1) **List and explain the common characteristics and advantages of Network centric computing Network Centric Content.**

*Network-centric computing and network-centric content share a number of characteristics:*

- ➔ Most applications are data-intensive. Computer simulation becomes a powerful tool for scientific research in virtually all areas of science, from physics, biology, and chemistry to archeology. Sophisticated tools for computer-aided design, such as Catia (Computer Aided Three-dimensional Interactive Application), are widely used in the aerospace and automotive industries. The widespread use of sensors contributes to increases in the volume of data. Multimedia applications are increasingly popular; the ever-larger media increase the load placed on storage, networking, and processing systems.
- ➔ Virtually all applications are network-intensive. Indeed, transferring large volumes of data requires high-bandwidth networks; parallel computing, computation steering, and data streaming are examples of applications that can only run efficiently on low-latency networks.
- ➔ The systems are accessed using thin clients running on systems with limited resources. In June 2011 Google released Google Chrome OS, designed to run on primitive devices and based on the browser with the same name.
- ➔ The infrastructure supports some form of workflow management. Indeed, complex computational tasks require coordination of several applications; composition of services is a basic tenet of Web 2.0.

*The advantages of network-centric computing and network-centric content paradigms are, at the same time, sources for concern;*

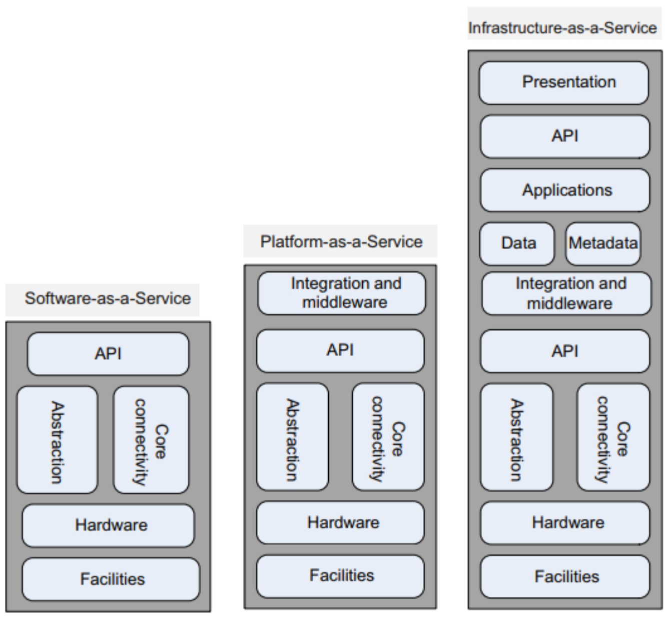
➔ Computing and communication resources (CPU cycles, storage, network bandwidth) are shared and resources can be aggregated to support data-intensive applications. Multiplexing leads to a higher resource utilization; indeed, when multiple applications share a system, their peak demands for resources are not synchronized and the average system utilization increases. On the other hand, the management of large pools of resources poses new challenges as complex systems are subject to phase transitions. New resource management strategies, such as self-organization, and decisions based on approximate knowledge of the state of the system must be considered. Ensuring quality-ofservice (QoS) guarantees is extremely challenging in such environments because total performance isolation is elusive.

➔ Data sharing facilitates collaborative activities. Indeed, many applications in science, engineering, and industrial, financial, and governmental applications require multiple types of analysis of shared data sets and multiple decisions carried out by groups scattered around the globe. Open software development sites are another example of such collaborative activities. Data sharing poses not only security and privacy challenges but also requires mechanisms for access control by authorized users and for detailed logs of the history of data changes.

➔ Cost reduction. Concentration of resources creates the opportunity to pay as you go for computing and thus eliminates the initial investment and reduces significantly the maintenance and operation costs of the local computing infrastructure.

➔ User convenience and elasticity, that is the ability to accommodate workloads with very large peakto-average ratios.

2) **Compare the three CC delivery models based on the limits of responsibility between a cloud user and the cloud service provider. Discuss the security and reliability provided by each model. Justify the difference with a neat diagram.**



**Software-as-a-Service (SaaS)**
gives the capability to use applications supplied by the service provider in a cloud infrastructure. The applications are accessible from various client devices through a thin-client interface such as a Web browser (e.g., Web-based email). The user does not manage or control the underlying cloud infrastructure, including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. SaaS is not suitable for applications that require real-time response or those for which data is not allowed to be hosted externally.

The most likely candidates for SaaS are applications for which:

➔ Many competitors use the same product, such as **email**.
➔ Periodically there is a significant peak in demand, such as **billing** and **payroll**.
➔ There is a need for Web or mobile access, such as mobile **sales management** software.
➔ There is only a short-term need, such as collaborative **software** for a project.

**Platform-as-a-Service (PaaS)**
gives the capability to deploy consumer-created or acquired applications using programming languages and tools supported by the provider. The user does not manage or control the underlying cloud infrastructure, including network, servers, operating systems, or storage. The user has control over the deployed applications and, possibly, over the application hosting environment configurations. Such services include session management, device integration, sandboxes, instrumentation and testing, contents management, knowledge management, and Universal Description, Discovery, and Integration (UDDI), a platform-independent Extensible Markup Language (XML)-based registry providing a mechanism to register and locate Web service applications.
PaaS is not particulary useful when the application must be portable, when proprietary programming languages are used, or when the underlaying hardware and software must be customized to improve the performance of the application. The major PaaS application areas are in software development where multiple developers and users collaborate and the deployment and testing services should be automated.

**Infrastructure-as-a-Service (IaaS)**
offers the capability to provision processing, storage, networks, and other fundamental computing resources; the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of some networking components, such as host firewalls. Services offered by this delivery model include: server hosting, Web servers, storage, computing hardware, operating systems, virtual instances, load balancing, Internet access, and bandwidth provisioning. The IaaS cloud computing delivery model has a number of characteristics, such as the fact that the resources are distributed and support dynamic scaling, it is based on a utility pricing model and variable cost, and the hardware is shared among multiple users. This cloud computing model is particulary useful when the demand is volatile and a new business needs computing resources and does not want to invest in a computing infrastructure or when an organization is expanding rapidly.

3) **List the reasons, why cloud computing could be successful, when other paradigms failed, and also mention the challenges faced by cloud computing.**

technological advances, a realistic system model, user convenience, and financial advantages. A nonexhaustive list of reasons for the success of cloud computing includes these points:

➔ Cloud computing is in a better position to exploit recent advances in software, networking, storage, and processor technologies. Cloud computing is promoted by large IT companies where these new technological developments take place, and these companies have a vested interest in promoting the new technologies.

➔ A cloud consists of a homogeneous set of hardware and software resources in a single administrative domain. In this setup, security, resource management, fault tolerance, and quality of service are less challenging than in a heterogeneous environment with resources in multiple administrative domains.

➔ Cloud computing is focused on enterprise computing; its adoption by industrial organizations, financial institutions, healthcare organizations, and so on has a potentially huge impact on the economy.

➔ A cloud provides the illusion of infinite computing resources; its elasticity frees application designers from the confinement of a single system.

➔ A cloud eliminates the need for up-front financial commitment, and it is based on a pay-as-you-go approach. This has the potential to attract new applications and new users for existing applications, fomenting a new era of industrywide technological advancements.

*Challenges*

➔ The most significant challenge is security [19]; gaining the trust of a large user base is critical for the future of cloud computing. It is unrealistic to expect that a public cloud will provide a suitable environment for all applications

➔ The SaaS model faces similar challenges as other online services required to protect private information, such as financial or healthcare services.

➔ The IaaS model is by far the most challenging to defend against attacks. Indeed, an IaaS user has considerably more degrees of freedom than the other two cloud delivery models.

➔ Virtualization is a critical design option for this model, but it exposes the system to new sources of attack. The trusted computing base (TCB) of a virtual environment includes not only the hardware and the hypervisor but also the management operating system.

➔ The next major challenge is related to resource management on a cloud.

➔ The last major challenge we want to address is related to interoperability and standardization. Vendor lock-in, the fact that a user is tied to a particular cloud service provider, is a major concern for cloud users. Standardization would support interoperability and thus alleviate some of the fears that a service critical for a large organization may not be available for an extended period of time. But imposing standards at a time when a technology is still evolving is not only challenging, it can be counterproductive because it may stifle innovation.

4) **List the procedure to construct a virtual machine using Eucalyptus. List and explain the components of the system.**

*Procedure*

➔ The euca2ools front end is used to request a VM.
➔ The VM disk image is transferred to a compute node.
➔ This disk image is modified for use by the VMM on the compute node.
➔ The compute node sets up network bridging to provide a virtual network interface controller (NIC)8 with a virtual Media Access Control (MAC) address.
➔ In the head node the DHCP is set up with the MAC/IP pair.
➔ VMM activates the VM.
➔ The user can now ssh directly into the VM.

*Components*

➔ **Virtual machine**. Runs under several VMMs, including Xen, KVM, and Vmware.
➔ **Node controller**. Runs on every server or node designated to host a VM and controls the activities of the node. Reports to a cluster controller.
➔ **Cluster controller**. Controls a number of servers. Interacts with the node controller on each server to schedule requests on that node. Cluster controllers are managed by the cloud controller.
➔ **Cloud controller**. Provides the cloud access to end users, developers, and administrators. It is accessible through command-line tools compatible withEC2 and through aWeb-based Dashboard. Manages cloud resources, makes high-level scheduling decisions, and interacts with cluster controllers.
➔ **Storage controller.** Provides persistent virtual hard drives to applications. It is the correspondent of EBS. Users can create snapshots from EBS volumes. Snapshots are stored in Walrus and made available across availability zones.
➔ **Storage service (Walrus)**. Provides persistent storage and, similarly to S3, allows users to store objects in buckets.

5) **Peer-to-peer systems and clouds share a few goals but not the means to accomplish them. Compare the two classes of systems in terms of architecture, resource management, scope, and security.**

*Refer 12th*

6) **Write the differences that arise among the three storage systems in AWS and explain the monitoring system of AWS.**

*Simple Storage System (S3):*

➔ a storage service designed to store large objects. It supports a minimal set of functions: write, read, and delete.
➔ S3 allows an application to handle an unlimited number of objects ranging in size from one byte to five terabytes.

➔ An object is stored in a bucket and retrieved via a unique developer-assigned key.

➔ A bucket can be stored in a region selected by the user. S3 maintains the name, modification time, an access control list, and up to four kilobytes of user-defined metadata for each object.

➔ The object names Services offered by AWS are accessible from the AWS Management Console.

➔ S3 supports PUT, GET, and DELETE primitives to manipulate objects but does not support primitives to copy, rename, or move an object from one bucket to another. Appending to an object requires a read followed by a write of the entire object.

➔ S3 computes the MD52 of every object written and returns it in a field called Etag.

➔ The Amazon S3 SLA guarantees reliability.

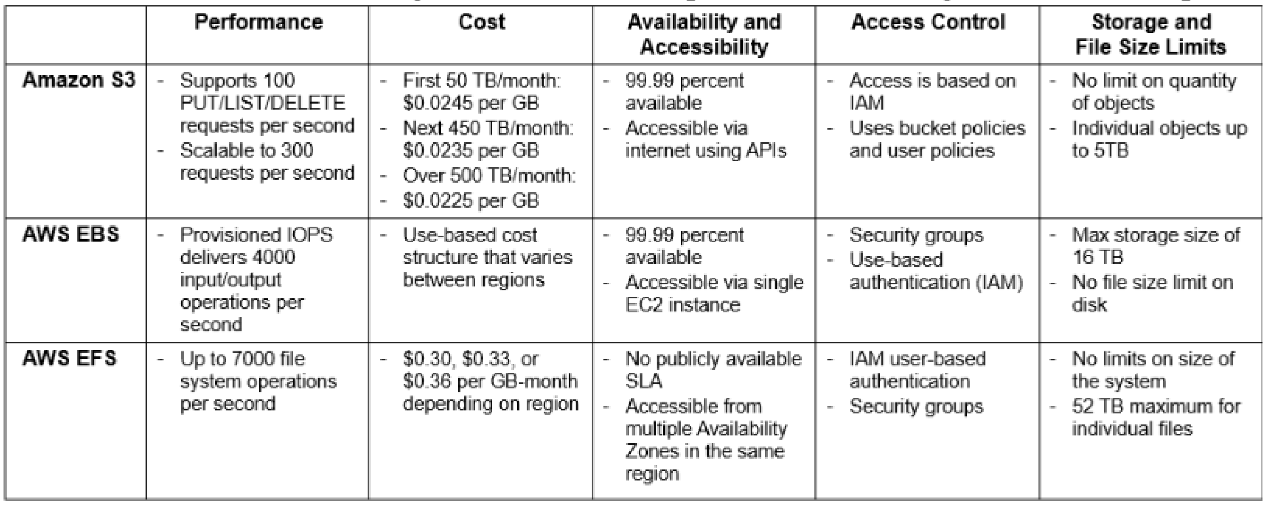
➔ S3 uses standards-based REST and SOAP interfaces.

*Elastic Block Storage(EBS)*

- ➔ provides persistent block-level storage volumes for use with Amazon 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. The volumes are grouped together in availability zones and are automatically replicated in each zone.
- ➔ An EC2 instance may mount multiple volumes, but a volume cannot be shared among multiple instances.
- ➔ The EBS supports the creation of snapshots of the volumes attached to an instance and then uses them to restart an instance.
- ➔ The storage strategy provided by EBS is suitable for database applications, file systems, and applications using raw data devices.

*Simple DB*

- ➔ a nonrelational 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-performanceWeb applications; at the same time,
- ➔ it automatically manages infrastructure provisioning, hardware and software maintenance, replication and indexing of data items, and performance tuning.



| | Performance | Cost | Availability and Accessibility | Access Control | Storage and File Size Limits |
|---|---|---|---|---|---|
| **Amazon S3** | - Supports 100 PUT/LIST/DELETE requests per second<br>- Scalable to 300 requests per second | - First 50 TB/month: $0.0245 per GB<br>- Next 450 TB/month: $0.0235 per GB<br>- Over 500 TB/month:<br>- $0.0225 per GB | - 99.99 percent available<br>- Accessible via internet using APIs | - Access is based on IAM<br>- Uses bucket policies and user policies | - No limit on quantity of objects<br>- Individual objects up to 5TB |
| **AWS EBS** | - Provisioned IOPS delivers 4000 input/output operations per second | - Use-based cost structure that varies between regions | - 99.99 percent available<br>- Accessible via single EC2 instance | - Security groups<br>- Use-based authentication (IAM) | - Max storage size of 16 TB<br>- No file size limit on disk |
| **AWS EFS** | - Up to 7000 file system operations per second | - $0.30, $0.33, or $0.36 per GB-month depending on region | - No publicly available SLA<br>- Accessible from multiple Availability Zones in the same region | - IAM user-based authentication<br>- Security groups | - No limits on size of the system<br>- 52 TB maximum for individual files |

**CloudWatch is a monitoring infrastructure** used by application developers, users, and system administrators to collect and track metrics important for optimizing the performance of applications and for increasing the efficiency of resource utilization. Without installing any software, a user can monitor approximately a dozen preselected metrics and then view graphs and statistics for these metrics.

When launching an Amazon Machine Image (AMI), a user can start the CloudWatch and specify the type of monitoring. Basic Monitoring is free of charge and collects data at five-minute intervals for up to 10 metrics; Detailed Monitoring is subject to a charge and collects data at one-minute intervals. This service can also be used to monitor the latency of access to EBS volumes, the available storage space for RDS DB instances, the number of messages in SQS, and other parameters of interest for applications.

**7) List any three Paas and Saas offered by Google and explain the same.**

*Sass :- Gmail, Google Drive, Google Calendar, Picasa, and Google Groups*

➔ The **Gmail** service hosts emails on Google servers and, provides a Web interface to access them and tools for migrating from Lotus Notes and Microsoft Exchange.

➔ **Google Docs** is Web-based software for building text documents, spreadsheets, and presentations. It supports features such as tables, bullet points, basic fonts, and text size; it allows multiple users to edit and update the same document and view the history of document changes; and it provides a spell checker. The service allows users to import and export files in several formats, including Microsoft Office, PDF, text, and OpenOffice extensions.

➔ **Google Calenda**r is a browser-based scheduler; it supports multiple calendars for a user, the ability to share a calendar with other users, the display of daily/weekly/monthly views, and the ability to search events and synchronize with the Outlook Calendar.

➔ **Picasa** is a tool to upload, share, and edit images; it provides 1 GB of disk space per user free of charge. Users can add tags to images and attach locations to photos using Google Maps.

➔ Google Groups allows users to host discussion forums to create messages online or via email.

*Pass :- AppEngine, Google Co-Op, Google Base, Google Drive*

➔ AppEngine is a developer platform hosted on the cloud. Initially it supported only Python, but support for Java was added later and detailed documentation for Java is available. The database for code development can be accessed with Google Query Language (GQL) with a SQL-like syntax.

➔ Google Co-op allows users to create customized search engines based on a set of facets or categories. For example, the facets for a search engine for the database research community available at http://data.cs.washington.edu/coop/dbresearch/index.html are professor, project, publication, jobs.

