for developing applications on top of existing programming models, implementing new programming models, and developing new services to integrate into the Aneka Cloud.

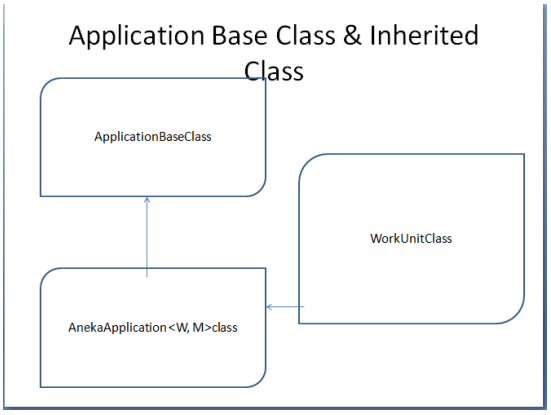
The SDK provides support for both programming models and services by

- The Application Model
- The Service Model.

Application Model

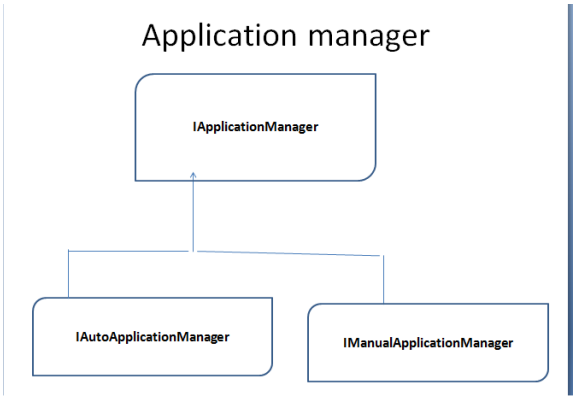
• The Application Model covers the development of applications and new programming models
• It Consists of Application Class & Application Manager
• Application Class – Provide user/developer view about distributed applications of the Aneka cloud
• Application Manager – Are Internal components that control and monitor the execution of Aneka clouds

The Application Class can be represented by following class diagram



Note: All the Aneka Application<W,M> class where W stands for Worker and M stands for Manager is inherited from base class and all Manual services are represented by *WorkUnitClass*. In addition, there are two other classes in Application Class representation viz: Configuration Class and Application Data Class

The Application manager is represented with following class diagram:



Also, the table given below summarizes Application Class, the programming models supported and work units assigned to them.

Table 5.1 Aneka's Application Model Features

Category	Description	Base Application Type	Work Units?	Programming Models
Manual	Units of work are generated by the user and submitted through the application.	*AnekaApplication < W,M >* *IManualApplicationManager < W >* *ManualApplicationManager < W >*	Yes	Task Model Thread Model Parameter Sweep Model
Auto	Units of work are generated by the runtime infrastructure and managed internally.	*ApplicationBase < M >* *IAutoApplicationManager*	No	*MapReduce* Model

The Service Model defines the general infrastructure for service development.

The Aneka Service Model defines the basic requirements to implement a service that can be hosted in an Aneka Cloud. The container defines the runtime environment in which services are hosted. Each service that is hosted in the container must use *IService* interface, which exposes the following methods and properties:

- Name and status
- Control operations such as *Start*, *Stop*, *Pause*, and *Continue* methods
- Message handling by means of the *HandleMessage* method

Module 3

5.

a. What is thread? Explain the thread API's techniques for parallel computation. **(08 Marks)**

A thread identifies a single control flow, which is a logical sequence of instructions, within a process. By logical sequence of instructions, we mean a sequence of instructions that have been designed to be executed one after the other one.

Operating systems that support multithreading identify threads as the minimal building blocks for expressing running code.

Each process contains at least one thread but, in several cases, is composed of many threads having variable lifetimes. Threads within the same process share the memory space and the execution context.

In a multitasking environment the operating system assigns different time slices to each process and interleaves their execution. The process of temporarily stopping the execution of one process, saving all the information in the registers, and replacing it with the information related to another process is known as a context switch.

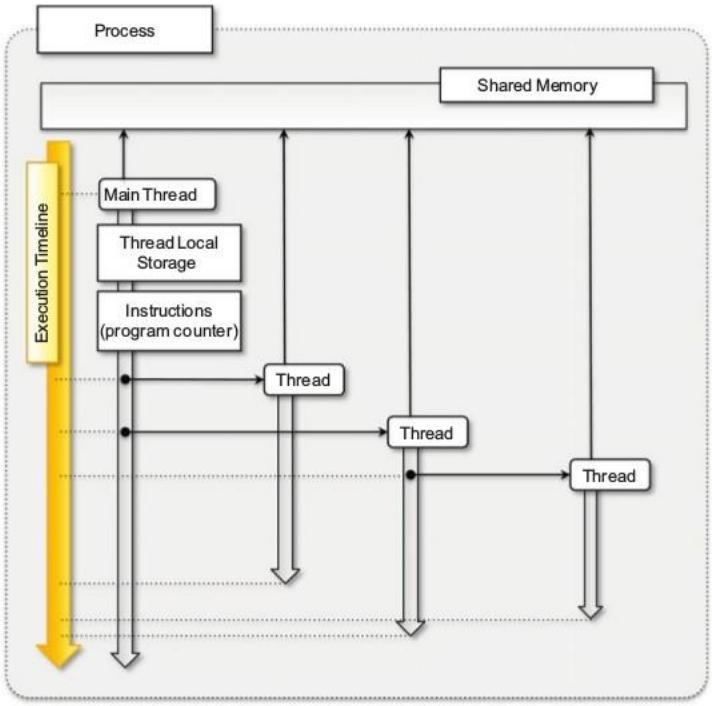


FIGURE 6.2

The relationship between processes and threads.

Figure 6.2 provides an overview of the relation between threads and processes and a simplified representation of the runtime execution of a multithreaded application. A running program is identified by a process, which contains at least one thread, also called the main thread. Such a thread is implicitly created by the compiler or the runtime environment executing the program. This thread is likely to last for the entire lifetime of the process and be the origin of other threads, which in general exhibit a shorter duration. As main threads, these threads can spawn other threads. There is no difference between the main thread and other threads created during the process lifetime. Each of them has its own local storage and a sequence of instructions to execute, and they all share the memory space allocated for the entire process. The execution of the process is considered terminated when all the threads are completed.

Thread APIs

1 POSIX Threads

2 Threading support in java and .NET

1 POSIX Threads

Portable Operating System Interface for Unix (POSIX) is a set of standards related to the application programming interfaces for a portable development of applications over the

Unix operating system flavors. Standard POSIX 1.c (IEEE Std 1003.1c-1995) addresses the implementation of threads and the functionalities that should be available for application programmers to develop portable multithreaded applications.

Important to remember from a programming point of view is the following:

•A thread identifies a logical sequence of instructions.

•A thread is mapped to a function that contains the sequence of instructions to execute.

•A thread can be created, terminated, or joined.

•A thread has a state that determines its current condition, whether it is executing, stopped,terminated, waiting for I/O, etc.