➔ Google Base is a service allowing users to load structured data from different sources to a central repository that is a very large, self-describing, semi-structured, heterogeneous database. It is selfdescribing because each item follows a simple schema: (item type, attribute names). Few users are aware of this service. Google Base is accessed in response to keyword queries posed on Google.com, provided that there is relevant data in the database. To fully integrate Google Base, the results should be ranked across properties. In addition, the service needs to propose appropriate refinements with candidate values in select menus; this is done by computing histograms on attributes and their values during query time.

➔ Google Drive is an online service for data storage that has been available since April 2012. It gives users 5 GB of free storage and charges $4 per month for 20 GB. It is available for PCs, MacBooks, iPhones, iPads, and Android devices and allows organizations to purchase up to 16 TB of storage. Specialized structure-aware search engines for several interest areas, including travel, weather, and local services, have already been implemented. However, the data available on the Web covers a wealth of human knowledge; it is not feasible to define all the possible domains and it is nearly impossible to decide where one domain ends and another begins.

8) **What is SLA? List the Objectives of SLA, Mention the common areas recorded by SLA and its coverage.**

A *service-level agreement (SLA)* is a negotiated contract between two parties, the customer and the service provider. The agreement can be legally binding or informal and specifies the services that the customer receive rather than how the service provider delivers the services.

*Objectives*

➔ Identify and define customers' needs and constraints, including the level of resources, security, timing, and quality of service.
➔ Provide a framework for understanding. A critical aspect of this framework is a clear definition of classes of service and costs.
➔ Simplify complex issues; for example, clarify the boundaries between the responsibilities of the clients and those of the provider of service in case of failures.
➔ Reduce areas of conflict.
➔ Encourage dialogue in the event of disputes.
➔ Eliminate unrealistic expectations.

*An SLA records a common understanding in several areas:*

➔ services,
➔ priorities,
➔ responsibilities,
➔ guarantees,
➔ warranties.

*An agreement usually covers:*

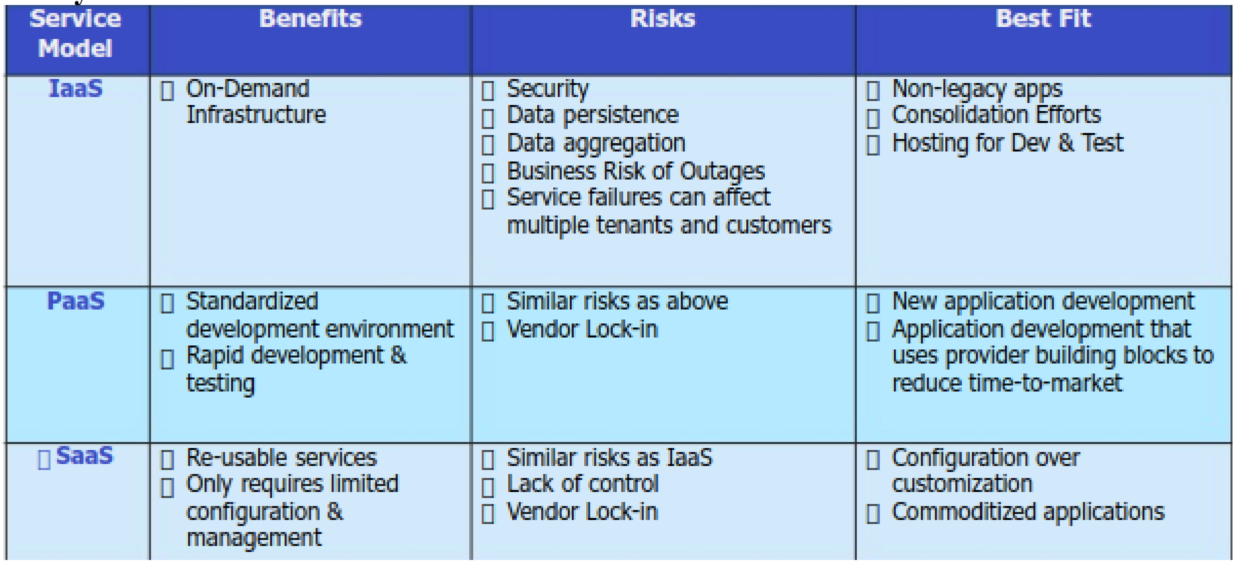
➔ services to be delivered,
➔ performance,
➔ tracking and reporting,
➔ problem management,
➔ legal compliance and resolution of disputes,
➔ customer duties and responsibilities,
➔ security,
➔ handling of confidential information, and termination.

Each area of service in cloud computing should define a "target level of service" or a "minimum level of service" and specify the levels of availability, serviceability, performance, operation, or other attributes of the service, such as billing.

9) **Define cloud computing. List the delivery and deployment models with examples. Compare them.**

Cloud computing is a model for enabling ubiquitous(found everywhere), 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.

**Delivery Models**



| Service Model | Benefits | Risks | Best Fit |
|---|---|---|---|
| IaaS | ▯ On-Demand Infrastructure | ▯ Security<br>▯ Data persistence<br>▯ Data aggregation<br>▯ Business Risk of Outages<br>▯ Service failures can affect multiple tenants and customers | ▯ Non-legacy apps<br>▯ Consolidation Efforts<br>▯ Hosting for Dev & Test |
| PaaS | ▯ Standardized development environment<br>▯ Rapid development & testing | ▯ Similar risks as above<br>▯ Vendor Lock-in | ▯ New application development<br>▯ Application development that uses provider building blocks to reduce time-to-market |
| ▯ SaaS | ▯ Re-usable services<br>▯ Only requires limited configuration & management | ▯ Similar risks as IaaS<br>▯ Lack of control<br>▯ Vendor Lock-in | ▯ Configuration over customization<br>▯ Commoditized applications |

*Deployment Models*

**Private Cloud**

It is a cloud-based infrastructure used by stand-alone organizations. It offers greater control over security. The data is backed up by a firewall and internally, and can be hosted internally or externally. Private clouds are perfect for organizations that have high-security requirements, high management demands, and availability requirements.

**Public Cloud**

This type of cloud services is provided on a network for public use. Customers have no control over the location of the infrastructure. It is based on a shared cost model for all the users, or in the form of a licensing policy such as pay per user. Public deployment models in the cloud are perfect for organizations with growing and fluctuating demands. It is also popular among businesses of all sizes for their web applications, webmail, and storage of non-sensitive data.

**Community Cloud**

It is a mutually shared model between organizations that belong to a particular community such as banks, government organizations, or commercial enterprises. Community members generally share similar issues of privacy, performance, and security. This type of deployment model of cloud computing is managed and hosted internally or by a third-party vendor.

**Hybrid Cloud**

This model incorporates the best of both private and public clouds, but each can remain as separate entities. Further, as part of this deployment of cloud computing model, the internal, or external providers can provide resources. A hybrid cloud is ideal for scalability, flexibility, and security. A perfect example of this scenario would be that of an organization who uses the private cloud to secure their data and interacts with its customers using the public cloud.

**10) List the Obstacles of cloud computing that provides opportunity for research fart.**

- ➔ ***Availability of service***. What happens when the service provider cannot deliver? Can a large company such as General Motors move its IT 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 (SLAs). A temporary fix with negative economical implications is overprovisioning, that is, having enough resources to satisfy the largest projected demand.
- ➔ ***Vendor lock-in***. Once a customer is hooked to one provider, it is hard to move to another. The standardization efforts at National Institute of Standards and Technology (NIST) attempt to address this problem.
- ➔ ***Data confidentiality and auditability***. This is indeed a serious problem;
- ➔ ***Data transfer bottlenecks***. Many applications are data-intensive. A very important strategy is to store the data as close as possible to the site where it is needed. Transferring 1 TB of data on a 1 Mbps network takes 8 million 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; for example, a 1 Gbps network would reduce this time to 8,000 s, or slightly more than 2 h.
- ➔ **Performance unpredictability.** This is one of the consequences of resource sharing. Strategies for performance isolation.
- ➔ **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 selfmanagement seems to be a promising avenue.

**11) Outline the features and functions of the following services offered by AWS: EC2, S3, SQS, Cloud Watch.**

**EC2**

Elastic Compute Cloud (EC2)1 is a Web service with a simple interface for launching instances of an application under several operating systems, such as several Linux distributions, Microsoft Windows Server 2003 and 2008, OpenSolaris, FreeBSD, and NetBSD.

An instance is created either from a predefined Amazon Machine Image (AMI) digitally signed and stored in S3 or from a user-defined image. The image includes the operating system, the run-time environment, the libraries, and the application desired by the user. AMI images create an exact copy of the original image but without configuration-dependent information such as the hostname or the MAC address.

EC2 allows the import of virtual machine images from the user environment to an instance through a facility called VM import. It also automatically distributes the incoming application traffic among multiple instances using the elastic load-balancing facility. EC2 associates an elastic IP address with an account;

- ➔ Launch an instance from an existing AMI and terminate an instance.
- ➔ start and stop an instance.
- ➔ create a new image.
- ➔ add tags to identify an image.
- ➔ reboot an instance.

**S3**

Simple Storage System (S3) is a storage service designed to store large objects. It supports a minimal set of functions: write, read, and delete. S3 allows an application to handle an unlimited number of objects ranging in size from one byte to five terabytes. An object is stored in a bucket and retrieved via a unique developer-assigned key. A bucket can be stored in a region selected by the user. S3 maintains the name, modification time, an access control list, and up to four kilobytes

of user-defined metadata for each object. The object names Services offered by AWS are accessible from the AWS Management Console.
S3 supports PUT, GET, and DELETE primitives to manipulate objects but does not support primitives to copy, rename, or move an object from one bucket to another. Appending to an object requires a read followed by a write of the entire object.
S3 computes the MD5 of every object written and returns it in a field called ETag. A user is expected to compute the MD5 of an object stored or written and compare this with the ETag; if the two values do not match, then the object was corrupted during transmission or storage. The Amazon S3 SLA guarantees reliability.
S3 uses standards-based REST and SOAP interfaces., the default download protocol is HTTP, but BitTorrent3 protocol interface is also provided to lower costs for high-scale distribution.

**SQS**
Simple Queue Service (SQS) is a hosted message queue. SQS is a system for supporting automated workflows; it allows multiple Amazon 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.
Applications using SQS can run independently and asynchronously and do not need to be developed with the same technologies.
A received message is "locked" during processing; if processing fails, the lock expires and the message is available again. The time-out for locking can be changed dynamically via the ChangeMessageVisibility operation.
Developers can access SQS through standards-based SOAP and Query interfaces. Queues can be shared with other AWS accounts and anonymously; queue sharing can also be restricted by IP address and time-of-day.

**Cloud Watch**
CloudWatch is a monitoring infrastructure used by application developers, users, and system administrators to collect and track metrics important for optimizing the performance of applications and for increasing the efficiency of resource utilization. Without installing any software, a user can monitor approximately a dozen preselected metrics and then view graphs and statistics for these metrics.
When launching an Amazon Machine Image (AMI), a user can start the CloudWatch and specify the type of monitoring. Basic Monitoring is free of charge and collects data at five-minute intervals for up to 10 metrics; Detailed Monitoring is subject to a charge and collects data at one-minute intervals. This service can also be used to monitor the latency of access to EBS volumes, the available storage space for RDS DB instances, the number of messages in SQS, and other parameters of interest for applications.

12) **Peer-to-peer systems and clouds share a few goals but not the means to accomplish them. Compare the two classes of systems in terms of architecture, resource management, scope, and security.**

Peer-to-peer systems allow individuals to share data and computing resources, primarily storage space without sharing the cost of these resources; sharing other types of resources, such as computing cycles, is considerably more difficult. P2P systems have several desirable properties

- ➔ Require a minimally dedicated infrastructure, since resources are contributed by the participating systems.
- ➔ Are highly decentralized.
- ➔ Are scalable; the individual nodes are not required to be aware of the global state.
- ➔ Are resilient to faults and attacks, since few of their elements are critical for the delivery of service and the abundance of resources can support a high degree of replication.
- ➔ Individual nodes do not require excessive network bandwidth the way servers used in case of the client-server model do.
- ➔ The systems are shielded from censorship due to the dynamic and often unstructured system architecture.

Decentralization raises the question of whether P2P systems can be managed effectively and provide the security required by various applications. The fact that they are shielded from censorship makes them a fertile ground for illegal activities, including distribution of copyrighted content.
A peer-to-peer system is based on an ad-hoc infrastructure where individuals contribute resources to the system, but join and leave the system as they wish; thus, a high degree of redundancy is necessary, the information must be replicated on multiple sites. A peer-to-peer system maintains a directory of data and sites; this directory can be at a central site or distributed

The need to make computing and storage more affordable and to liberate users from the concerns regarding system and software maintenance reinforced the idea of concentrating computing resources in large cloud computing centers
The defining attributes of the cloud computing paradigm are:

- ➔ Cloud computing uses Internet technologies to offer elastic services. The term elastic computing refers to the ability to dynamically acquire computing resources and support a variable workload. A cloud service provider maintains a massive infrastructure to support elastic services.
- ➔ The resources used for these services can be metered and the users can be charged only for the resources they use.
- ➔ Maintenance and security are ensured by service providers.
- ➔ Economy of scale allows service providers to operate more efficiently due to specialization and centralization.
- ➔ Cloud computing is cost-effective due to resource multiplexing; lower costs for the service provider are passed on to the cloud users.
- ➔ The application data is stored closer to the site where it is used in a device- and locationindependent manner; potentially, this data storage strategy increases reliability and security and, at the same time, it lowers communication costs.

A cloud computing center is managed by a cloud service provider which owns the servers and the networking infrastructure. Services are provided based on SLAs (Service Level Agreement). A cloud provides reliable storage and the data is automatically replicated. The level of user control of the cloud resources is different for the three cloud delivery models. Security is a major concern in both cases, especially in the case of cloud computing.

13) **An organization debating whether to install a private cloud or to use a public cloud, e.g., the AWS, for its computational and storage needs, asks your advice. What information will you require to base your recommendation on, and how will you use each one of the following items: (a) the description of the algorithms and the type of the applications the organization will run; (b) the system software used by these applications; (c) the resources needed by each application; (d) the size of the user population; (e) the relative experience of the user population; (d) the costs involved.**

Public clouds have distinct cost advantages over private clouds; there is no initial investment in the infrastructure, no recurring costs for administration, maintenance, energy consumption, and for the user support personnel. The main concern is security and privacy. An organization with very strict security and privacy concerns is very unlikely to use a public cloud.

The type of applications play a critical role, scientific and engineering computations which require a low latency interconnection network and enjoy only fine-grain parallelism are unlikely to fare well on either a public or a private cloud. A large user population is more likely to use identical or similar software and to cooperate by sharing the raw data and the results; thus, a private cloud seems more advantageous in this case. Some of the services offered by private clouds target experiences users, e.g., AWS services such as ElasticBeanstalk, while others are accessible to lay persons.

## UNIT 2

**14) Compare the traditional execution of program with a workflow using a neat diagram. Mention the different types of tasks and explain.**

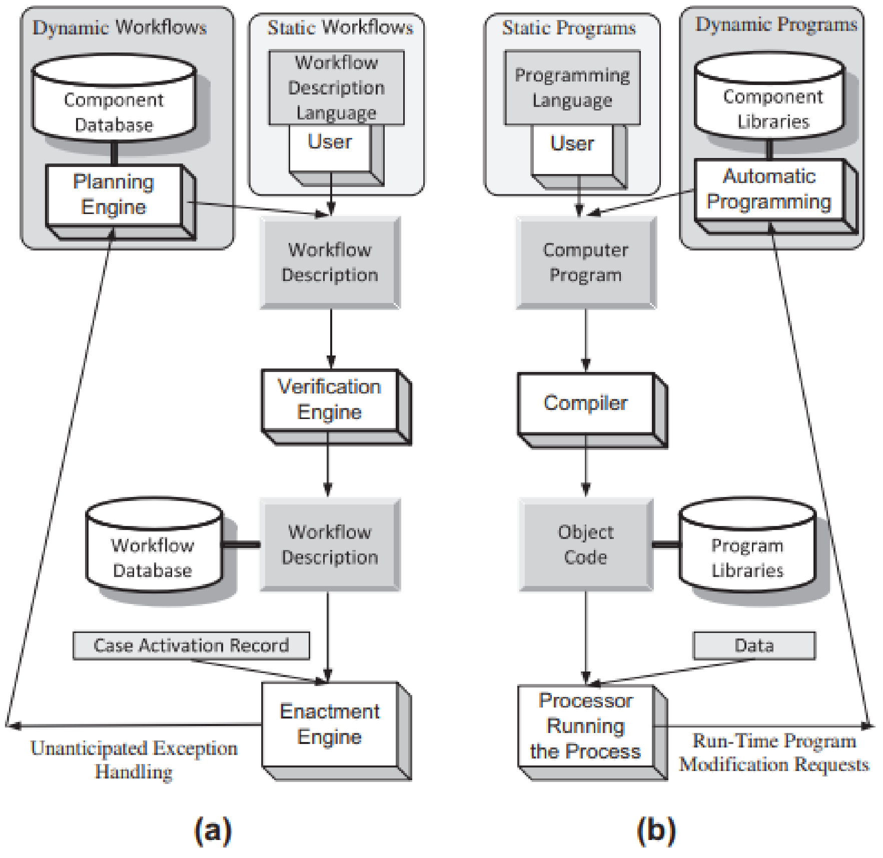


**FIGURE 4.1**

A parallel between workflows and programs. (a) The life cycle of a workflow. (b) The life cycle of a computer program. The workflow definition is analogous to writing a program. Planning is analogous to automatic

- ➔ The phases in the life cycle of a workflow are creation, definition, verification, and enactment. There is a striking similarity between the life cycle of a workflow and that of a traditional computer program, namely, creation, compilation, and execution
- ➔ The workflow specification by means of a workflow description language is analogous to writing a program. Planning is equivalent to automatic program generation. Workflow verification corresponds to syntactic verification of a program, and workflow enactment mirrors the execution of a compiled program
- ➔ A case is an instance of a process description. The start and stop symbols in the workflow description enable the creation and the termination of a case, respectively. An enactment model describes the steps Taken to process a case. When a computer executes all tasks required by a workflow the enactment can be performed by a program called an enactment engine.
- ➔ The state of a case at time t is defined in terms of tasks already completed at that time. Events cause transitions between states
- ➔ the new workflow description is generated automatically, knowing a set of tasks and the preconditions and post-conditions for each one of them. In artificial intelligence (AI) this activity is known as planning.
- ➔ Workflow pattern refers to the temporal relationship among the tasks of a process. The workflow description languages and the mechanisms to control the enactment of a case must have provisions to support these temporal relationships.

*different types of tasks*

A **composite task** is a structure describing a subset of tasks and the order of their execution. A primitive task is one that cannot be decomposed into simpler tasks. A composite task inherits some properties from workflows; it consists of tasks and has one start symbol and possibly several end symbols. At the same time, a composite task inherits some properties from tasks; it has a name, preconditions, and post-conditions.

A **routing task** is a special-purpose task connecting two tasks in a workflow description. The task that has just completed execution is called the **predecessor task**; the one to be initiated next is called the **successor task**. A routing task could trigger a sequential, concurrent, or iterative execution. A **fork routing task** triggers execution of several successor tasks. **A join routing task** waits for completion of its predecessor tasks.

15) **Define a set of keywords that are ordered based on their relevance to the topic of cloud security. Then search the Web using these keywords to locate 10–20 papers and store the papers in an S3 bucket. Create a MapReduce application modeled after the one discussed in Section 4.7 to rank the papers based on the incidence of the relevant keywords. Compare your ranking with the rankings of the search engine you used to identify the papers.**

**16) With a neat sketch of Grep the web workflow, explain the different phases of Grep the Web.**

- ➔ The startup phase. Creates several queues – launch, monitor, billing, and shutdown queues. Starts the corresponding controller threads. Each thread periodically polls its input queue and, when a message is available, retrieves the message, parses it, and takes the required actions.
- ➔ The processing phase. This phase is triggered by a StartGrep user request; then a launch message is enqueued in the launch queue. The launch controller thread picks up the message and executes the launch task; then, it updates the status and time stamps in the Amazon. Simple DB domain. Finally, it enqueues a message in the monitor queue and deletes the message from the launch queue.
    - ➔ The launch task starts Amazon EC2 instances.
    - ➔ Hadoop runs map tasks on Amazon EC2 slave nodes in parallel.
    - ➔ Final results are stored on Amazon S3 in the output bucket
- ➔ The monitoring phase. The monitor controller thread retrieves the message left at the beginning of the processing phase, validates the status/error in Amazon Simple DB, and executes the monitor task. It updates the status in the Amazon Simple DB domain and enqueues messages in the shutdown and billing queues.
- ➔ The shutdown phase. The shutdown controller thread retrieves the message from the shutdown queue and executes the shutdown task, which updates the status and time stamps in the Amazon Simple DB domain. Finally, it deletes the message from the shutdown queue after processing.
    - ➔ The shutdown phase. The shutdown controller thread retrieves the message from the shutdown queue and executes the shutdown task, which updates the status and time stamps in the Amazon Simple DB domain. Finally, it deletes the message from the shutdown queue after processing.
    - ➔ The billing task gets the EC2 topology information, Simple DB usage, and S3 file and query input, calculates the charges, and passes the information to the billing service.
- ➔ The cleanup phase. Archives the Simple DB data with user info
- ➔ User interactions with the system. Get the status and output results. The GetStatus is applied to the service endpoint to get the status of the overall system (all controllers and Hadoop) and download the filtered results from Amazon S3 after completion.

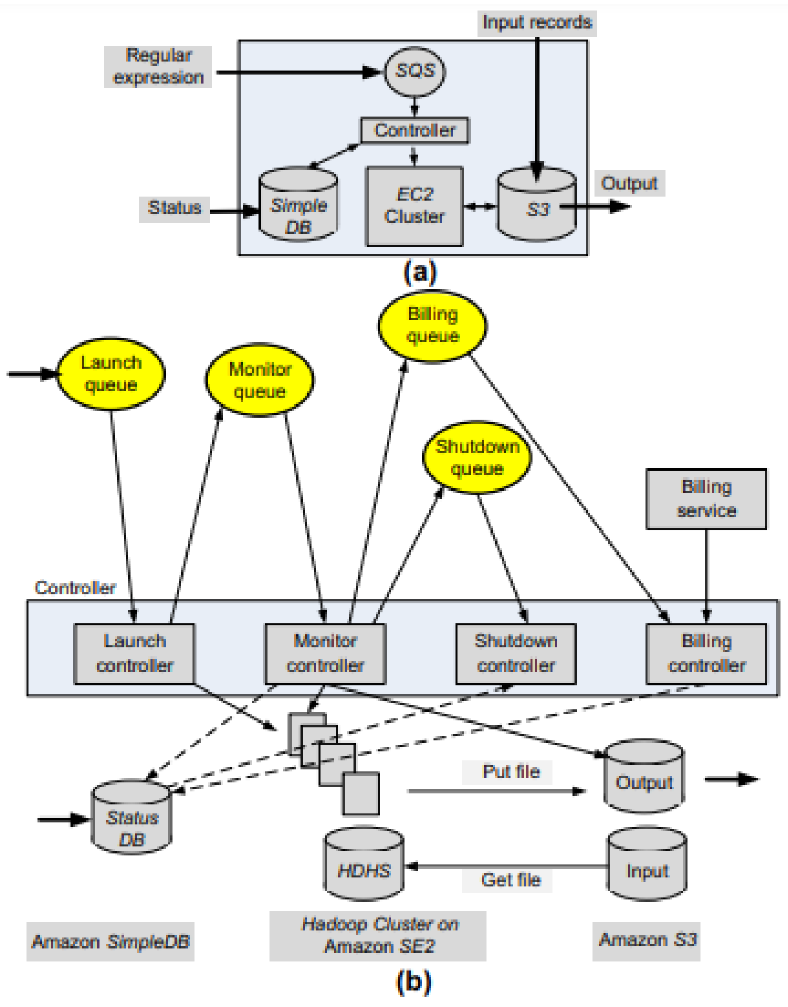
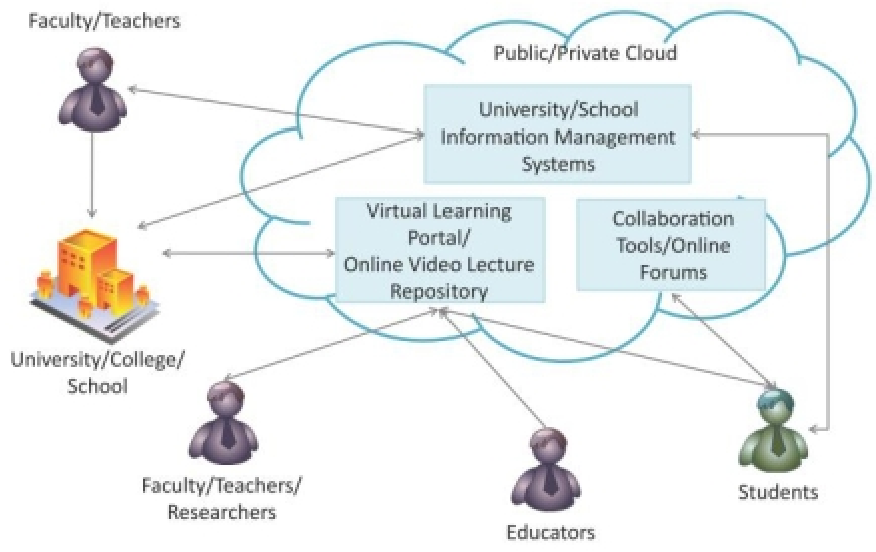


**FIGURE 4.7**

The organization of the *GrepTheWeb* application. The application uses the *Hadoop MapReduce* software and four Amazon services: *EC2*, *Simple DB*, *S3*, and *SQS*. (a) The simplified workflow showing the two inputs, the regular expression and the input records generated by the Web crawler. A third type of input is the user commands to report the current status and to terminate the processing. (b) The detailed workflow; the system is based on message passing between several queues; four controller threads periodically poll their associated input queues, retrieve messages, and carry out the required actions.

**17) Explain with a neat diagram, the application of cloud computing environments to the Education and transportation systems.**


Faculty/Teachers
Public/Private Cloud
University/School Information Management Systems
Virtual Learning Portal/ Online Video Lecture Repository
Collaboration Tools/Online Forums
University/College/ School
Faculty/Teachers/ Researchers
Educators
Students

**Online Learning Platforms & Collaboration Tools**

- Cloud computing is bringing a transformative impact in the field of education by improving the reach of quality education to students through the use of online learning platforms and collaboration tools.

**• MOOCs**

- MOOCs are aimed for large audiences and use cloud technologies for providing audio/video content, readings, assignment and exams.
- Cloud-based auto-grading applications are used for grading exams and assignments. Cloud-based applications for peer grading of exams and assignments are also used in some MOOCs.

**• Online Programs**

- Many universities across the world are using cloud platforms for providing online degree programs.
- Lectures are delivered through live/recorded video using cloud based content delivery networks to students across the world.

**Online Proctoring**

- Online proctoring for distance learning programs is also becoming popular through the use of cloud-based live video streaming technologies where online proctors observe test takers remotely through video.

**Virtual Labs**

- Access to virtual labs is provided to distance learning students through the cloud. Virtual labs provide remote access to the same software and applications that are used by students on campus.

**Course Management Platforms**

- Cloud-based course management platforms are used to for sharing reading materials, providing assignments and releasing grades, for instance.
- Cloud-based collaboration applications such as online forums, can help student discuss common problems and seek guidance from experts.

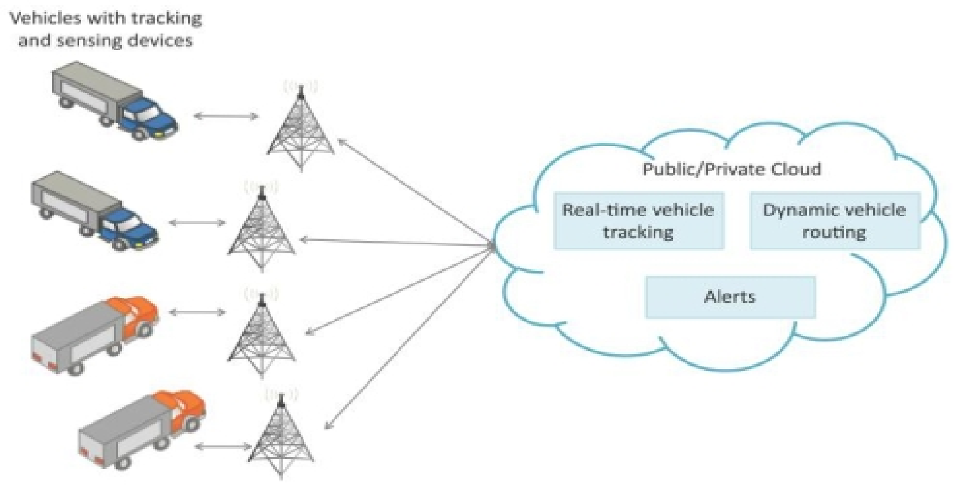
**Information Management**

- Universities, colleges and schools can use cloud-based information management systems to improve administrative efficiency, offer online and distance education programs, online exams, track progress of students, collect feedback from students, for instance.

**Reduce Cost of Education**

- Cloud computing thus has the potential of helping in bringing down the cost of education by increasing the student-teacher ratio through the use of online learning platforms and new evaluation approaches without sacrificing quality.

**Transportation**


Vehicles with tracking and sensing devices
Public/Private Cloud
Real-time vehicle tracking
Dynamic vehicle routing
Alerts

- Modern transportation systems are driven by data collected from multiple sources which is processed to provide new services to the stakeholders.
- By collecting large amount of data from various sources and processing the data into useful information, data-driven transportation systems can provide new services such as:
    - Advanced route guidance
    - Dynamic vehicle routing
    - Anticipating customer demands for pickup and delivery problem

**Fleet Tracking**

• Vehicle fleet tracking systems use GPS technology to track the locations of the vehicles in real-time.

• The vehicle locations and routes data can be aggregated and analyzed in the cloud for detecting bottlenecks in the supply chain such as traffic congestions on routes, assignments and generation of alternative routes, and supply chain optimization.

**• Route Generation & Scheduling**

• Route generation and scheduling systems can generate end-to-end routes using combination of route patterns and transportation modes and feasible schedules based on the availability of vehicles.

• As the transportation network grows in size and complexity, the number of possible route combinations increases exponentially.

• Cloud-based route generation and scheduling systems can provide fast response to the route generation queries and can be scaled up to serve a large transportation network.

• Condition Monitoring Condition monitoring solutions for transportation systems allow monitoring the conditions inside containers.

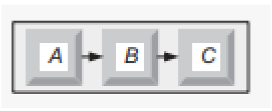
**• Planning, Operations & Services**

• Different transportation solutions (such as fleet tracking, condition monitoring, route generation, scheduling, cargo operations, fleet maintenance, customer service, order booking, billing & collection, for instance.) can be moved to the cloud to provide a seamless integration between order management, tactical planning & execution and customer facing processes & systems.

18) **Identify the basic workflow pattern for the following scenarios. Represent it using a neat diagram.**

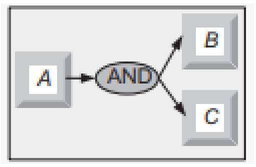
**i. An insurance claim is evaluated after the client's file is retrieved.**

The *sequence pattern* occurs when several tasks have to be scheduled one after the completion of the other.



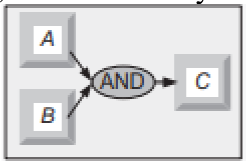
**ii. The execution of the activity payment enables the execution of the activities ship goods and inform customer.**

The *AND split* pattern requires several tasks to be executed concurrently. Both tasks B and C are activated when task A terminates.



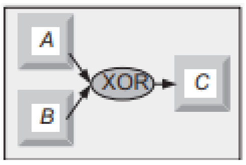
**iii. Insurance claims are evaluated after the policy has been checked and the actual damage has been assessed.**

The synchronization pattern requires several concurrent activities to terminate before an activity can start. In our example, task C can only start after both tasks A and B terminate.

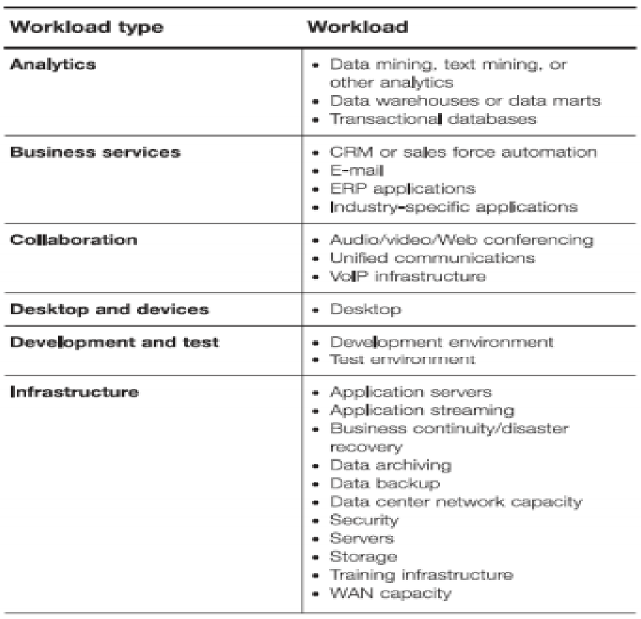


**iv. After the payment is received or the credit is granted the car is delivered to the customer.**

In the *XOR join(XOR Merge)*, several alternatives are merged into one. In our example, task C is enabled when either A or B terminates



**19) Sketch a strategy to reduce the power consumption in a lightly loaded cloud and discuss the steps for placing a computational server in standby mode and then bringing it back up to active mode.**



| Workload type | Workload |
|---|---|
| **Analytics** | • Data mining, text mining, or other analytics<br>• Data warehouses or data marts<br>• Transactional databases |
| **Business services** | • CRM or sales force automation<br>• E-mail<br>• ERP applications<br>• Industry-specific applications |
| **Collaboration** | • Audio/video/Web conferencing<br>• Unified communications<br>• VoIP infrastructure |
| **Desktop and devices** | • Desktop |
| **Development and test** | • Development environment<br>• Test environment |
| **Infrastructure** | • Application servers<br>• Application streaming<br>• Business continuity/disaster recovery<br>• Data archiving<br>• Data backup<br>• Data center network capacity<br>• Security<br>• Servers<br>• Storage<br>• Training infrastructure<br>• WAN capacity |

Figure 4: Workloads discussed in the IBM survey.

even at a very low load a server uses close to 50% of the energy consumption when delivering peak performance. The basic philosophy is to maintain all servers in an optimal region, a region where the relative performance is high while the relative power consumption is low. To achieve this goal, the applications running on a server working outside its optimal region should be migrated to other servers and the server should be switched to a sleep mode when its power consumption is negligible.