•The sequence of states that the thread undergoes is partly determined by the operating system

scheduler and partly by the application developers.

•Threads share the memory of the process, and since they are executed concurrently, they needsynchronization structures.

•Different synchronization abstractions are provided to solve different synchronization problems.

2 Threading support in java and .NET

Languages such as Java and C# provide a rich set of functionalities for multithreaded programming by using an object-oriented approach. both Java and .NET execute code on top of a virtual machine, the APIs exposed by the libraries refer to managed or logical threads. These are mapped to physical threads.

Both Java and .NET provide class Thread with the common operations on threads: start, stop, suspend, resume, abort, sleep, join, and interrupt. Start and stop/abort are used to control the lifetime of the thread instance. Suspend and resume are used to programmatically pause and then continue the execution of a thread. Sleep operation allows pausing the execution of a thread for a predefined period of time. Join operation that makes one thread wait until another thread is completed. Waiting states can be interrupted by using the interrupt operation.

b. Differentiate Aneka thread with local thread with diagram. **(06 Marks)**

The pictures of Aneka thread model and the related normal thread model are given below:

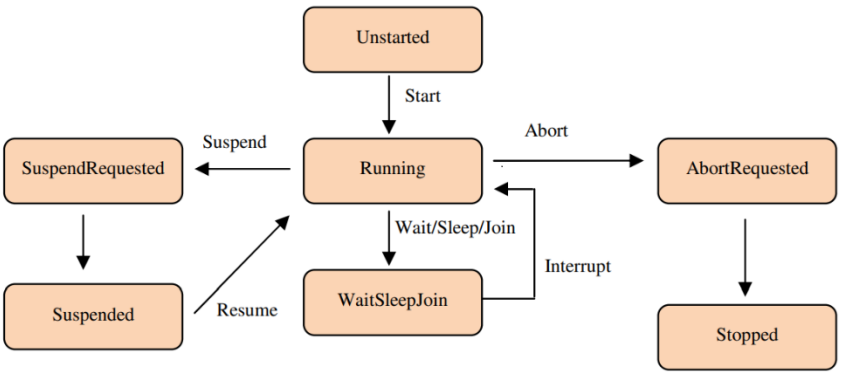


Figure 4: The life cycle of local threads in .Net

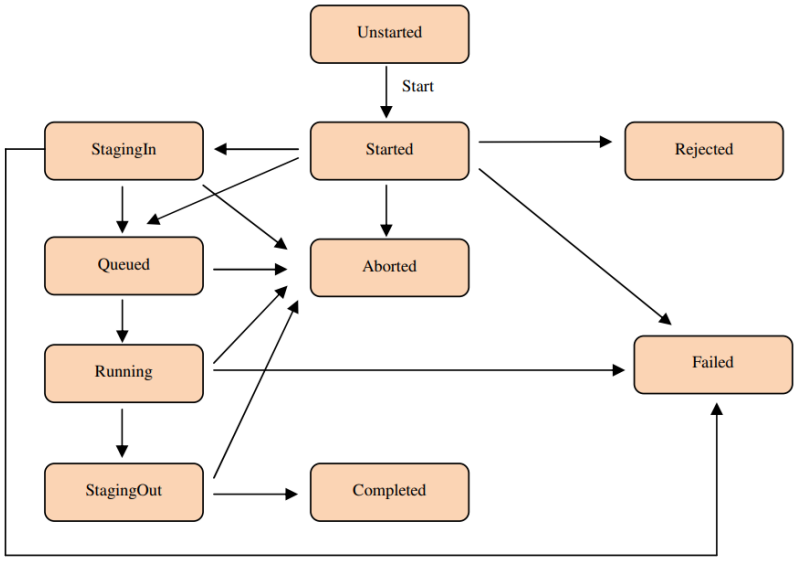


Figure 5: The life cycle of AnekaThreads

Limitations of Aneka Thread model

• Even though a distributed facility can dramatically increase the degree of parallelism of applications, its use comes with a cost in term of application design and performance.
• For example, since the different units of work are not executing within the same process space but on different nodes both the code and the data needs to be moved to a different execution context.
• the same happens for results that need to be collected remotely and brought back to the master process.
• Moreover, if there is any communication among the different workers it is necessary to redesign the communication model eventually by leveraging the APIs provided by the middleware if any.
• In other words, the transition from a single process multi-threaded execution to a distributed execution is not transparent and application redesign and re-implementation are often required.
• The amount of effort required to convert an application often depends on the facilities offered by the middleware managing the distributed infrastructure.
• Aneka, as a middleware for managing clusters, Grids, and Clouds, provides developers with advanced capabilities for implementing distributed applications.
• In particular, it takes traditional thread programming a step further. It lets you write multi-threaded applications in the traditional way, with the added twist that each of these threads can now be executed outside the parent process and on a separate machine.
• In reality, these "threads" are independent processes executing on different nodes, and do not share memory or other resources, but they allow you to write applications using the same thread constructs for concurrency and synchronization as with traditional threads.
• Aneka threads, as they are called, let you easily port existing multi-threaded compute intensive applications to distributed versions that can run faster by utilizing multiple machines simultaneously, with a minimum conversion effort.

c. Explain the programming applications with Aneka thread. **(06 Marks)**

To show how it is possible to quickly port multithreaded application to Aneka threads, we provide a distributed implementation of the previously discussed examples for local threads. Aneka threads application model.

The Thread Programming Model is a programming model in which the programmer creates the units of work as Aneka threads. Therefore, it is necessary to utilize the AnekaApplicatioN<W,M> class, which is the application reference class for all the programming models.

The Aneka APIs support different programming models through template

specialization. Hence, to develop distributed applications with Aneka threads, it is necessary to specialize the template type as follows: AnekaApplication<AnekaThread, ThreadManager>

These two types are defined in the Aneka.Threading namespace noted in the Aneka. Threading.dll library of the Aneka SDK. Another important component of the application model is the Configuration class, which is defined in the Aneka.Entity namespace (Aneka.dll). This class contains a set of properties that allow the application class to configure its interaction with the middleware, such as the address of the Aneka index service, which constitutes the main entry point of Aneka Clouds.

Listing 6.4 demonstrates how to create a simple application instance and configure it to connect to an Aneka Cloud whose index service is local.

```
// namespaces containing types of common use using System;
using System.Collections.Generic;
// common Aneka namespaces. using Aneka;
using Aneka.Util; using Aneka.Entity;
// Aneka Thread Programming Model user classes using Aneka.Threading;
///Creates an instance of the Aneka Application configured to use Thread Programming
    Model.
/// <returns>Application instance.</returns>
private AnekaApplication<AnekaThread,ThreadManager> CreateApplication();
{
Configuration conf =new Configuration();
// this is the common address and port of a local installation of Aneka Cloud.
    conf.SchedulerUri = newUri("tcp://localhost:9090/Aneka"); conf.Credentials
    =newUserCredentials("Administrator", string.Empty);
// we will not need support for file transfer, hence we optimize the
// application in order to not require any file transfer service. conf.UseFileTransfer = false;
// we do not need any other configuration setting
// we create the application instance and configure it.
    AnekaApplication<AnekaThread,ThreadManager> app =
new AnekaApplication<AnekaThread,ThreadManager>(conf); return app;
}
```

LISTING 6.4 Application Creation and Configuration.

Listing 6.5 provides a very simple example of how to create Aneka threads.

```
// ..... continues from the previous listing
///Thread worker method private void WorkerMethod()
{
// .....
}
///Creates a collection of threads that are executed in the context of given application.
/// <param name="app">>Application instance.</param>
private void CreateThreads(AnekaApplication<AnekaThread,ThreadManager> app);
{
// creates a delegate to the method to execute inside the threads.
ThreadStart worker = newThreadStart(this.WorkerMethod);
// iterates over a loop and creates ten threads.
for(int i=0; i<10; i++)
{
AnekaThread thread = new AnekaThread(worker, app); thread.Start();
}
}
```

OR

6.

a. Explain the features of workflow applications with task dependencies. **(10 Marks)**

Workflow applications are characterized by a collection of tasks that exhibit dependencies among them. Such dependencies, which are mostly data dependencies

determine the way in which the applications are scheduled as well as where they are scheduled.

What is a workflow?

A workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant (a resource; human or machine) to another for action, according to a set of procedural rules.

The concept of workflow as a structured execution of tasks that have dependencies on each other has demonstrated itself to be useful for expressing many scientific experiments and gave birth to the idea of scientific workflow. In the case of scientific workflows, the process is identified by an application to run, the elements that are passed among participants are mostly tasks and data, and the participants are mostly computing or storage nodes. The set of procedural rules is defined by a workflow definition scheme that guides the scheduling of the application.

A scientific workflow generally involves data management, analysis, simulation, and middleware supporting the execution of the workflow. A scientific workflow is generally expressed by a directed acyclic graph (DAG), which defines the dependencies among tasks or operations. The nodes on the DAG represent the tasks to be executed in a workflow application; the arcs connecting the nodes identify the dependencies among tasks and the data paths that connect the tasks.

The most common dependency that is realized through a DAG is data dependency, which means that the output files of a task constitute the input files of another task. The DAG in Figure 7.6 describes a sample Montage workflow. Montage is a toolkit for assembling images into mosaics; it has been specially designed to support astronomers in composing the images taken from different telescopes or points of view into a coherent image. The workflow depicted in Figure 7.6 describes the general process for composing a mosaic; the labels on the right describe the different tasks that have to be performed to compose a mosaic.

In the case presented in the diagram, a mosaic is composed of seven images. For each of the image files, the following process has to be performed: image file transfer, reprojection, calculation of the difference, and common plane placement. Therefore, each of the images can be processed in parallel for these tasks.

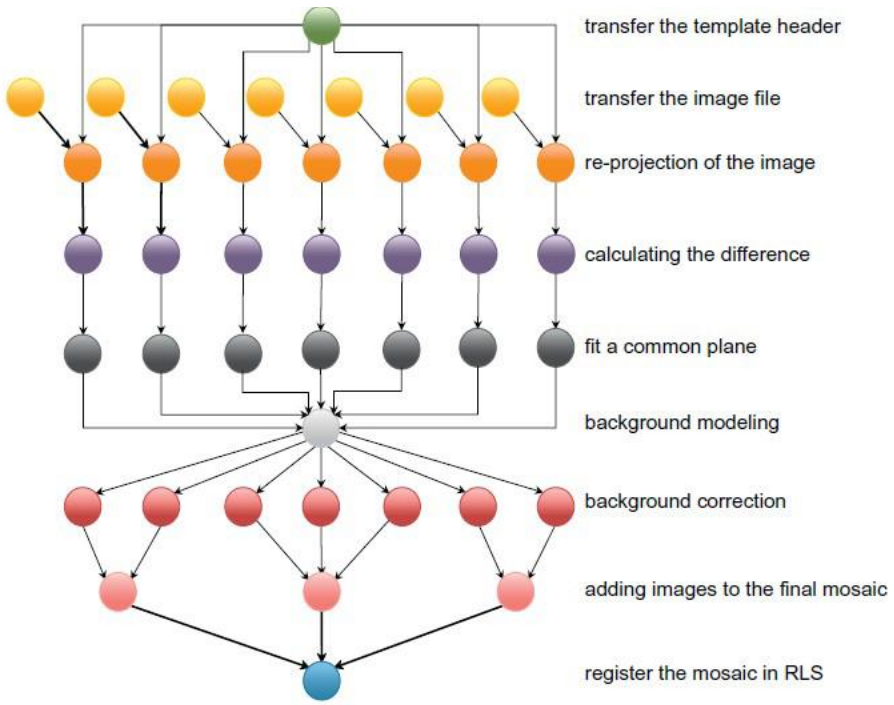


FIGURE 7.6

Sample Montage workflow.

2 Workflow technologies

Business-oriented computing workflows are defined as compositions of services. There are specific languages and standards for the definition of workflows, such as Business Process Execution Language (BPEL). An abstract reference model for a workflow management system, as depicted in Figure 7.7. Design tools allow users to visually compose a workflow

application.

This specification is stored in the form of an XML document based on a specific workflow language and constitutes the input of the workflow engine, which controls the execution of the workflow by leveraging a distributed infrastructure. The workflow engine is a client- side component that might interact directly with resources or with one or several middleware components for executing the workflow.

b. List and explain popular framework for task computing. **(10 Marks)**

Some popular software systems that support the task-computing framework are:

1. Condor
2. Globus Toolkit
3. Sun Grid Engine (SGE)
4. BOINC
5. Nimrod/G

Architecture of all these systems is similar to the general reference architecture depicted in Figure 7. They consist of two main components: a scheduling node (one or more) and worker nodes. The organization of the system components may vary.

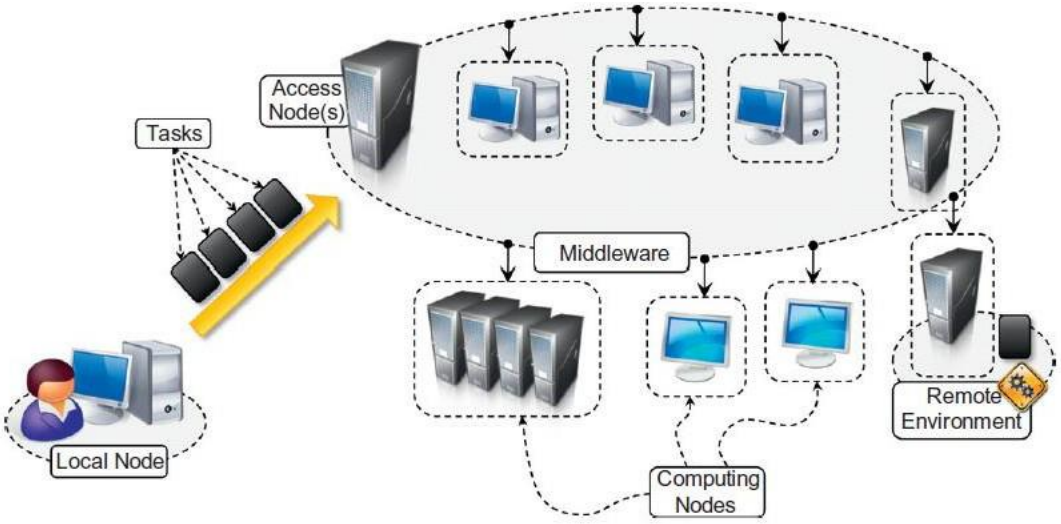


FIGURE 7.1

Task computing scenario.