➔ In an ideal world, the energy consumed by an idle system should be near zero and should grow linearly with the system load. In real life, even machines whose power requirements scale linearly, use more than half the power when idle than they use at full load

➔ Energy-proportional systems could lead to large savings in energy costs for computing clouds. An energy-proportional system consumes no power when idle, very little power under a light load, and gradually more power as the load increases.

➔ an ideal energy-proportional system is always operating at 100% efficiency. Humans are a good approximation of an ideal energy proportional system; human energy consumption is about 70 W at rest and 120 W on average on a daily basis and can go as high as 1,000–2,000 W during a strenuous, short effort.

➔ Even when power requirements scale linearly with the load, the energy efficiency of a computing system is not a linear function of the load; even when idle, a system may use 50% of the power corresponding to the full load. Data collected over a long period of time shows that the typical operating region for the servers at a data center is from about 10% to 50% of the load.

➔ these networks operate in a very narrow dynamic range, and the power consumed when the network is idle is significant compared to the power consumed when the network is fully utilized.

➔ A strategy to reduce energy consumption is to concentrate the workload on a small number of disks and allow the others to operate in a low-power mode.

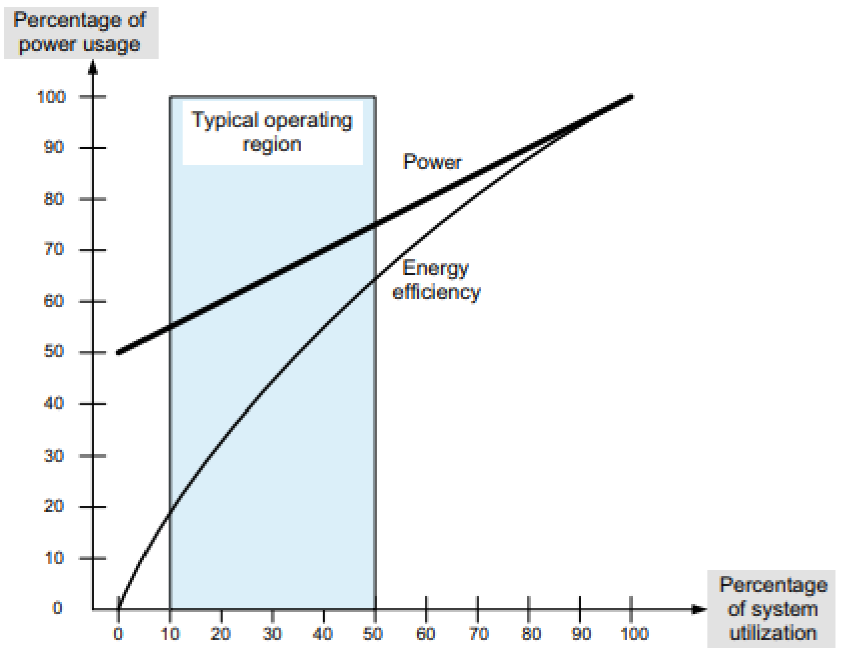
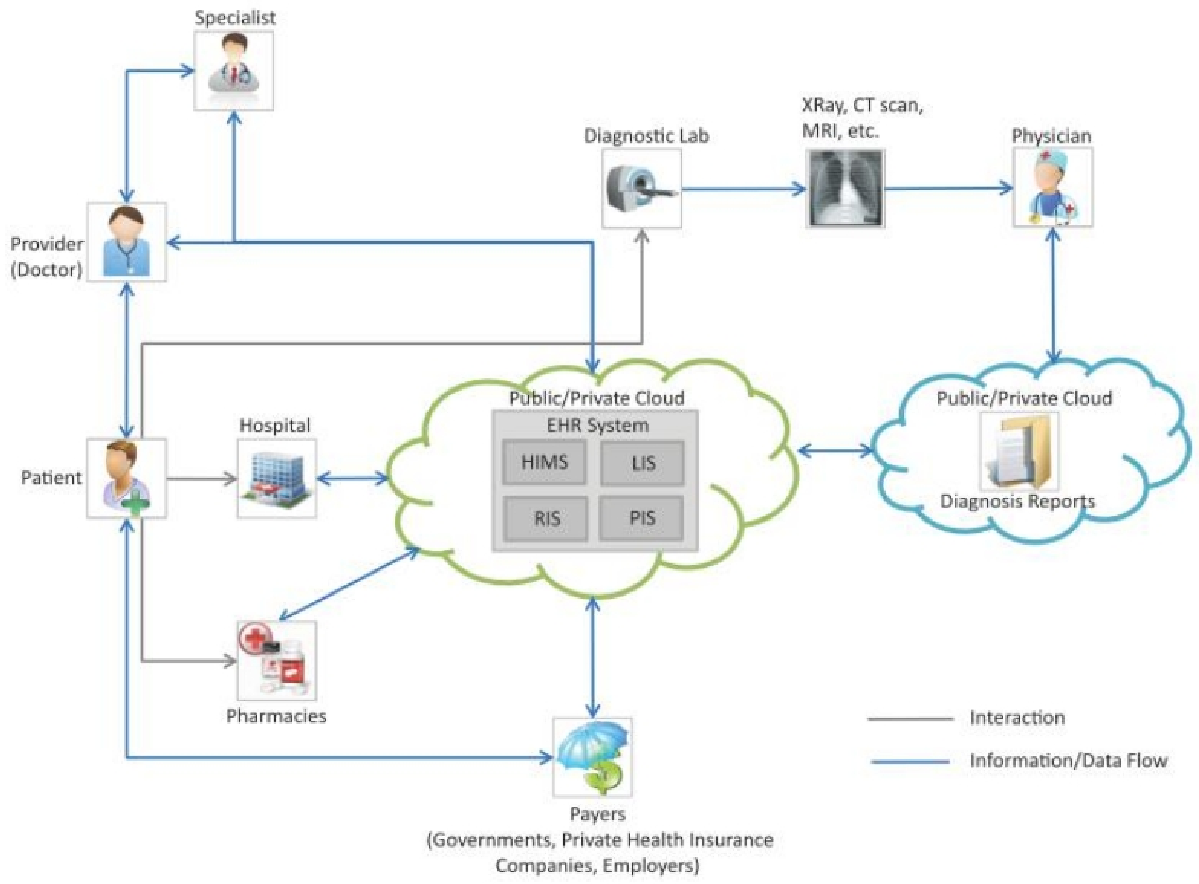


**FIGURE 3.6**

Even when power requirements scale linearly with the load, the energy efficiency of a computing system is not a linear function of the load; even when idle, a system may use 50% of the power corresponding to the full load. Data collected over a long period of time shows that the typical operating region for the servers at a data center is from about 10% to 50% of the load.

**20) Design architecture for establishing a private cloud at MSRIT campus, such that it accommodates auto scaling, different data centers, low latency, fault tolerance and recover disaster.**

**21) Show with a neat diagram, the application of cloud computing environments to the Health care ecosystems. Explain the same.**



**Heathcase Ecosystem**

• The healthcare ecosystem consists of numerous entities including healthcare providers (primary care physicians, specialists, hospitals, for instance), payers (government, private health insurance companies, employers), pharmaceutical, device and medical service companies, IT solutions and services firms, and patients.

**• Healthcare Data**

•The process of provisioning healthcare involves massive healthcare data that exists in different forms (structured or unstructured), is stored in disparate data sources (such as relational databases, file servers, for instance) and in many different formats.

The cloud can provide several benefits to all the stakeholders in the healthcare ecosystem through systems such as

• Health Information Management System (HIMS), Laboratory Information System (LIS), Radiology Information System (RIS), Pharmacy Information System (PIS), for instance.

**Providers & Hospitals**

• With public cloud based EHR systems hospitals don't need to spend a significant portion of their budgets on IT infrastructure.

• Public cloud service providers provide on-demand provisioning of hardware resources with pay-per-use pricing models.

• Thus hospitals using public cloud based EHR systems can save on upfront capital investments in hardware and data center infrastructure and pay only for the operating expenses of the cloud resources used.

• Hospitals can access patient data stored in the cloud and share the data with other hospitals.

**• Patients**

• Patients can provide access to their health history and information stored in the cloud (using SaaS applications) to hospitals so that the admissions, care and discharge processes can be streamlined.

• Physicians can upload diagnosis reports (such as pathology reports) to the cloud so that they can be accessed by doctors remotely for diagnosing the illness.
• Patients can manage their prescriptions and associated information such as dosage, amount and frequency, and provide this information to their healthcare provider.

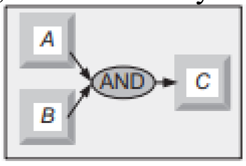
• **Payers**

• Health payers can increase the effectiveness of their care management programs byproviding value added services and giving access to health information to members.

**22) Use the basic workflow pattern and Prepare a neat workflow pattern for the following scenarios.**

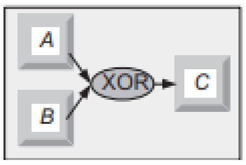
**i. Online banking transaction – for an online transaction to complete it requires cash and receipt to be collected in any order.**

The synchronization pattern requires several concurrent activities to terminate before an activity can start. In our example, task C can only start after both tasks A and B terminate.



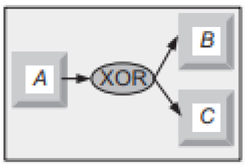
**ii. an application user selecting a task from the worklist, or a message being received by the process execution engine.**

In the *XOR join(XOR Merge)*, several alternatives are merged into one. In our example, task C is enabled when either A or B terminates



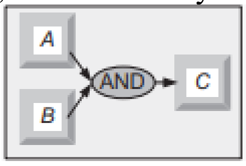
**iii. After an order has been received, the payment can be performed by either credit card or bank transfer.**

The XOR split requires a decision; after the completion of task A, either B or C can be activated

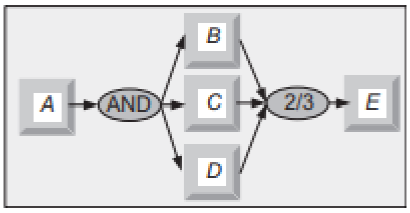


**iv. project team - Sarah must wait for several tasks to be completed by Kevin and George before she can execute her task.**

The synchronization pattern requires several concurrent activities to terminate before an activity can start. In our example, task C can only start after both tasks A and B terminate.

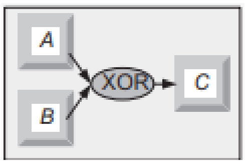


The N out of M join construct provides a barrier synchronization. Assuming that M > N tasks run concurrently, N of them have to reach the barrier before the next task is enabled. In our example, any two out of the three tasks A, B, and C have to finish before E is enabled.



**v. staff are serving at a counter - Raoul can serve a customer in his queue without waiting for Jamie to serve a customer in his queue.**

In the *XOR join(XOR Merge)*, several alternatives are merged into one. In our example, task C is enabled when either A or B terminates.



23) **Define a set of keywords that are ordered based on their relevance to the topic of cloud security. Then search the Web using these keywords to locate 10–20 papers and store the papers in an S3 bucket. Create a MapReduce application to rank the papers based on the incidence of the relevant keywords. Compare your ranking with the rankings of the search engine you used to identify the papers.**

*Refer 15th*

24) **Explain process description, workflow description and compare the lifecycle of a workflow with the life cycle of a computer program with a diagram.**

A *process description*, also called a workflow schema, is a structure describing the tasks or activities to be executed and the order of their execution. A process description contains one start symbol and one end symbol. A process description can be provided in a workflow definition language (WFDL), supporting constructs for choice, concurrent execution, the classical fork, join constructs, and iterative execution.

a *workflow description* resembles a flowchart, a concept we are familiar with from programming. The phases in the life cycle of a workflow are creation, definition, verification, and enactment. There is a striking similarity between the life cycle of a workflow and that of a traditional computer program, namely, creation, compilation, and execution. The workflow specification by means of a workflow description language is analogous to writing a program. Planning is equivalent to automatic program generation. Workflow verification corresponds to syntactic verification of a program, and workflow enactment mirrors the execution of a compiled program.

Refer 14th

25) **Research the power consumption of processors used in mobile devices and their energy efficiency. Rank the components of a mobile device in terms of power consumption. Establish a set of guidelines to minimize the power consumption of mobile applications.**

First, we overview the mobile processors produced by several manufacturers and the mobile systems using them. The ARM architecture was developed by ARM Holdings which does not manufacture its own electronic chips, but licenses its designs to other semiconductor manufacturers. The name ARM is an acronym for Advanced RISC Machine.

T*he ARM architectur*e is the primary hardware environment for most mobile device operating systems such as iOS, Android, Windows Phone, Blackberry OS/Blackberry 10, Firefox OS, Tizen and Ubuntu Touch. ARM-based processors and systems-on-a-chip include the Qualcomm

Snapdragon, nVidia Tegra, Marvell Xscale and Texas Instruments OMAP, as well as ARM's Cortex series and Apple System on Chips (used in its iPhones).

*Nvidia's Tegra 3 processor* has been the first successful quad-core mobile processor. The Tegra 3 has what the company terms a 12-core GeForce GPU and new video engines with support of 1080p video at 40Mbps. The processor is used in HTC One X, LG Optimus 4X HD, and ZTE Era, as well as a new high-end phone from Fujitsu and some new tablets from Asus. Nvidia's 4-Plus-1 architecture has four high-power A9s and one low-power one. Qualcomm's processors are in a large range of products, from some models of the Samsung Galaxy Note and Galaxy S II to the HTC One S, XL, and virtually all Windows Phones. Snapdragon S4 MSM8960 is a dual-core chip based on the company's new "Krait" architecture, which seems to be the first 28nm mobile processor shipping in volume.

Qualcomm is designing its own cores that are compatible with the ARM v7 instruction set, rather than using the standard ARM cores. The company also has its own graphics engine, known as Adreno. What really makes this stand out is an integrated LTE modem.

*Texas Instrument*s uses the OMAP 4 family of processors in Kindle Fires and Nook tablets, as well as the first Android 4.0 devices like the Google Nexus. The upcoming OMAP 5 family will introduce the new ARM Cortex-A15 core, faster and more powerful, with twin SGX-544 graphics cores, video compression, and two Cortex-M4s. In this "smart multicore architecture," the M4 processors handle a real-time embedded OS, while the A15s run Android.

*Intel Core i7* Extreme Edition has a maximum turbo frequency of 3.5 GHz, 8 MB as L3 cache, and is realized with the 32 nm technology; the memory speed is 1.6 Ghz.

There is very little information available about power consumption of mobile processors. Table 2 shows the evolution of the average power consumption for volume (Vol) servers - servers with a price less than $ 25 K, mid-range (Mid) servers - servers with a price between $25 K and $499 K, and high-end (High) servers - servers with a price tag larger than $500 K.

Table 2: Estimated average power use of volume, mid-range, and high-end servers (in Watts) along the years

| Type | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 |
|---|---|---|---|---|---|---|---|
| Vol | 186 | 193 | 200 | 207 | 213 | 219 | 225 |
| Mid | 424 | 457 | 491 | 524 | 574 | 625 | 675 |
| High | 5,534 | 5,832 | 6,130 | 6,428 | 6,973 | 7,651 | 8,163 |

The largest consumer of power of a system is the processor, followed by memory, and storage systems. The power consumption can vary from 45W to 200W per multi-core CPU; newer processors include power saving technologies. Most servers have several dual in-line memory modules (DIMMs); the power consumption of and a DIMM can vary from 5W up to 21W. Large servers often use 32 to 64 DIMMs; server memory cooling requires additional power. A server with 2-4 hard disk drives (HDDs) consumes 24 - 48W.

## UNIT 3

**26) Mention the objective of Layering and interfaces. Justify how a portable code is produced and distributed using system ISA.**

A common approach to managing system complexity is to identify a set of layers with well-defined interfaces among them. The interfaces separate different levels of abstraction. Layering minimizes the interactions among the subsystems and simplifies the description of the subsystems. Each subsystem is abstracted through its interfaces with the other subsystems.