1. Condor

Condor is the most widely used and long-lived middleware for managing clusters, idle workstations, and a collection of clusters. Condor supports features of batch-queuing systems along with the capability to checkpoint jobs and manage overload nodes. It provides a powerful job resource-matching mechanism, which schedules jobs only on resources that have the appropriate runtime environment. Condor can handle both serial and parallel jobs on a wide variety of resources.

It is used by hundreds of organizations in industry, government, and academia to manage infrastructures. Condor-G is a version of Condor that supports integration with grid computing resources, such as those managed by Globus.

2. Globus Toolkit

The Globus Toolkit is a collection of technologies that enable grid computing. It provides a comprehensive set of tools for sharing computing power, databases, and other services across corporate, institutional, and geographic boundaries. The toolkit features software services, libraries, and tools for resource monitoring, discovery, and management as well as security and file management. The toolkit defines a collection of interfaces and protocol for interoperation that enable different systems to integrate with each other and expose resources outside their boundaries.

3. Sun Grid Engine (SGE)

Sun Grid Engine (SGE), now Oracle Grid Engine, is middleware for workload and distributed resource management. Initially developed to support the execution of jobs on clusters, SGE integrated additional capabilities and now is able to manage heterogeneous resources and constitutes middleware for grid computing.

It supports the execution of parallel, serial, interactive, and parametric jobs and features advanced scheduling capabilities such as budget-based and group- based scheduling, scheduling applications that have deadlines, custom policies, and advance reservation.

4. BOINC

Berkeley Open Infrastructure for Network Computing (BOINC) is framework for volunteer and grid computing. It allows us to turn desktop machines into volunteer computing nodes that are leveraged to run jobs when such machines become inactive. BOINC supports job check pointing and duplication. BOINC is composed of two main components: the BOINC server and the BOINC client. The BOINC server is the central node that keeps track of all the available resources and scheduling jobs.

The BOINC client is the software component that is deployed on desktop machines and that creates the BOINC execution environment for job submission. BOINC systems can be easily set up to provide more stable support for job execution by creating computing grids with dedicated machines. When installing BOINC clients, users can decide the application project to which they want to donate the CPU cycles of their computer. Currently several projects, ranging from medicine to astronomy and cryptography, are running on the BOINC infrastructure.

5. Nimrod/G

Tool for automated modeling and execution of parameter sweep applications over global computational grids. It provides a simple declarative parametric modeling language for expressing parametric experiments. It uses novel resource management and scheduling algorithms based on economic principles.

It supports deadline- and budget-constrained scheduling of applications on distributed grid resources to minimize the execution cost and at the same deliver results in a timely manner. It has been used for a very wide range of applications over the years, ranging from quantum chemistry to policy and environmental impact.

Module 4

7.

a. What is Data Intensive Computing? List all the challenges in data intensive computing and explain each in detail. **(10 Marks)**

Data-intensive computing is concerned with production, manipulation, and analysis of large-scale data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond. Dataset is commonly used to identify a collection of information elements that is relevant to one or more applications. Datasets are often maintained in repositories, which are infrastructures supporting the storage, retrieval, and indexing of large amounts of information.

To facilitate classification and search, relevant bits of information, called metadata, are attached to datasets. Data-intensive computations occur in many application domains. Computational science is one of the most popular ones. People conducting scientific simulations and experiments are often keen to produce, analyze, and process huge volumes of data. Hundreds of gigabytes of data are produced every second by telescopes mapping the sky; the collection of images of the sky easily reaches the scale of petabytes over a year.

Bioinformatics applications mine databases that may end up containing terabytes of data. Earthquake simulators process a massive amount of data, which is produced as a result of recording the vibrations of the Earth across the entire globe.

Characterizing data-intensive computations
Challenges ahead

Historical perspective

1 The early age: high-speed wide-area networking 2 Data grids

3 Data clouds and “Big Data”

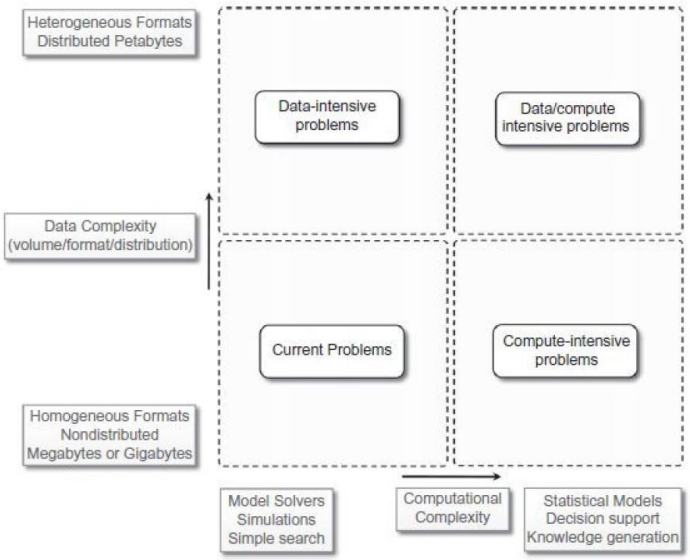


FIGURE 8.1
Data-intensive research issues.

Characterizing data-intensive computations

Data-intensive applications deals with huge volumes of data, also exhibit compute-intensive properties. Figure 8.1 identifies the domain of data-intensive computing in the two upper quadrants of the graph. Data-intensive applications handle datasets on the scale of multiple terabytes and petabytes.

b. Explain Amazon Dynamo architecture that support data intensive applications. **(10 Marks)**

The main goal of Dynamo is to provide an incrementally scalable and highly available storage system. This goal helps in achieving reliability at a massive scale, where thousands of servers and network components build an infrastructure serving 10 million requests per day. Dynamo provides a simplified interface based on get/put semantics, where objects are stored and retrieved with a unique identifier (key).

The architecture of the Dynamo system, shown in Figure 8.3, is composed of a collection of storage peers organized in a ring that shares the key space for a given application. The key space is partitioned among the storage peers, and the keys are replicated across the ring, avoiding adjacent peers. Each peer is configured with access to a local storage facility where original objects and replicas are stored.

Each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.