- ➔ The instruction set architecture (ISA) defines a processor's set of instructions. For example, the Intel architecture is represented by the x86-32 and x86-64 instruction sets for systems supporting 32-bit addressing and 64-bit addressing, respectively.
- ➔ The hardware supports two execution modes, a *privileged, or kernel mode and a user mode.* The instruction set consists of two sets of instructions, ***p*****rivileged instructions that can only be executed in kernel mode** and **nonprivileged instructions that can be executed in user mode.** There are also sensitive instructions that can be executed in kernel and in user mode but that behave differently.
- ➔ The hardware consists of one or more multicore processors, a system interconnect (e.g., one or more buses), a memory translation unit, the main memory, and I/O devices, including one or more networking interfaces.
- ➔ Applications written mostly in high-level languages (HLL) often call library modules and are compiled into object code. Privileged operations, such as I/O requests, cannot be executed in user mode; instead, application and library modules issue system calls and the operating system determines whether the privileged operations required by the application do not violate system security or integrity and, if they don't, executes them on behalf of the user.
- ➔ The next interface is the application binary interface (ABI), which allows the ensemble consisting of the application and the library modules to access the hardware
- ➔ The ABI does not include privileged system instructions; instead it invokes system calls. Finally, the application program interface (API) defines the set of instructions the hardware was designed to execute and gives the application access to the ISA
- ➔ The ABI is the projection of the computer system seen by the process, and the API is the projection of the system from the perspective of the HLL program.

*the binaries created by a compiler for a specific ISA and a specific operating system are not portable. Such code cannot run on a computer with a different ISA or on computers with the same ISA but different operating systems. However, it is possible to compile an HLL program for a VM environment, as shown in Figure 5.2, where portable code is produced and distributed and then converted by binary translators to the ISA of the host system. A dynamic binary translation converts blocks of guest instructions from the portable code to the host instruction and leads to a significant performance improvement as such blocks are cached and reused.*

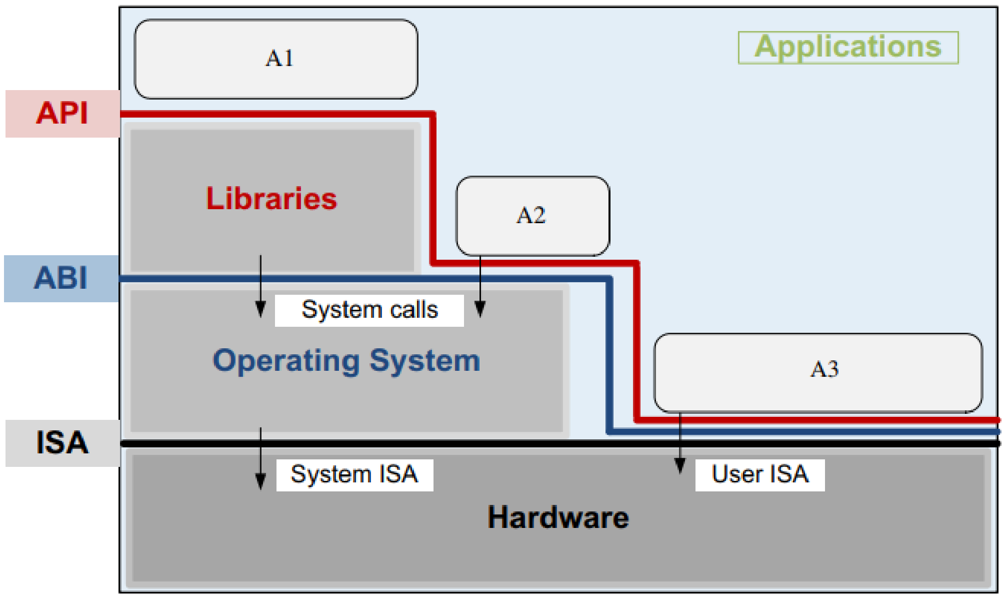


**FIGURE 5.1**

Layering and interfaces between layers in a computer system. The software components, including applications, libraries, and operating system, interact with the hardware via several interfaces: the *application programming interface* (API), the *application binary interface* (ABI), and the *instruction set architecture* (ISA). An application uses library functions (A1), makes system calls (A2), and executes machine instructions (A3).

27) **Define Virtual machine monitor. Mention its characteristics and functions in the cloud environment.**

*A virtual machine monitor (VMM), also called a hypervisor, is the software that securely partitions the resources of a computer system into one or more virtual machines.*

- ➔ A guest operating system is an operating system that runs under the control of a VMM rather than directly on the hardware. The VMM runs in kernel mode, whereas a guest OS runs in user mode.
- ➔ VMMs allow several operating systems to run concurrently on a single hardware platform; at the same time, VMMs enforce isolation among these systems, thus enhancing security.
- ➔ A VMM controls how the guest operating system uses the hardware resources. The events occurring in one VM do not affect any other VM running under the same VMM. At the same time, the VMM enables:
  - ➔ Multiple services to share the same platform.
  - ➔ The movement of a server from one platform to another, the so-called live migration.
  - ➔ System modification while maintaining backward compatibility with the original system.
- ➔ When a guest OS attempts to execute a privileged instruction, the VMM traps the operation and enforces the correctness and safety of the operation.
- ➔ The VMM guarantees the isolation of the individual VMs, and thus ensures security and encapsulation, a major concern in cloud computing.
- ➔ the VMM monitors system performance and takes corrective action to avoid performance degradation
- ➔ A VMM virtualizes the CPU and memory. For example, the VMM traps interrupts and dispatches them to the individual guest operating systems. If a guest OS disables interrupts, the VMM buffers such interrupts until the guest OS enables them.
- ➔ The VMM maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table. This shadow page table points to the actual page frame and is used by the hardware component called the memory management unit (MMU) for dynamic address translation.
- ➔ Memory virtualization has important implications on performance. VMMs use a range of optimization techniques; for example, VMware systems avoid page duplication among different virtual machines

28) **List and explain the function and Paravirtualization strategies for virtual memory management,CPU and I/O devices for the original x86 architecture.**

Paravirtualization. The guest operating system is modified to use only instructions that can be virtualized.

paravirtualization, in which each virtual machine runs on a slightly modified copy of the actual hardware.

The reasons that paravirtualization is often adopted are (i) some aspects of the hardware cannot be virtualized; (ii) to improve performance; and (iii) to present a simpler interface

Ex: Xen, Denali are Paravitulised

paravirtualization is done because some architectures such as x86 are not easily virtualizable. Paravirtualization demands that the guest OS be modified to run under the VMM; furthermore, the guest OS code must be ported for individual hardware platforms.

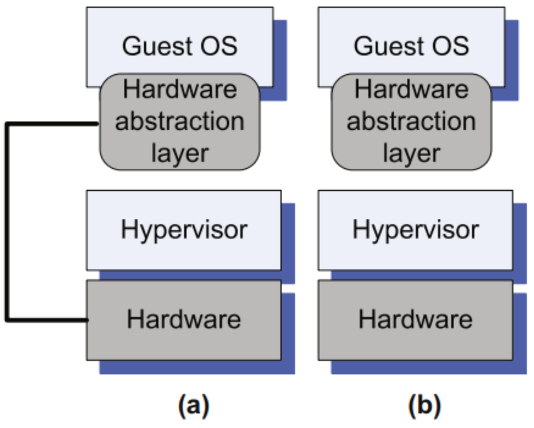


**FIGURE 5.4**

(a) Full virtualization requires the hardware abstraction layer of the guest OS to have some knowledge about the hardware. (b) Paravirtualization avoids this requirement and allows full compatibility at the application binary interface (ABI).

**Table 5.2** Paravirtualization strategies for virtual memory management, CPU multiplexing, and I/O devices for the original *x86 Xen* implementation.

| **Function** | **Strategy** |
|---|---|
| Paging | A domain may be allocated discontinuous pages. A guest OS has direct access to page tables and handles page faults directly for efficiency. Page table updates are batched for performance and validated by *Xen* for safety. |
| Memory | Memory is statically partitioned between domains to provide strong isolation. `XenoLinux` implements a *balloon driver* to adjust domain memory. |
| Protection | A guest OS runs at a lower priority level, in ring 1, while *Xen* runs in ring 0. |
| Exceptions | A guest OS must register with *Xen* a description table with the addresses of exception handlers previously validated. Exception handlers other than the page fault handler are identical to *x86* native exception handlers. |
| System calls | To increase efficiency, a guest OS must install a "fast" handler to allow system calls from an application to the guest OS and avoid indirection through *Xen*. |
| Interrupts | A lightweight event system replaces hardware interrupts. Synchronous system calls from a domain to *Xen* use *hypercalls*, and notifications are delivered using the asynchronous event system. |
| Multiplexing | A guest OS may run multiple applications. |
| Time | Each guest OS has a timer interface and is aware of "real" and "virtual" time. |
| Network and I/O devices | Data is transferred using asynchronous I/O rings. A ring is a circular queue of descriptors allocated by a domain and accessible within *Xen*. |
| Disk access | Only *Dom0* has direct access to IDE and SCSI disks. All other domains access persistent storage through the virtual block device (VBD) abstraction. |

***29) Can virtualization empower the creators of malware to carry out their mischievous activities with impunity and with minimal danger of being detected? How difficult is it to implement such a system? What are the means to prevent this type of malware to be put in place?***

It is well understood that in a layered structure a defense mechanism at some layer can be disabled by malware running a layer below it. Thus, the winner in the continuous struggle between the attackers and the defenders of a computing system is the one in control of the lowest layer of the software stack – the one that controls the hardware (see Figure 5.10).

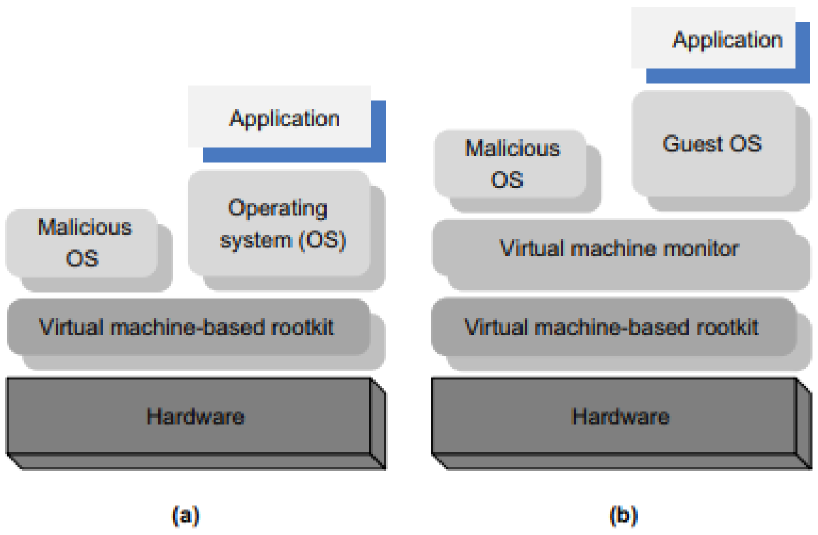


**FIGURE 5.10**

The insertion of a *virtual machine-based rootkit* (VMBR) as the lowest layer of the software stack running on the physical hardware. (a) Below an operating system; (b) Below a legitimate virtual machine monitor. The VMBR enables a malicious OS to run surreptitiously and makes it invisible to the genuine or the guest OS and to the application.

- ➔ a VMM allows a guest operating system to run on virtual hardware. The VMM offers to the guest operating systems a hardware abstraction and mediates its access to the physical hardware. We argued that a VMM is simpler and more compact than a traditional operating system; thus, it is more secure. But what if the VMM itself is forced to run above another software layer so that it is prevented from exercising direct control of the physical hardware?
- ➔ it is feasible to insert a "rogue VMM" between the physical hardware and an operating system. Such a rogue VMM is called a **virtual machine-based rootkit** (VMBR). The term rootkit refers to malware with privileged access to a system. The name comes from root, the most privileged account on a Unix system, and kit, a set of software components.
- ➔ It is also feasible to insert the VMBR between the physical hardware and a "legitimate VMM." As a virtual machine running under a legitimate VMM sees virtual hardware, the guest OS will not notice any change of the environment; so the only trick is to present the legitimate VMM with a hardware abstraction, rather than allow it to run on the physical hardware.
- ➔ The only way for a VMBR to take control of a system is to modify the boot sequence and to first load the malware and only then load the legitimate VMM or the operating system. This is only possible if the attacker has root privileges. Once the VMBR is loaded it must also store its image on the persistent storage.
- ➔ The VMBR can enable a separate malicious OS to run surreptitiously and make this malicious OS invisible to the guest OS and to the application running under it. Under the protection of the VMBR, the malicious OS could
  - ➔ (i) observe the data, the events, or the state of the target system;
  - ➔ (ii) run services such as spam relays or distributed denial-of-service attacks; or (
  - ➔ iii) interfere with the application

**30) Discuss the means through which virtualization simulates the interface to a physical object and mention the VMM characteristics.**

Virtualization simulates the interface to a physical object by any one of four means:

- ➔ **Multiplexing**. Create multiple virtual objects from one instance of a physical object. For example, a processor is multiplexed among a number of processes or threads.

- ➔ **Aggregation**. Create one virtual object from multiple physical objects. For example, a number of physical disks are aggregated into a RAID disk.
- ➔ **Emulation**. Construct a virtual object from a different type of physical object. For example, a physical disk emulates a random access memory.
- ➔ **Multiplexing and emulation**. Examples: Virtual memory with paging multiplexes real memory and disk, and a Virtual address emulates a real address; TCP emulates a reliable bit pipe and multiplexes a physical communication channel and a processor.

Virtualization is a critical aspect of cloud computing, equally important to the providers and consumers of cloud services, and plays an important role in:

• System security because it allows isolation of services running on the same hardware.
• Performance and reliability because it allows applications to migrate from one platform to another.
• The development and management of services offered by a provider.
• Performance isolation.

- ➔ Using Virtual Machine Manager allows us to do this in an automated way. It also allows what's called intelligent placement. So when a virtual machine is deployed, Virtual Machine Manager analyzes the data and the resource requirements for both the workload that's going to happen on the virtual machine and the host that hosts the virtual machine. This analysis allows the Virtual Machine Manager administrator to fine-tune placement algorithms to get the customized deployment recommendations.

- ➔ It also allows for centralized resource management and optimization. This provides a central work area for performing resource tuning. The various different settings can be changed on virtual machines without interrupting the workloads, and virtual machines can be migrated from one host to another, and we can then optimize those physical resources so we have the proper amount of RAM, hard drive space, and processors. Rapid deployment and migration of virtual machines is also a great feature.

- ➔ Virtual Machine Manager enables quick provisioning of new virtual machines. That means we can deploy them very quickly to other hosts that can host Hyper-V virtual machines. And we use a wizard to do this, which allows this to be a very simple process. Virtual Machine Manager administrators can rapidly deploy virtual machines across the entire enterprise. That means if you have multiple domains or forests, it can cover all of them using the Virtual Machine Manager.

- ➔ It also allows for management and migration of existing virtual machines, so if you'd like to move or manage virtual machines that were already deployed prior to your installation you can manage those and move those as well. Virtual Machine Manager comes with a centralized library. That includes all the building blocks for making virtual machines such as virtual hard disks, ISO images, which take the place of DVDs, post-deployment customization scripts to automate the post-deployment of a virtual machine, special hardware profiles in case you want to have a profile for a specific task that you need to use over and over again.

- ➔ It also allows for guest operating system profiles as well as virtual machine templates that can also be used repeatedly. Templates represent standard virtual machine configurations similar to the mini setup and CIS prep utility that comes with all Windows deployments. Templates also encapsulate best practices regarding hardware and guest operating system configuration. That will help you stay secure, and also make sure you have uniform access across the environment.

- ➔ Centralized monitoring and reporting helps the Virtual Machine Manager administrator with reports and monitoring data from within the Virtual Machine Manager console. It integrates with Operations Manager. These capabilities can be extended even further by using Operation Manager directly if you have another server running that particular application inside the System Center suite. You don't have to be an administrator in order to provision a Hyper-V virtual machine.
- ➔ You can set up self-service provisioning. That allows end users to be able to be grantedcontrolled access to specific virtual machines, templates, and other Virtual Machine Manager resources by way of a web-based portal. This particular access enables end users such as test and development users to quickly provision new virtual machines for themselvesaccording to controls set by the administrator. You can also take advantage of existing SAN networks.
- ➔ Virtual machine images can be very large, and it can be slow to move them across a LAN network. You can configure Virtual Machine Manager for use in an environment that has a fiber channel or an iSCSI storage and network SAN and perform SAN transfers within the Virtual Machine Manager. SAN, of course, stands for Storage Area Network, and it is physical storage that can be attached to any physical server. It can also be used to be attached to virtual servers to expand their storage as well.

**31) Examine the problems faced by virtualization of the x86 architecture and suggest the Hardware support for virtualization.**

hardware support for virtualization was necessary, and Intel and AMD started work on the first-generation virtualization extensions of the x86 3 architecture. In 2005 Intel released two Pentium 4 models supporting VT-x, and in 2006 AMD announced Pacifica and then several Athlon 64 models. A 2006 paper [253] analyzes the challenges to virtualizing Intel architectures and then presents VT-x and VT-i virtualization architectures for x86 and Itanium architectures,

- ➔ ***Ring deprivileging.*** This means that a VMM forces the guest software, the operating system, and the applications to run at a privilege level greater than 0. Recall that the x86 architecture provides four protection rings at levels 0–3. Two solutions are then possible: (a) The (0/1/3) mode, in which the VMM, the OS, and the application run at privilege levels 0, 1, and 3, respectively; or (b) the (0,3,3) mode, in which the VMM, a guest OS, and applications run at privilege levels 0, 3, and 3, respectively. The first mode is not feasible for x86 processors in 64-bit mode, as we shall see shortly.
- ➔ ***Ring aliasing.*** Problems created when a guest OS is forced to run at a privilege level other than that it was originally designed for. For example, when the CR register4 is PUSHed, the current privilege level is also stored on the stack [253].
- ➔ ***Address space compression***. A VMM uses parts of the guest address space to store several system data structures, such as the interrupt-descriptor table and the global-descriptor table. Such data structures must be protected, but the guest software must have access to them.
- ➔ ***Nonfaulting access to privileged state***. Several instructions, LGDT, SIDT, SLDT, and LTR that load the registers GDTR, IDTR, LDTR, and TR, can only be executed by software running at privilege level 0, because these instructions point to data structures that control the CPU operation.
- ➔ ***Guest system calls.*** Two instructions, SYSENTER and SYSEXIT, support low-latency system calls. The first causes a transition to privilege level 0, whereas the second causes a

transition from privilege level 0 and fails if executed at a level higher than 0. The VMM must then emulate every guest execution of either of these instructions, which has a negative impact on performance.