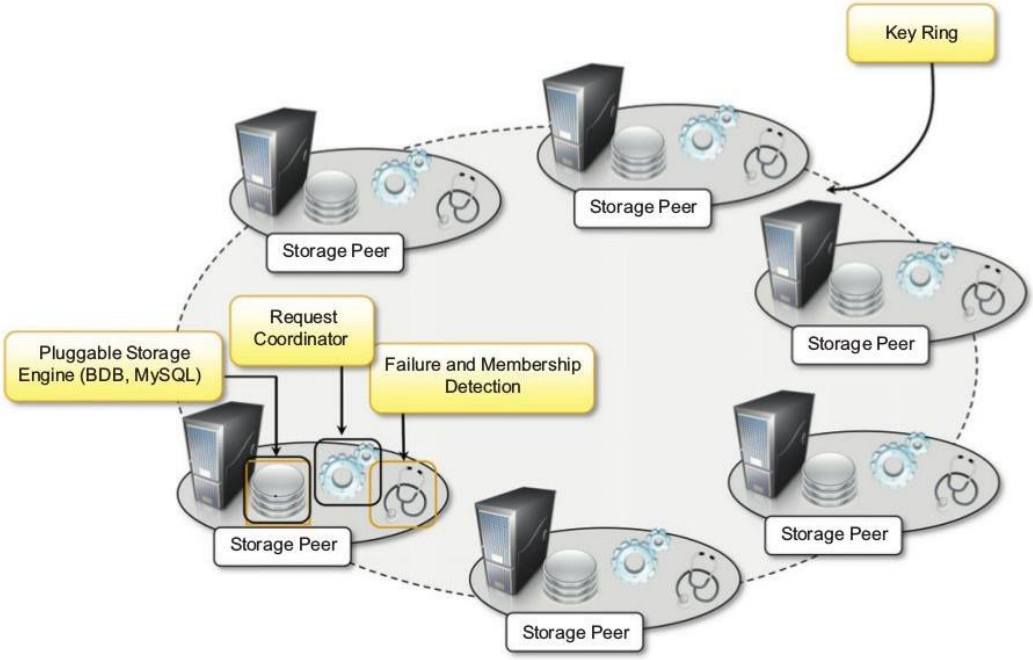


FIGURE 8.3

Amazon Dynamo architecture.

OR

8.

a. Explain the following: i) IBM General Parallel File System ii) Google File System iii) Amazon Simple Storage Service. **(10 Marks)**

i) IBM General Parallel File System

GPFS is the high-performance distributed file system developed by IBM that provides support for the RS/6000 supercomputer and Linux computing clusters. GPFS is a multiplatform distributed file system built over several years of academic research and provides advanced recovery mechanisms. GPFS is built on the concept of shared disks, in which a collection of disks is attached to the file system nodes by means of some switching fabric. The file system makes this infrastructure transparent to users and stripes large files over the disk array by replicating portions of the file to ensure high availability.

ii) Google File System

GFS is the storage infrastructure that supports the execution of distributed applications in Google's computing cloud.

GFS is designed with the following assumptions:

1. The system is built on top of commodity hardware that often fails.
2. The system stores a modest number of large files; multi-GB files are common and should be treated efficiently, and small files must be supported, but there is no need to optimize for that.
3. The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.
4. The workloads also have many large, sequential writes that append data to files.
5. High-sustained bandwidth is more important than low latency.

The architecture of the file system is organized into a single master, which contains the metadata of the entire file system, and a collection of chunk servers, which provide storage space. From a logical point of view the system is composed of a collection of software daemons, which implement either the master server or the chunk server.

iii) Amazon Simple Storage Service

Amazon S3 is the online storage service provided by Amazon. The system offers a flat storage space organized into buckets, which are attached to an Amazon Web Services (AWS)

account. Each bucket can store multiple objects, each identified by a unique key. Objects are identified by unique URLs and exposed through HTTP, thus allowing very simple get-put semantics.

b. Design and implement an application for log parsing, Mapper and Reducer with Aneka map Reduce. **(10 Marks)**

Aneka components produce a lot of information that is stored in the form of log files.

In this example, we parse these logs to extract useful information about the execution of applications and the usage of services in the Cloud.

The entire framework leverages the log4net library for collecting and storing the log information. Some examples of formatted log messages are:

15 Mar 2011 10:30:07 DEBUGSchedulerService: . . . HandleSubmitApplicationSchedulerService: . . .
15 Mar 2011 10:30:07 INFOSchedulerService: Scanning candidate storage . . .
15 Mar 2011 10:30:10 INFOAdded [WU: 51d55819-b211-490f-b185-8a25734ba705, 4e86fd02. . .
15 Mar 2011 10:30:10 DEBUGStorageService:NotifySchedulerSending FileTransferMessage. . .
15 Mar 2011 10:30:10 DEBUGIndependentSchedulingService:QueueWorkUnitQueueing. . .
15 Mar 2011 10:30:10 INFOAlgorithmBase::AddTasks[64] Adding 1 Tasks 15 Mar 2011 10:30:10 DEBUGAlgorithmBase:FireProvisionResourcesProvision

Possible information that we might want to extract from such logs is the following:

- The distribution of log messages according to the level
- The distribution of log messages according to the components

This information can be easily extracted and composed into a single view by creating Mapper tasks that count the occurrences of log levels and component names and emit one simple key-value pair in the form (level-name, 1) or (component-name, 1) for each of the occurrences. The Reducer task will simply sum up all the key-value pairs that have the same key. For both problems, the structure of the map and reduce functions will be the following:

map: (long; string) => (string; long)
reduce: (long; string) => (string; long)

The Mapper class will then receive a key-value pair containing the position of the line inside the file as a key and the log message as the value component. It will produce a key-value pair containing a string representing the name of the log level or the component name and 1 as value. The Reducer class will sum up all the key-value pairs that have the same name.

Module 5

9.

a. Describe the core components of Google App Engine. **(10 Marks)**

Google AppEngine is a PaaS implementation

Distributed and scalable runtime environment that leverages Google's distributed infrastructure to scale out applications.

Architecture and core concepts

AppEngine is a platform for developing scalable applications accessible through the Web. Figure 9.2. The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services.

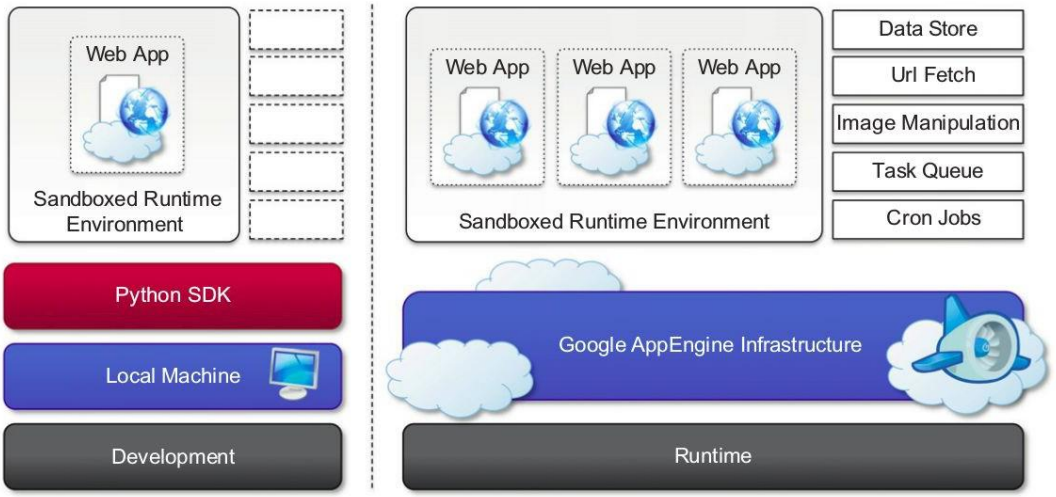


FIGURE 9.2

Google AppEngine platform architecture.