- ➔ ***Interrupt virtualization.*** In response to a physical interrupt, the VMM generates a "virtual interrupt" and delivers it later to the target guest OS. But every OS has the ability to mask interrupts5; thus the virtual interrupt could only be delivered to the guest OS when the interrupt is not masked. Keeping track of all guest OS attempts to mask interrupts greatly complicates the VMM and increases the overhead.
- ➔ ***Access to hidden state.*** Elements of the system state (e.g., descriptor caches for segment registers) are hidden; there is no mechanism for saving and restoring the hidden components when there is a context switch from one VM to another.
- ➔ ***Ring compression***. Paging and segmentation are the two mechanisms to protect VMM code from being overwritten by a guest OS and applications. Systems running in 64-bit mode can only use paging, but paging does not distinguish among privilege levels 0, 1, and 2, so the guest OS must run at privilege level 3, the so-called (0/3/3) mode. Privilege levels 1 and 2 cannot be used; thus the name ring compression.
- ➔ ***Frequent access to privileged resources increases*** VMM overhead. The task-priority register (TPR) is frequently used by a guest OS. The VMM must protect the access to this register and trap all attempts to access it. This can cause a significant performance degradation.

**32) Explain the technique of achieving the optimization of network virtualization in xen 2.0 with a neat diagram.**

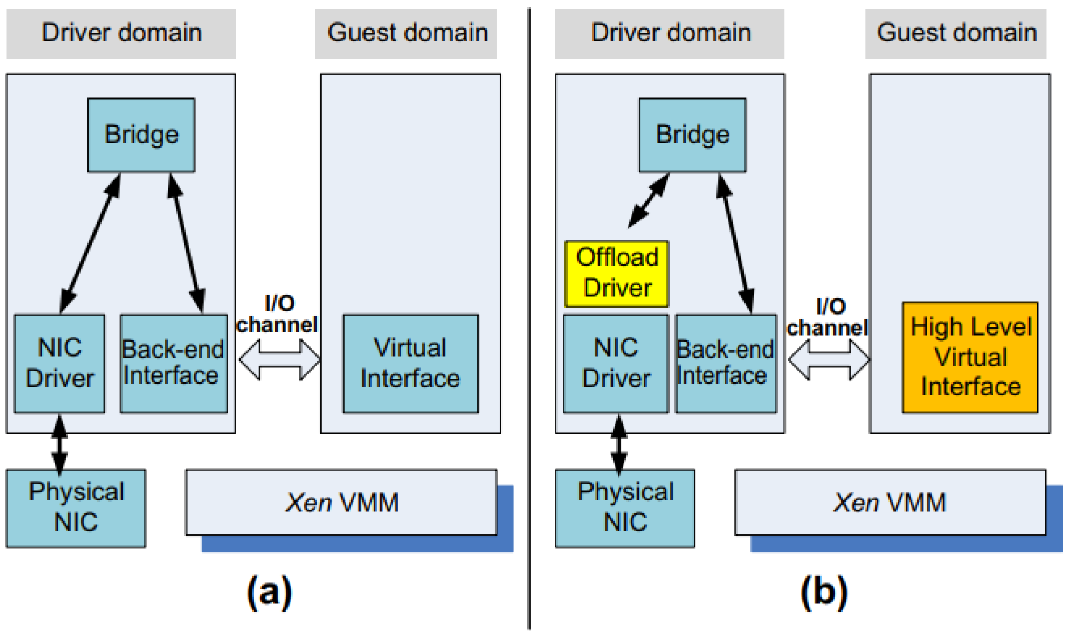


**FIGURE 5.8**

*Xen* network architecture. (a) The original architecture. (b) The optimized architecture.

The Xen network optimization
(i) the virtual interface; (ii) the I/O channel; and (iii) the virtual memory.

- ➔ the virtual interface; There is a tradeoff between generality and flexibility on one hand and performance on the other hand. The original virtual network interface provides the guest domain with the abstraction of a simple low-level network interface supporting sending and receiving primitives. This design supports a wide range of physical devices attached to the driver domain but does not take advantage of the capabilities of some physical NICs such as checksum offload and scatter-gather DMA support. These features are supported by the high-level virtual interface of the optimized system.

- ➔ The next target of the optimization effort is the communication between the guest domain and the driver domain. Rather than copying a data buffer holding a packet, each packet is allocated in a new page and then the physical page containing the packet is remapped into the target domain.
- ➔ The next target of the optimization effort is the communication between the guest domain and the driver domain. Rather than copying a data buffer holding a packet, each packet is allocated in a new page and then the physical page containing the packet is remapped into the target domain.
- ➔ The third optimization covers virtual memory. Virtual memory in Xen 2.0 takes advantage of the superpage and global page-mapping hardware features available on Pentium and Pentium Pro processors. A superpage increases the granularity of the dynamic address translation; a superpage entry covers 1, 024 pages of physical memory, and the address translation mechanism maps a set of contiguous pages to a set of contiguous physical pages.
- ➔ When new processes are created, the guest OS must allocate read-only pages for the page tables of the address spaces running under the guest OS, and that forces the system to use traditional page mapping rather than superpage mapping. The optimized version uses a special memory allocator to avoid this problem.

**33) Define Virtual machine. Distinguish the two types of VM. Explain it with a neat diagram and examples.**

A virtual machine (VM) is an isolated environment that appears to be a whole computer but actually only has access to a portion of the computer resources.

Types of VM -: ***Process VMs and System VMs***

- ➔ A **process** VM is a virtual platform created for an individual process and destroyed once the process terminates. Virtually all operating systems provide a process VM for each one of the applications running, but the more interesting process VMs are those that support binaries compiled on a different instruction set.
- ➔ A **system** VM supports an operating system together with many user processes. When the VM runs under the control of a normal OS and provides a platform-independent host for a single application, we have an application virtual machine (e.g., Java Virtual Machine [JVM]).

- ➢ **Traditional**. VM also called a "bare metal" VMM. A thin software layer that runs directly on the host machine hardware; its main advantage is performance [see Figure 5.3(b)]. Examples: VMWare ESX, ESXi Servers, Xen, OS370, and Denali.
- ➢ **Hybrid**. The VMM shares the hardware with the existing OS [see Figure 5.3(c)]. Example: VMWare Workstation.
- ➢ **Hosted**. The VM runs on top of an existing OS [see Figure 5.3(d)]. The main advantage of this approach is that the VM is easier to build and install. Another advantage of this solution is that the VMM could use several components of the host OS, such as the scheduler, the pager, and the I/O drivers, rather than providing its own., Ex. User-Mode Linux

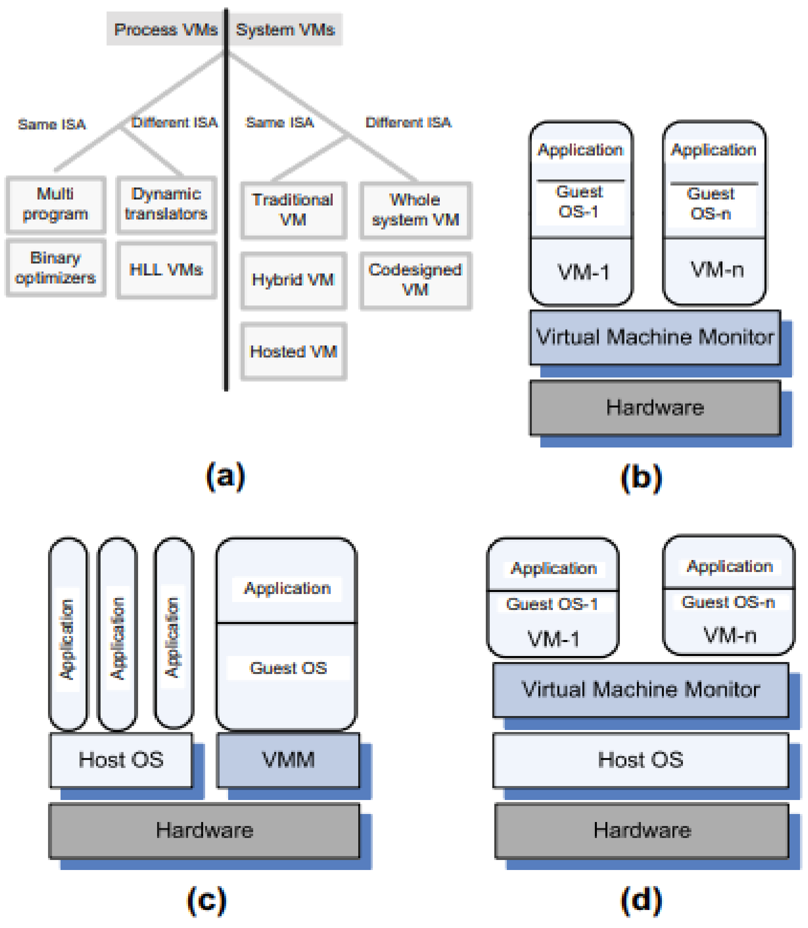


**FIGURE 5.3**

(a) A taxonomy of process and system VMs for the same and for different ISAs. Traditional, hybrid, and hosted are three classes of VM for systems with the same ISA. (b) Traditional VMs. The VMM supports multiple VMs and runs directly on the hardware. (c) A hybrid VM. The VMM shares the hardware with a host operating system and supports multiple virtual machines. (d) A hosted VM. The VMM runs under a host operating system.

**34) Define performance and security isolation in Virtual Machine Monitor.**

Performance isolation is a critical condition for quality-of-service (QoS) guarantees in shared computing environments. Indeed, if the run-time behavior of an application is affected by other applications running concurrently and, thus, is competing for CPU cycles, cache, main memory, and disk and network access, it is rather difficult to predict the completion time. Moreover, it is equally difficult to optimize the application.

Several operating systems, including Linux/RK [270], QLinux [343], and SILK [44], support some performance isolation, but problems still exist because one has to account for all resources used and to distribute the overhead for different system activities, including context switching and paging, to individual users – a problem often described as QoS crosstalk.

➔ Processor virtualization presents multiple copies of the same processor or core on multicore systems. The code is executed directly by the hardware, whereas processor emulation presents a model of another hardware system in which instructions are "emulated" in software more slowly than virtualization. An example is Microsoft's VirtualPC, which could run on chip sets other than the x86 family. It was used on Mac hardware until Apple adopted Intel chips.

➔ Traditional operating systems multiplex multiple processes or threads, whereas a virtualization supported by a VMM multiplexes full operating systems. Obviously, there is a performance penalty because an OS is considerably more heavyweight than a process and the overhead of context switching is larger

- ➔ Operating systems use process abstraction not only for resource sharing but also to support isolation. Unfortunately, this is not sufficient from a security perspective. Once a process is compromised, it is rather easy for an attacker to penetrate the entire system.
- ➔ A VMM is a much simpler and better specified system than a traditional operating system. For example, the Xen VMM., whereas the Denali VMM [372] has only about half that, or 30,000 lines of code. The security vulnerability of VMMs is considerably reduced because the systems expose a much smaller number of privileged functions.

**35) Distinguish the two types of virtual machines and give examples for each type of virtual machine.**

Refer 33

**36) Define layering. Justify with a diagram and explanation of achieving Code portability in cloud computing.**

*Refer 26*

*the binaries created by a compiler for a specific ISA and a specific operating system are not portable. Such code cannot run on a computer with a different ISA or on computers with the same ISA but different operating systems. However, it is possible to compile an HLL program for a VM environment, as shown in Figure 5.2, where portable code is produced and distributed and then converted by binary translators to the ISA of the host system. A dynamic binary translation converts blocks of guest instructions from the portable code to the host instruction and leads to a significant performance improvement as such blocks are cached and reused.*

**37) List the means by which Virtualization simulates the interface to a physical object and mentions its important role.**

*Refer 30*

**38) Mention the architectural enhancement provided by the VT-x.Define Virtual machine monitor and summarize its importance in virtualization.**

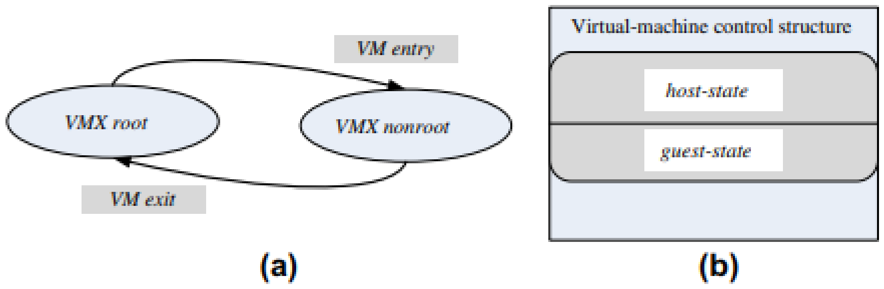


**FIGURE 5.5**

(a) The two modes of operation of VT-x, and the two operations to transit from one to another. (b) The VMCS includes *host-state* and *guest-state* areas that control the *VM entry* and *VM exit* transitions.

- ➔ A major architectural enhancement provided by the VT-x is the support for two modes of operations and a new data structure called the virtual machine control structure (VMCS), including host-state and guest-state areas.
- ➔ VMX root. Intended for VMM operations and very close to the x86 without VT-x.
- ➔ VMX nonroot. Intended to support a VM
- ➔ When executing a VM entry operation, the processor state is loaded from the guest-state of the VM scheduled to run; then the control is transferred from the VMM to the VM. A VM exitsaves the processor state in the guest-state area of the running VM; then it loads the

processor state from the host-state area and finally transfers control to the VMM. Note that all VM exit operations use a common entry point to the VMM.

- ➔ Each VM exit operation saves the reason for the exit and, eventually, some qualifications in VMCS. Some of this information is stored as bitmaps. For example, the exception bitmap specifies which one of 32 possible exceptions caused the exit. The I/O bitmap contains one entry for each port in a 16-bit I/O space.
- ➔ Processors based on two new virtualization architectures, VT-d 6 and VT-c, have been developed. The first supports the I/O memory management unit (I/O MMU) virtualization and the second supports network virtualization.
- ➔ Also known as PCI pass-through, I/O MMU virtualization gives VMs direct access to peripheral devices. VT-d supports:
  - ➔ DMA address remapping, which is address translation for device DMA transfers.
  - ➔ Interrupt remapping, which is isolation of device interrupts and VM routing.
  - ➔ I/O device assignment, in which an administrator can assign the devices to a VM in any configuration.
  - ➔ Reliability features, which report and record DMA and interrupt errors that may otherwise corrupt memory and impact VM isolation.

**UNIT 4**

**39) Compare Start time fair queuing with borrowed virtual time with respect to scheduling of resources, management and performance.**

**Start -Time Fair Queuing(SFQ)**

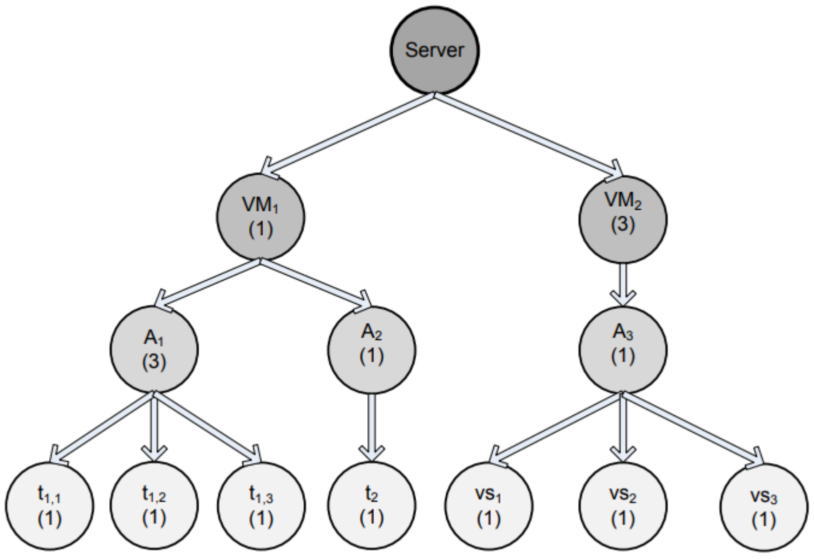
Organize the consumers of the CPU bandwidth in a tree structure.

The root node is the processor and the leaves of this tree are the threads of each application.

¨ When a virtual machine is not active, its bandwidth is reallocated to the other VMs active at the time.

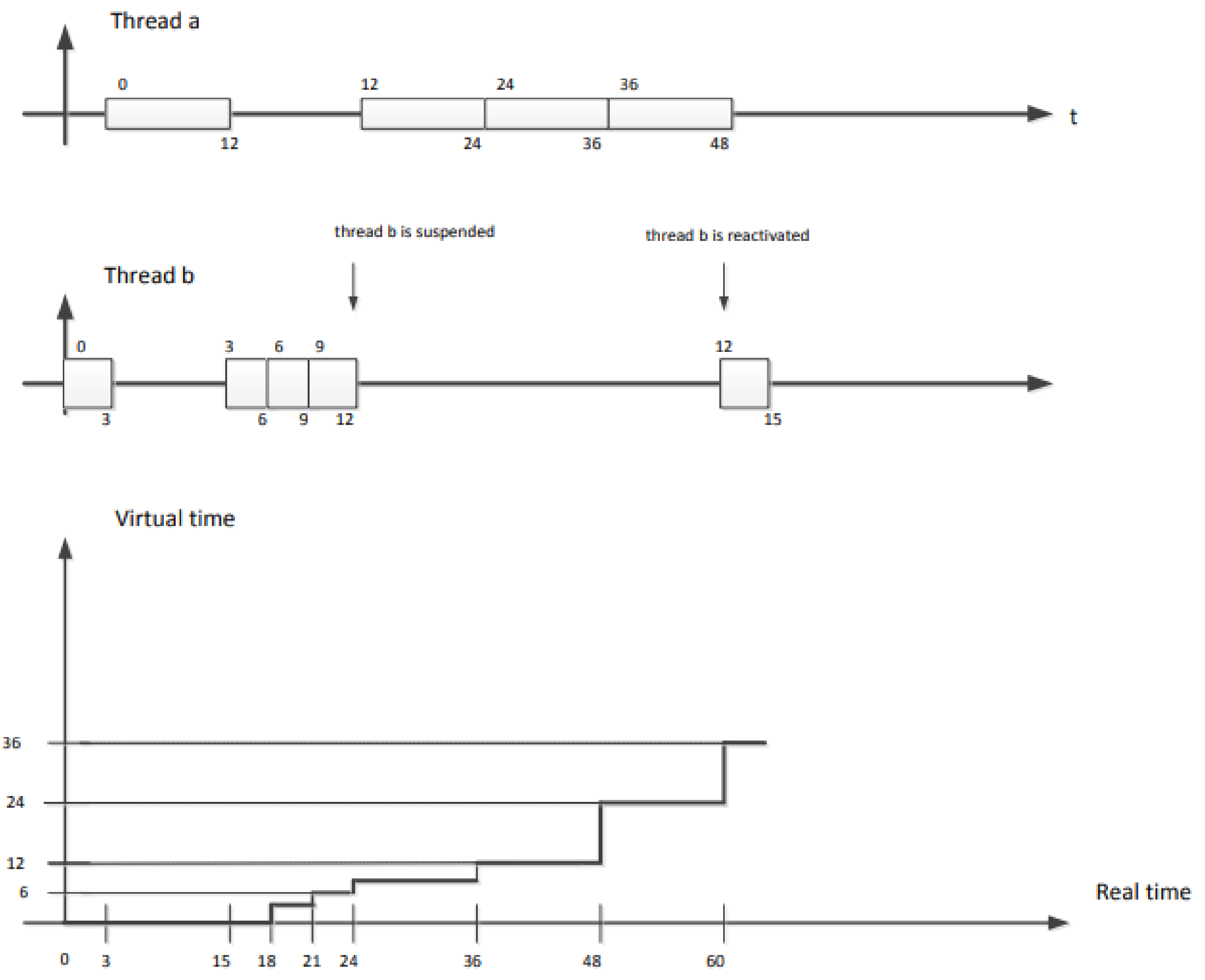
¨ When one of the applications of a virtual machine is not active, its allocation is transferred to the other applications running on the same VM.

¨ If one of the threads of an application is not runnable then its allocation is transferred to the other threads of the applications.



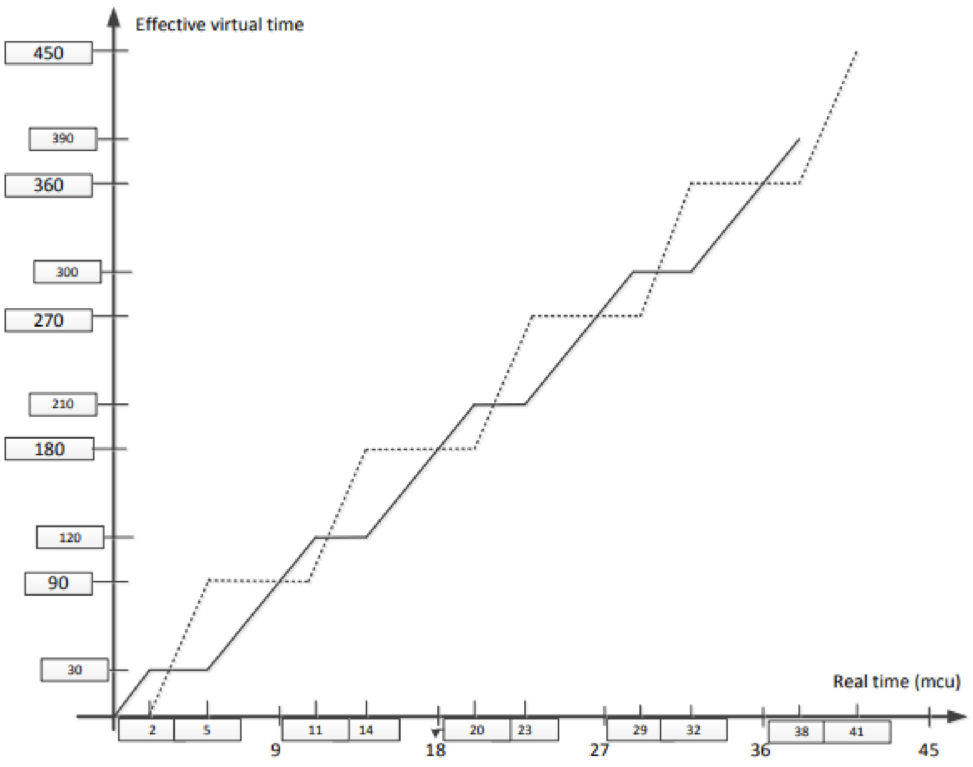
The SFQ tree for scheduling when two virtual machines VM1 and VM2 run on a powerful server

- Top à the virtual startup time and the virtual finish time and function of the real time t for each activation of threads a and b.
- Bottom à the virtual time of the scheduler v(t) function of the real time



**BVT**

- ➔ Objective - support low-latency dispatching of real-time applications, and weighted sharing of CPU among several classes of applications.
- ➔ A thread i has
  - ➔ ¨ an effective virtual time, Ei .
  - ➔ ¨ an actual virtual time, Ai .
  - ➔ ¨ a virtual time warp, Wi .
- ➔ The scheduler thread maintains its own scheduler virtual time (SVT) defined as the minimum actual virtual time of any thread.
- ➔ The threads are dispatched in the order of their effective virtual time, policy called the Earliest Virtual Time (EVT).
- ➔ Context switches are triggered by events such as:
  - ➔ ¨ the running thread is blocked waiting for an event to occur.
  - ➔ ¨ the time quantum expires.
  - ➔ ¨ an interrupt occurs.
  - ➔ ¨ when a thread becomes runnable after sleeping.



The effective virtual time and the real time of the threads ***a*** (solid line) and ***b*** (dotted line) with weights $w_a = 2\ w_b$ when the actual virtual time is incremented in steps of 90 mcu.

**40) Can specialized autonomic performance managers cooperate to optimize power consumption and, at the same time, satisfy the requirements of SLAs?**

Virtually all modern processors support dynamic voltage scaling (DVS) as a mechanism for energy saving. Indeed, the energy dissipation scales quadratically with the supply voltage. The power management controls the CPU frequency and, thus, the rate of instruction execution. For some compute-intensive workloads the performance decreases linearly with the CPU clock frequency,

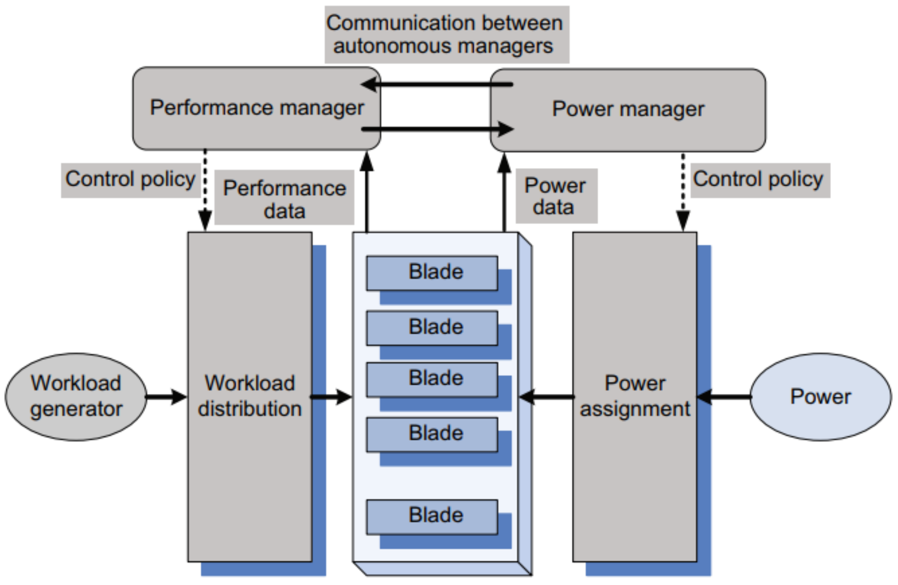


**FIGURE 6.3**

Autonomous performance and power managers cooperate to ensure SLA prescribed performance and energy optimization. They are fed with performance and power data and implement the performance and power management policies, respectively.

The approach to coordinating power and performance management in is based on several ideas:
• Use a joint utility function for power and performance. The joint performance-power utility function, Upp(R, P), is a function of the response time, R, and the power, P, and it can be of the form

$$U_{pp}(R,P) = U(R) - \epsilon \times P \quad \text{or} \quad U_{pp}(R,P) = \frac{U(R)}{P}, \tag{6.18}$$

with U(R) the utility function based on response time only and a parameter to weight the influence of the two factors, response time and power.
• Identify a minimal set of parameters to be exchanged between the two managers.
• Set up a power cap for individual systems based on the utility-optimized power management policy.
• Use a standard performance manager modified only to accept input from the power manager regarding the frequency determined according to the power management policy. The power manager consists of Tcl (Tool Command Language) and C programs to compute the per-server (per-blade) power caps and send them via IPMI5 to the firmware controlling the blade power. The power manager and the performance manager interact, but no negotiation between the two agents is involved.
Use standard software systems. For example, use the WebSphere Extended Deployment (WXD), middleware that supports setting performance targets for individual Web applications and for the monitor response time, and periodically recompute the resource allocation parameters to meet the targets set. Use the Wide-Spectrum Stress Tool from the IBM Web Services Toolkit as a workload generator.

41) **Explain and write the psuedocode of the ASCA combinatorial Auction algorithm. Say how the excess vector is computed and the proxies can be modeled. List the constraints for a combinatorial algorithm.**

*Refer 47*

42) **List the cloud resource management policies and illustrate the methodology for optimal resource management based on control theory concepts.**

**Cloud resource management policies** can be loosely grouped into five classes:
*1. Admission control.*
*2. Capacity allocation.*
*3. Load balancing.*
*4. Energy optimization.*
*5. Quality-of-service (QoS) guarantees.*

**Admission control**

The explicit goal of an admission control policy is to prevent the system from accepting workloads in violation of high-level system policies; for example, a system may not accept an additional workload that would prevent it from completing work already in progress or contracted. Limiting the workload requires some knowledge of the global state of the system. In a dynamic system such knowledge, when available, is at best obsolete.

**Capacity allocation**

Capacity allocation means to allocate resources for individual instances; an instance is an activation of a service. Locating resources subject to multiple global optimization constraints requires a search of a very large search space when the state of individual systems changes rapidly.

**Load balancing**

Load balancing and **energy optimization** can be done locally, but global load-balancing and energy optimization policies encounter the same difficulties as the one we have already discussed.

Load balancing and energy optimization are correlated and affect the cost of providing the services. Indeed, it was predicted that by 2012 up to 40% of the budget for IT enterprise infrastructure would be spent on energy.

**Quality-of-service (QoS) guarantees**

Quality of service is that aspect of resource management that is probably the most difficult to address and, at the same time, possibly the most critical to the future of cloud computing.

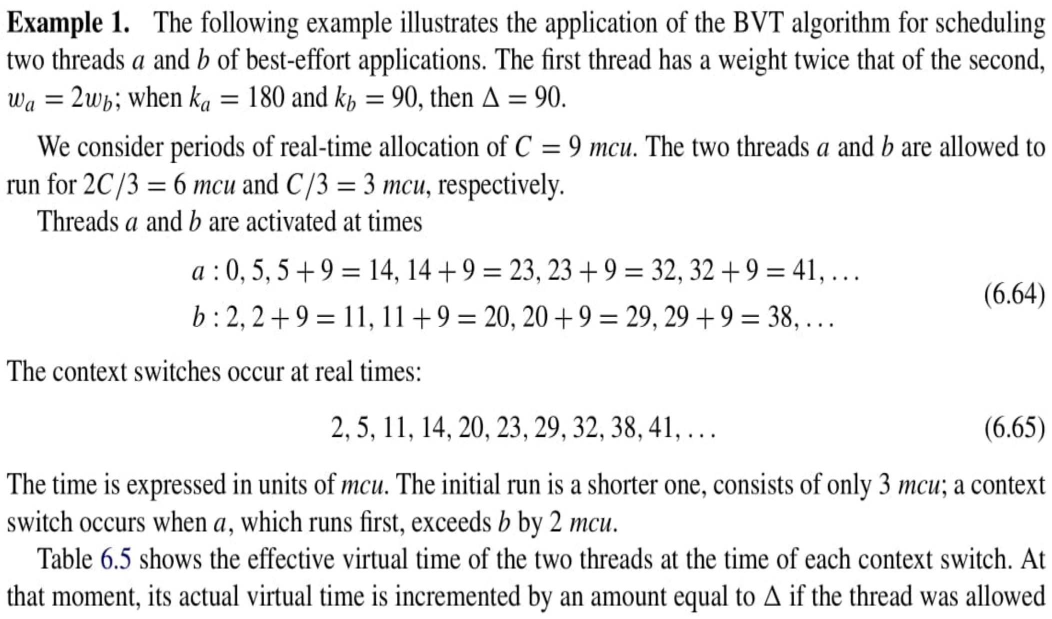
**Page No – 166,167,168**

43) **Outline the loosely grouped five classes of cloud resource management policies and brief the four basic mechanisms used for implementation of resource management policies.**

*Refer 42.*

- ➔ ***Control theory***. Control theory uses the feedback to guarantee system stability and predict transient behavior but can be used only to predict local rather than global behavior. Kalman filters have been used for unrealistically simplified models.
- ➔ ***Machine learning***. A major advantage of machine learning techniques is that they do not need a performance model of the system. This technique could be applied to coordination of several autonomic system managers, .
- ➔ ***Utility-based***. Utility-based approaches require a performance model and a mechanism to correlate user-level performance with cost.
- ➔ ***Market-oriented/economic mechanisms***. Such mechanisms do not require a model of the system, e.g., combinatorial auctions for bundles of resources.

44) **Illustrate the application of the BVT algorithm for scheduling two threads a and b of best-effort applications. The first thread has a weight twice that of the second, Wa = 2Wb; when Ka = 180 and Kb = 90, then Δ= 90. We consider periods of real-time allocation of C = 9 mcu. The two threads a and b are allowed to run for 2C/3 = 6 mcu and C/3 = 3 mcu, respectively.**



**Example 1.** The following example illustrates the application of the BVT algorithm for scheduling two threads $a$ and $b$ of best-effort applications. The first thread has a weight twice that of the second, $w_a = 2w_b$; when $k_a = 180$ and $k_b = 90$, then $\Delta = 90$.

We consider periods of real-time allocation of $C = 9$ *mcu*. The two threads $a$ and $b$ are allowed to run for $2C/3 = 6$ *mcu* and $C/3 = 3$ *mcu*, respectively.

Threads $a$ and $b$ are activated at times

$$\begin{aligned} &a : 0, 5, 5+9 = 14, 14+9 = 23, 23+9 = 32, 32+9 = 41, \ldots \\ &b : 2, 2+9 = 11, 11+9 = 20, 20+9 = 29, 29+9 = 38, \ldots \end{aligned} \quad (6.64)$$

The context switches occur at real times:

$$2, 5, 11, 14, 20, 23, 29, 32, 38, 41, \ldots \quad (6.65)$$

The time is expressed in units of *mcu*. The initial run is a shorter one, consists of only 3 *mcu*; a context switch occurs when $a$, which runs first, exceeds $b$ by 2 *mcu*.