1 Infrastructure

AppEngine hosts Web applications, and its primary function is to serve users requests efficiently. AppEngine's infrastructure takes advantage of many servers available within Google data centers. For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources or redirects the request to an existing server. The infrastructure is also responsible for monitoring application performance and collecting statistics on which the billing is calculated.

2 Runtime environment

The runtime environment represents the execution context of applications hosted on AppEngine. Sandboxing- One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context in which it can execute without causing a threat to the server and without being influenced by other applications. In other words, it provides applications with a sandbox. If an application tries to perform any operation that is considered potentially harmful, an exception is thrown and the execution is interrupted. Supported runtimes- Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go. AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP), and the applications interact with the environment by using the Java Servlet standard.

Support for Python is provided by an optimized Python 2.5.2 interpreter. As with Java, the runtime environment supports the Python standard library. Developers can use a specific Python Web application framework, called webapp, simplifying the development of Web applications. The Go runtime environment allows applications developed with the Go programming language to be hosted and executed in AppEngine. Currently the release of Go that is supported by AppEngine is r58.1. The SDK includes the compiler and the standard libraries for developing applications in Go and interfacing it with AppEngine services.

3 Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data. Static file servers- Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application or data files. DataStore- DataStore is a service that allows developers to store semi-structured data. The service is designed to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key. DataStore imposes less constraint on the regularity of the data but, at the same time, does not implement some of the features of the relational model. The underlying infrastructure of DataStore is based on Bigtable, a redundant, distributed, and semistructured data store that organizes data in the form of tables.

DataStore provides high-level abstractions that simplify interaction with Bigtable. Developers define their data in terms of entity and properties, and these are persisted and maintained by the service into tables in Bigtable. DataStore also provides facilities for creating indexes on data and to update data within the context of a transaction. Indexes are used to support and speed up queries. A query can return zero or more objects of the same kind or simply the corresponding keys.

4 Application services

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in

Web applications UrlFetch - The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service. Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal request- handling cycle of the application.

UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed applications. MemCache- This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging- AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients. AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address.

Account management- AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts. Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates.

5 Compute services

AppEngine offers additional services such as Task Queues and Cron Jobs that simplify the execution of computations. Task queues- A task is defined by a Web request to a given URL, and the queue invokes the request handler by passing the payload as part of the Web request to the handler. It is the responsibility of the request handler to perform the "task execution," which is seen from the queue as a simple Web request. Cron jobs- the required operation needs to be performed at a specific time of the day, which does not coincide with the time of the Web request. In this case, it is possible to schedule the required operation at the desired time by using the Cron Jobs service.

b. Explain the Windows Azure platform architecture. **(10 Marks)**

Microsoft Windows Azure is a cloud operating system built on top of Microsoft datacenters' infrastructure and provides developers with a collection of services for building applications with cloud technology.

Services range from compute, storage, and networking to application connectivity, access control, and business intelligence. Figure 9.3 provides an overview of services provided by Azure. These services can be managed and controlled through the Windows Azure Management Portal, which acts as an administrative console for all the services.

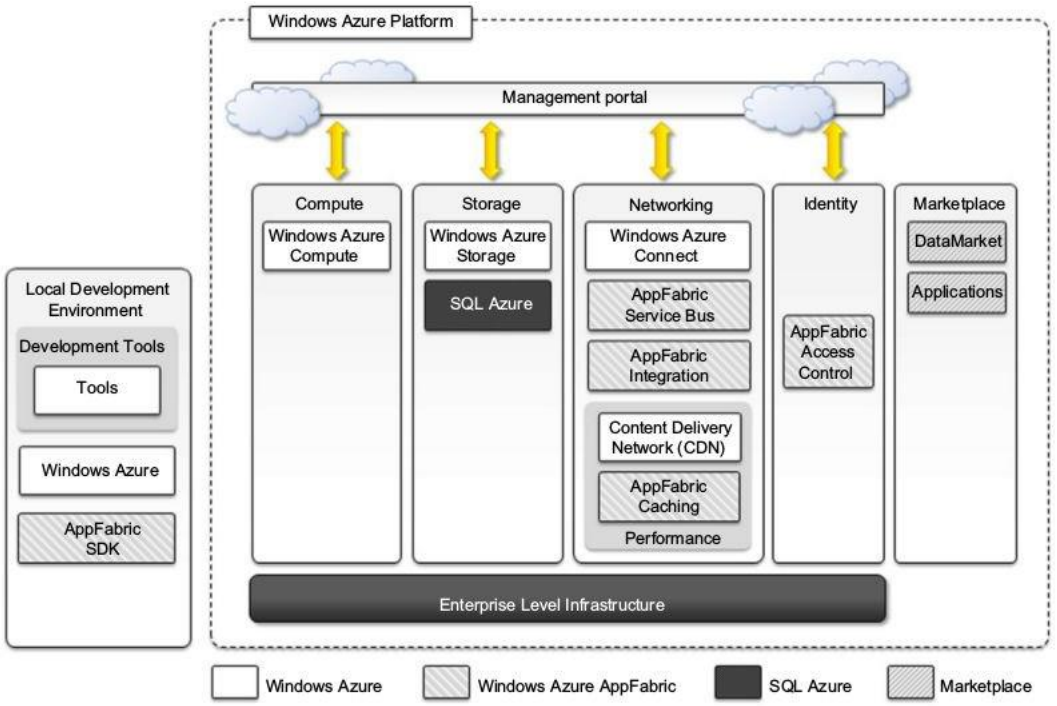


FIGURE 9.3

Microsoft Windows Azure Platform Architecture.

Azure core concepts

The Windows Azure platform is made up of a foundation layer and a set of developer services that can be used to build scalable applications. These services cover compute, storage, networking, and identity management, which are tied together by middleware called AppFabric.

1 Compute services

Compute services are the core components of Microsoft Windows Azure, and they are delivered by means of the abstraction of roles. Currently, there are three different roles: Web role, Worker role, and Virtual Machine (VM) role.

Web role - The Web role is designed to implement scalable Web applications. Web roles represent the units of deployment of Web applications within the Azure infrastructure. They are hosted on the IIS 7 Web Server. Since version 3.5, the .NET technology natively supports Web roles; developers can directly develop their applications in Visual Studio, test them locally, and upload to Azure. Web roles can be used to run and scale PHP Web applications on Azure (CGI Web Role).

Worker role - Worker roles are designed to host general compute services on Azure. They can be used to quickly provide compute power or to host services that do not communicate with the external world through HTTP. A common practice for Worker roles is to use them to provide background processing for Web applications developed with Web roles.

A Worker role runs continuously from the creation of its instance until it is shut down. The Azure SDK provides developers with convenient APIs and libraries that allow connecting the role with the service provided by the runtime and easily controlling its startup as well as being notified of changes in the hosting environment.

Virtual machine role - The Virtual Machine role allows developers to control computing stack of their compute service by defining a custom image of the Windows Server 2008 R2 operating system and all the service stack required by their applications. The Virtual Machine role is based on the Windows Hyper-V virtualization technology.

Developers can image a Windows server installation complete with all the required applications and components, save it into a Virtual Hard Disk (VHD).

2 Storage services

Compute resources are equipped with local storage in the form of a directory on the local file system. Windows Azure provides different types of storage solutions that complement compute services with a more durable and redundant option.

Blobs

Azure allows storing large amount of data in the form of binary large objects (BLOBs) by means of the blobs service.

Block blobs - Block blobs are composed of blocks and are optimized for sequential access;

therefore they are appropriate for media streaming. Currently, blocks are of 4 MB, and a single block blob can reach 200 GB in dimension.

Page blobs - Page blobs are made of pages that are identified by offset from beginning of blob. A page blob can be split into multiple pages or constituted of single page. This type of blob is optimized for random access and can be used to host data different from streaming. Maximum dimension of page blob can be 1TB.

Azure drive

Page blobs can be used to store an entire file system in the form of a single Virtual Hard Drive (VHD) file. This can then be mounted as a part of the NTFS file system by Azure compute resources, thus providing persistent and durable storage.

Tables

Tables constitute a semistructured storage solution, allowing users to store information in the form of entities with a collection of properties. Entities are stored as rows in the table and are identified by a key, which also constitutes the unique index built for the table. Users can insert, update, delete, and select a subset of the rows stored in the table.

Currently, a table can contain up to 100 TB of data, and rows can have up to 255 properties, with a maximum of 1 MB for each row. The maximum dimension of a row key and partition keys is 1 KB.

Queues

Queue storage allows applications to communicate by exchanging messages through durable queues, thus avoiding lost or unprocessed messages. Applications enter messages into a queue, and other applications can read them in a first-in, first-out (FIFO) style.

3 Core infrastructure: AppFabric

AppFabric is a comprehensive middleware for developing, deploying, and managing applications on the cloud or for integrating existing applications with cloud services.

AppFabric implements an optimized infrastructure supporting scaling out and high availability; sandboxing and multitenancy; state management; and dynamic address resolution and routing.

Access control - AppFabric provides the capability of encoding access control to resources in Web applications and services into a set of rules that are expressed outside the application code base. These rules give a great degree of flexibility in terms of the ability to secure components of the application and define access control policies for users and groups.

Service bus - Service Bus constitutes the messaging and connectivity infrastructure provided with AppFabric for building distributed and disconnected applications The service is designed to allow transparent network traversal and to simplify the development of loosely coupled applications, without renouncing security and reliability and letting developers focus on the logic of the interaction rather than the details of its implementation. Service Bus allows services to be available by simple URLs, which are untied from their deployment location.

Azure cache - Windows Azure provides a set of durable storage solutions that allow applications to persist their data. Azure Cache is a service that allows developers to quickly access data persisted on Windows Azure storage or in SQL Azure. The service implements a distributed in-memory cache of which the size can be dynamically adjusted by applications according to their needs.

4 Other services

Windows Azure virtual network - Networking services for applications are offered under the name Windows Azure Virtual Network, which includes Windows Azure Connect and Windows Azure Traffic Manager. Windows Azure Connect allows easy setup of IP-based network connectivity among machines hosted on the private premises and the roles deployed on the Azure Cloud. This service is particularly useful in the case of VM roles.

Windows Azure content delivery network - Windows Azure Content Delivery Network (CDN) is the content delivery network solution that improves the content delivery capabilities of Windows Azure Storage and several other Microsoft services, such as Microsoft Windows Update and Bing maps.

OR

10.

a. Discuss in detail the following media applications of cloud computing technologies.
i) Animoto ii) Maya Rendering with Aneka iii) Video encoding on cloud. **(12 Marks)**

i) Animoto

Animoto is the most popular example of media applications on the cloud. The Website provides users with a very straightforward interface for quickly creating videos out of images, music, and video fragments submitted by users. Users select a specific theme for a video, upload the photos and videos and order them in the sequence they want to appear, select the song for the music, and render the video. The process is executed in the background and the user is notified via email once the video is rendered.

A proprietary artificial intelligence (AI) engine, which selects the animation and transition effects according to pictures and music, drives the rendering operation. Users only have to define the storyboard by organizing pictures and videos into the desired sequence.

The infrastructure of Animoto is complex and is composed of different systems that all need to scale (Figure 10.8). The core function is implemented on top of the Amazon Web Services infrastructure. It uses Amazon EC2 for the Web front-end and worker nodes; Amazon S3 for the storage of pictures, music, and videos; and Amazon SQS for connecting all the components. The system's auto-scaling capabilities are managed by Rightscale, which monitors the load and controls the creation of new worker instances.

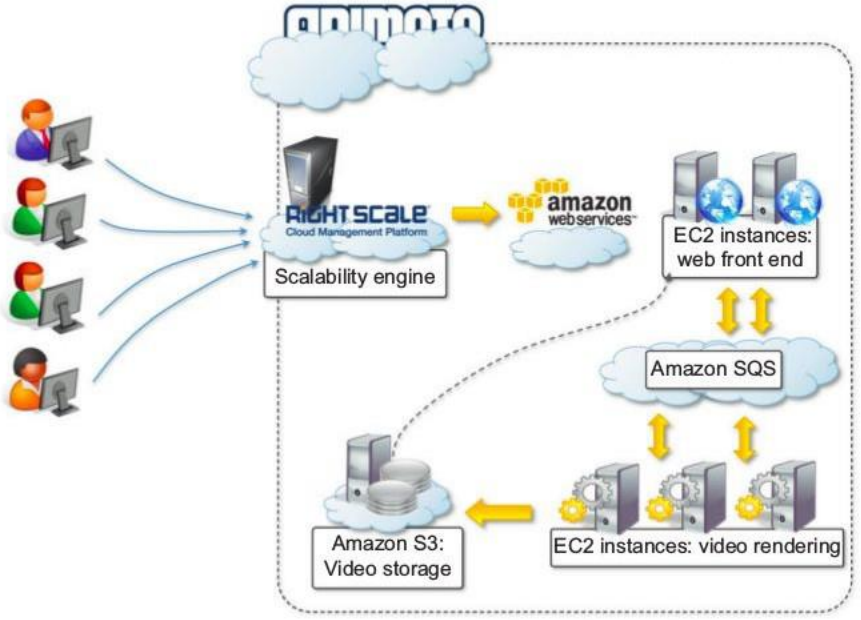


FIGURE 10.8

Animoto reference architecture.

ii) Maya Rendering with Aneka

A private cloud solution for rendering train designs has been implemented by the engineering department of GoFront group, a division of China Southern Railway (Figure 10.9). The department is responsible for designing models of high-speed electric locomotives, metro cars, urban transportation vehicles, and motor trains. The design process for prototypes requires high-quality, three-dimensional (3D) images. The analysis of these images can help engineers identify problems and correct their design.

Three-dimensional rendering tasks take considerable amounts of time, especially in the case of huge numbers of frames, but it is critical for the department to reduce the time spent in these iterations. This goal has been achieved by leveraging cloud computing technologies, which turned the network of desktops in the department into a desktop cloud managed by Aneka.

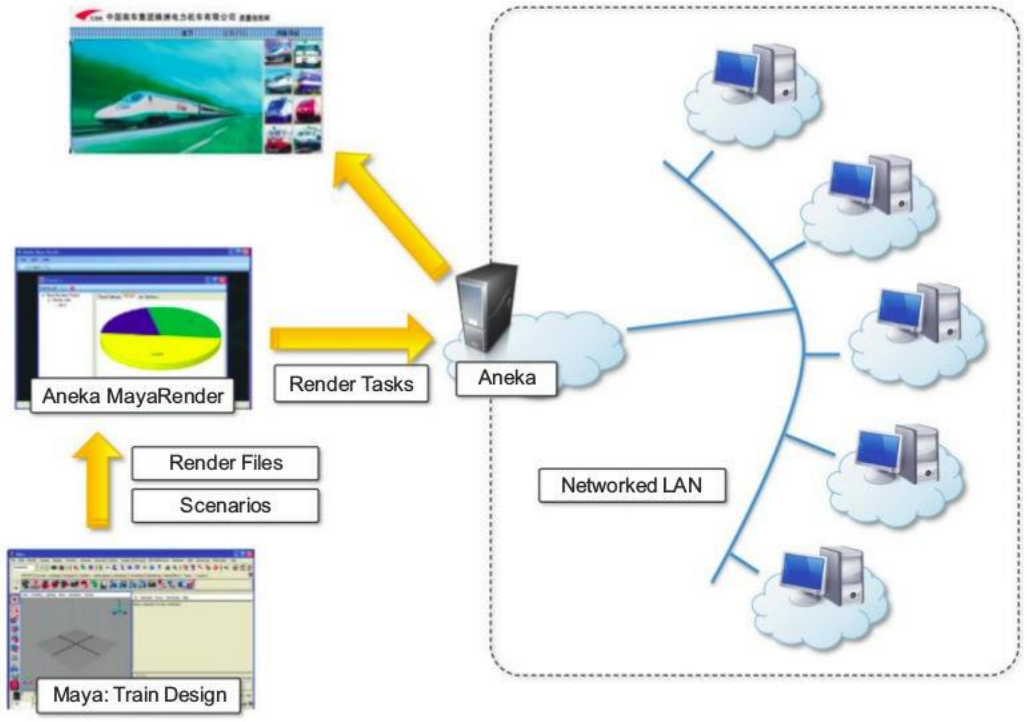


FIGURE 10.9

3D rendering on private clouds.

iii) Video encoding on cloud

Video encoding and transcoding are operations that can greatly benefit from using cloud technologies: They are computationally intensive and potentially require considerable amounts of storage.

Encoding.com is a software solution that offers video-transcoding services on demand and leverages cloud technology to provide both the horsepower required for video conversion and the storage for staging videos. The service integrates with both Amazon Web Services technologies (EC2, S3, and CloudFront) and Rackspace (Cloud Servers, Cloud Files, and Limelight CDN access).

To use the service, users have to specify the location of the video to transcode, the destination format, and the target location of the video. Encoding.com also offers other video-editing operations such as the insertion of thumbnails, watermarks, or logos. Moreover, it extends its capabilities to audio and image conversion.

b. Explain in detail about the application of cloud computing in satellite Image processing.

(08 Marks)

Geoscience applications collect, produce, and analyze massive amounts of geospatial and nonspatial data. As the technology progresses and our planet becomes more instrumented, volume of data that needs to be processed increases significantly. Geographic information system (GIS) applications capture, store, manipulate, analyze, manage, and present all types of geographically referenced data. Cloud computing is an attractive option for executing these demanding tasks and extracting meaningful information to support decision makers.

Satellite remote sensing generates hundreds of gigabytes of raw images that need to be further processed to become the basis of several different GIS products. This process requires both I/O and compute-intensive tasks. Large images need to be moved from a ground station's local storage to compute facilities, where several transformations and corrections are applied.

The system shown in Figure 10.4 integrates several technologies across the entire computing stack. A SaaS application provides a collection of services for such tasks as geocode generation and data visualization. At the PaaS level, Aneka controls the importing of data into the virtualized infrastructure and the execution of image-processing tasks that produce the desired outcome from raw satellite images. The platform leverages a Xen private cloud and the Aneka technology to dynamically provision the required resources.

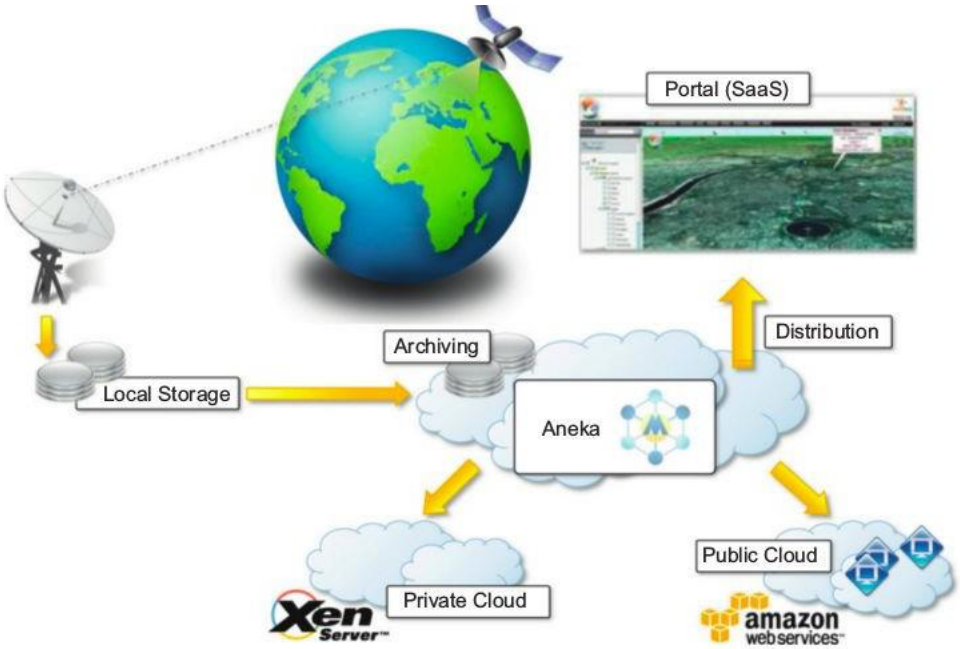


FIGURE 10.4

A cloud environment for satellite data processing.

* * * * *