Table 6.5 shows the effective virtual time of the two threads at the time of each context switch. At that moment, its actual virtual time is incremented by an amount equal to $\Delta$ if the thread was allowed

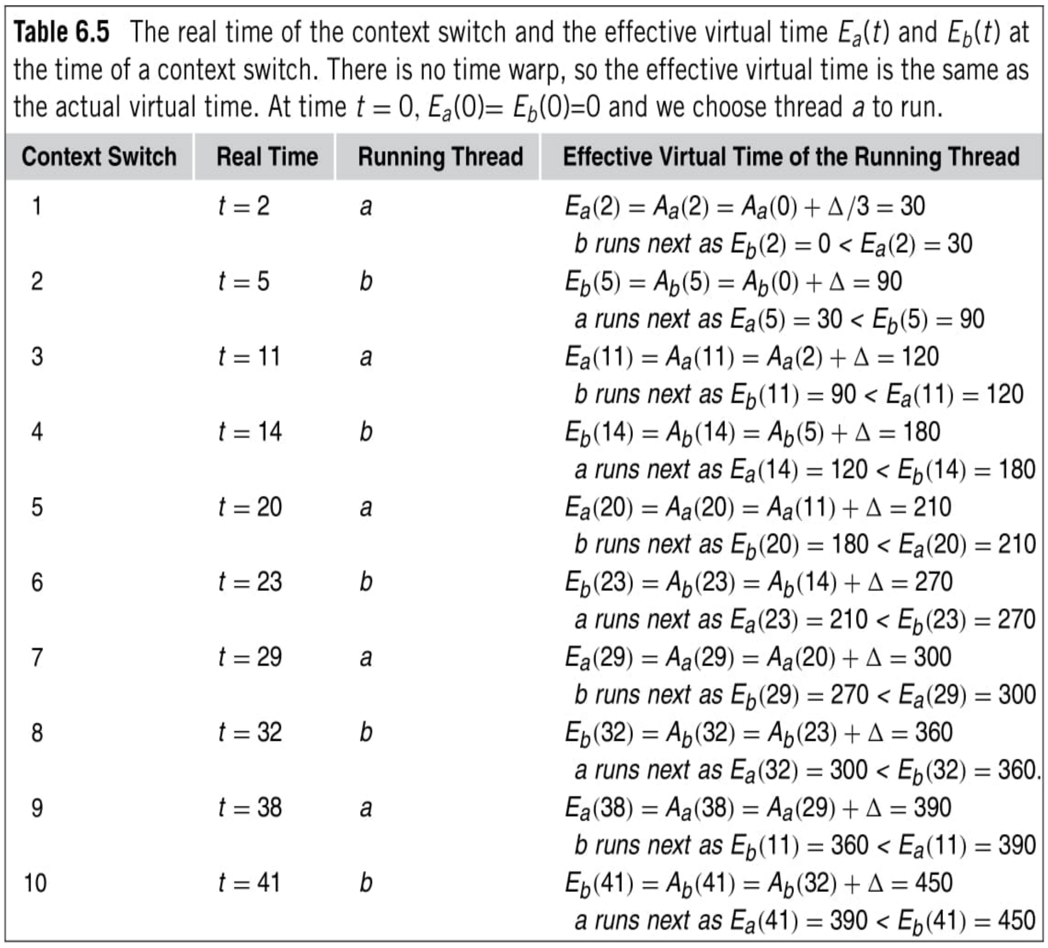
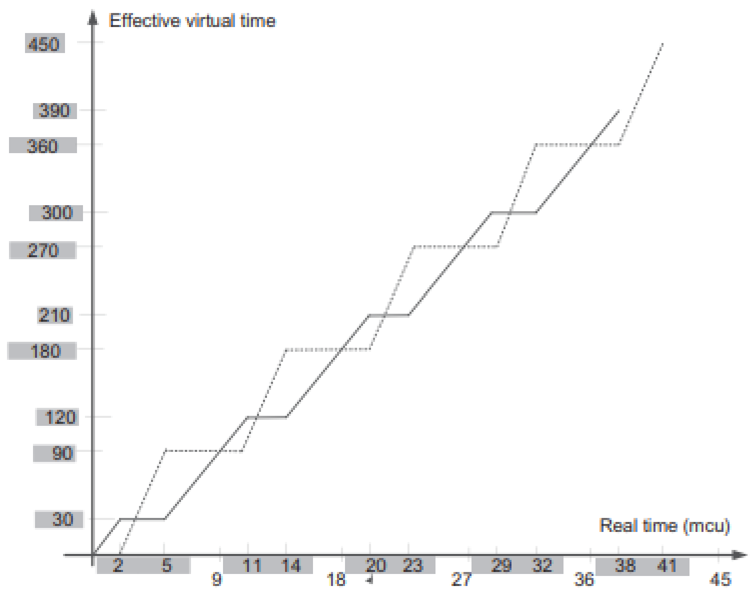


**Table 6.5** The real time of the context switch and the effective virtual time $E_a(t)$ and $E_b(t)$ at the time of a context switch. There is no time warp, so the effective virtual time is the same as the actual virtual time. At time $t = 0$, $E_a(0)= E_b(0)=0$ and we choose thread $a$ to run.

| Context Switch | Real Time | Running Thread | Effective Virtual Time of the Running Thread |
|---|---|---|---|
| 1 | $t = 2$ | $a$ | $E_a(2) = A_a(2) = A_a(0) + \Delta/3 = 30$ <br> *b runs next as* $E_b(2) = 0 < E_a(2) = 30$ |
| 2 | $t = 5$ | $b$ | $E_b(5) = A_b(5) = A_b(0) + \Delta = 90$ <br> *a runs next as* $E_a(5) = 30 < E_b(5) = 90$ |
| 3 | $t = 11$ | $a$ | $E_a(11) = A_a(11) = A_a(2) + \Delta = 120$ <br> *b runs next as* $E_b(11) = 90 < E_a(11) = 120$ |
| 4 | $t = 14$ | $b$ | $E_b(14) = A_b(14) = A_b(5) + \Delta = 180$ <br> *a runs next as* $E_a(14) = 120 < E_b(14) = 180$ |
| 5 | $t = 20$ | $a$ | $E_a(20) = A_a(20) = A_a(11) + \Delta = 210$ <br> *b runs next as* $E_b(20) = 180 < E_a(20) = 210$ |
| 6 | $t = 23$ | $b$ | $E_b(23) = A_b(23) = A_b(14) + \Delta = 270$ <br> *a runs next as* $E_a(23) = 210 < E_b(23) = 270$ |
| 7 | $t = 29$ | $a$ | $E_a(29) = A_a(29) = A_a(20) + \Delta = 300$ <br> *b runs next as* $E_b(29) = 270 < E_a(29) = 300$ |
| 8 | $t = 32$ | $b$ | $E_b(32) = A_b(32) = A_b(23) + \Delta = 360$ <br> *a runs next as* $E_a(32) = 300 < E_b(32) = 360.$ |
| 9 | $t = 38$ | $a$ | $E_a(38) = A_a(38) = A_a(29) + \Delta = 390$ <br> *b runs next as* $E_b(11) = 360 < E_a(11) = 390$ |
| 10 | $t = 41$ | $b$ | $E_b(41) = A_b(41) = A_b(32) + \Delta = 450$ <br> *a runs next as* $E_a(41) = 390 < E_b(41) = 450$ |



45) Use the start-time fair queuing (SFQ) scheduling algorithm to compute the virtual start-up and the virtual finish time for two threads a and b with weights wa = 1 and wb = 4 when the time quantum is q = 12 and thread b blocks at time t = 24 and wakes up at time t = 60. Plot the virtual time of the scheduler function of the real time.

As in Problem 6, we consider two threads with the weights $w_a = 1$ and $w_b = 5$ and the time quantum is $q = 15$, and thread $b$ blocks at time $t = 24$ and wakes up at time $t = 60$.

Initially $S_a^0 = 0$, $S_b^0 = 0$, $v_a(0) = 0$, and $v_b(0) = 0$. The scheduling decisions are made as follows:

1. t=0: we have a tie, $S_a^0 = S_b^0$ and arbitrarily thread $b$ is chosen to run first; the virtual finish time of thread $b$ is

$$F_b^0 = S_b^0 + q/w_b = 0 + 15/5 = 3. \tag{22}$$

2. t=3: both threads are runnable and thread $b$ was in service, thus, $v(3) = S_b^0 = 0$; then

$$S_b^1 = \max[v(3), F_b^0] = \max(0, 3) = 3. \tag{23}$$

But $S_a^0 < S_b^1$ thus thread $a$ is selected to run. Its virtual finish time is

$$F_a^0 = S_a^0 + q/w_a = 0 + 15/1 = 15. \tag{24}$$

3. t=18: both threads are runnable and thread $a$ was in service at this time thus,

$$v(18) = S_a^0 = 0 \tag{25}$$

and

$$S_a^1 = \max[v(18), F_a^0] = \max[0, 15] = 15. \tag{26}$$

As $S_b^1 = 3 < 12$, thread $b$ is selected to run; the virtual finish time of thread $b$ is now

$$F_b^1 = S_b^1 + q/w_b = 3 + 15/5 = 6. \tag{27}$$

4. t=21: both threads are runnable and thread $b$ was in service at this time, thus,

$$v(21) = S_b^1 = 3 \tag{28}$$

and

$$S_b^2 = \max[v(21), F_b^1] = \max[3, 6] = 6. \tag{29}$$

As $S_b^2 < S_a^1 = 15$, thread $b$ is selected to run again; its virtual finish time is

$$F_b^2 = S_b^2 + q/w_b = 6 + 15/5 = 9. \tag{30}$$

5. t=24: Thread $b$ was in service at this time, thus,

$$v(24) = S_b^2 = 6 \tag{31}$$

$$S_b^3 = \max[v(24), F_b^2] = \max[6, 9] = 9. \tag{32}$$

Thread $b$ is suspended till $t = 60$, thus, the thread $a$ is activated; its virtual finish time is

$$F_a^1 = S_a^1 + q/w_a = 9 + 15/1 = 24. \tag{33}$$

6. t=39: thread $a$ was in service and it is the only runnable thread at this time, thus,

$$v(39) = S_a^1 = 15 \tag{34}$$

and

$$S_a^1 = \max[v(39), F_a^1] = \max[15, 24] = 24. \tag{35}$$

Then,

$$F_a^2 = S_a^2 + q/w_a = 24 + 15/1 = 39. \tag{36}$$

7. t=54: thread $a$ was in service and it is the only runnable thread at this time thus,

$$v(54) = S_a^1 = 24 \tag{37}$$

and

$$S_a^2 = \max[v(54), F_a^2] = \max[24, 39] = 39. \tag{38}$$

Then,

$$F_a^2 = S_a^3 + q/w_a = 39 + 15/1 = 54. \tag{39}$$

8. t=69: thread $a$ was in service at this time, thus,

$$v(69) = S_a^2 = 39 \tag{40}$$

and

$$S_a^4 = \max[v(69), F_a^2] = \max[39, 54] = 54. \tag{41}$$

But now thread $b$ is runnable and $S_b^3 = 9$.

Thus, thread $b$ is activated and

$$F_b^3 = S_b^3 + q/w_b = 9 + 15/5 = 12. \tag{42}$$

**46) Write the objective of Pricing and Allocation algorithm. Write its characteristics and mention the constraints for a combinatorial auction algorithm.**

A pricing and allocation algorithm partitions the set of users into two disjoint sets, winners and losers, denoted as W and L, respectively. The algorithm should:

- ➔ Be computationally tractable. Traditional combinatorial auction algorithms such as Vickey-ClarkeGroves (VLG) fail this criteria, because they are not computationally tractable.
- ➔ Scale well. Given the scale of the system and the number of requests for service, scalability is a necessary condition.
- ➔ Be objective. Partitioning in winners and losers should only be based on the price πu of a user's bid. If the price exceeds the threshold, the user is a winner; otherwise the user is a loser.
- ➔ Be fair. Make sure that the prices are uniform. All winners within a given resource pool pay the same price.
- ➔ Indicate clearly at the end of the auction the unit prices for each resource pool.
- ➔ Indicate clearly to all participants the relationship between the supply and the demand in the system.

*Refer 47*

**47) Explain and write the psuedocode of the ASCA combinatorial Auction algorithm. Say how the excess vector is computed and the proxies can be modeled. List the constraints for a combinatorial algorithm.**

Informally, in the ASCA algorithm the participants at the auction specify the resource and the quantities of that resource offered or desired at the price listed for that time slot. Then the ***excess vector.***

$$z(t) = \sum_u x_u(t)$$

There is a slight complication as the algorithm involves user bidding in multiple rounds. To address this problem the user proxies automatically adjust their demands on behalf of the actual bidders, as shown in Figure 6.6. **These proxies can be modeled** as functions that compute the “best bundle” from each Qu set given the current price.

$$Q_u = \begin{cases} \hat{q}_u & \text{if } \hat{q}_u^T p \leqslant \pi_u \quad \text{with} \quad \hat{q}_u \in \arg\min (q_u^T p) \\ 0 & \text{otherwise} \end{cases} .$$

The input to the ASCA algorithm: U users, R resources, $\bar{p}$ the starting price, and the update increment function, $g : (x, p) \rightarrow R^R$

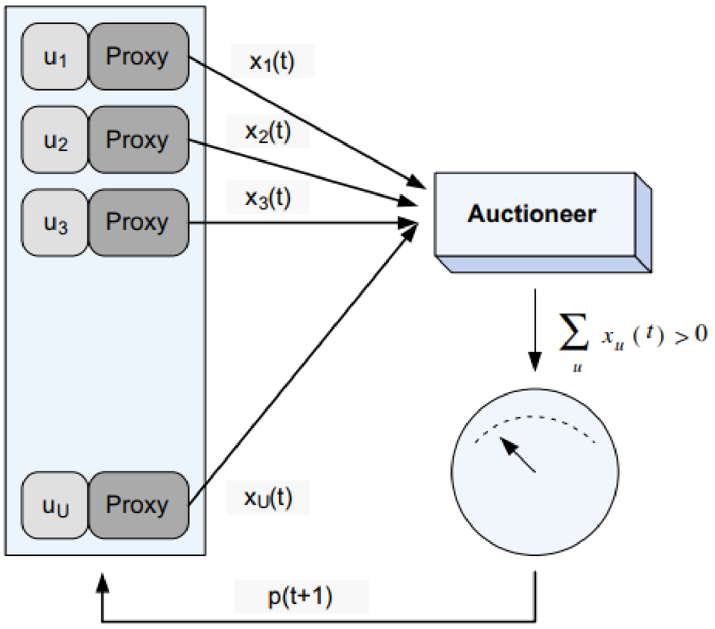


FIGURE 6.6

The schematics of the ASCA algorithm. To allow for a single round, auction users are represented by proxies that place the bids $x_u(t)$. The auctioneer determines whether there is an excess demand and, in that case, raises the price of resources for which the demand exceeds the supply and requests new bids.

**Table 6.4** The constraints for a combinatorial auction algorithm.

| | |
|---|---|
| $x_u \in \{0 \cup \mathcal{Q}_u\},\ \forall u$ | A user gets all resources or nothing. |
| $\sum_u x_u \leqslant 0$ | Final allocation leads to a net surplus of resources. |
| $\pi_u \geqslant (x_u)^T p,\ \forall u \in \mathcal{W}$ | Auction winners are willing to pay the final price. |
| $(x_u)^T p = \min_{q \in \mathcal{Q}_u} (q^T p),\ \forall u \in \mathcal{W}$ | Winners get the cheapest bundle in $\mathcal{I}$. |
| $\pi_u < \min_{q \in \mathcal{Q}_u} (q^T p),\ \forall u \in \mathcal{L}$ | The bids of the losers are below the final price. |
| $p \geqslant 0$ | Prices must be nonnegative. |

The constraints in Table 6.4 correspond to our intuition: (a) the first one states that a user either gets one of the bundles it has opted for or nothing; no partial allocation is acceptable. (b) The second constraint expresses the fact that the system awards only available resources; only offered resources can be allocated. (c) The third constraint is that the bid of the winners exceeds the final price. (d) The fourth constraint states that the winners get the least expensive bundles in their indifference set. (e) The fifth constraint states that losers bid below the final price. (f) The last constraint states that all prices are positive numbers.

```
1: set $t = 0$, $p(0) = \bar{p}$
2: loop
3:    collect bids $x_u(t) = \mathcal{G}_u(p(t)),\ \forall u$
4:    calculate excess demand $z(t) = \sum_u x_u(t)$
5:    if z(t) <0 then
6:       break
7:       else

8:          update prices $p(t+1) = p(t) + g(x(t), p(t))$
9:          $t \leftarrow t + 1$
10:   end if
11: end loop
```

**48) Write the expression and condition that has to be satisfied for max-min criterion for fair allocation and CPU scheduling.**

according to the max-min criterion, the following conditions must be satisfied by a fair allocation:

➔ C1. The amount received by any user is not larger than the amount requested, Bi bi .

➔ C2. If the minimum allocation of any user is Bmin no allocation satisfying condition C1 has a higher Bmin than the current allocation.

➔ C3. When we remove the user receiving the minimum allocation Bmin and then reduce the total amount of the resource available from B to (B − Bmin), the condition C2 remains recursively true.

A fairness criterion for CPU scheduling [142] requires that the amount of work in the time interval from t1 to t2 of two runnable threads a and b, a(t1, t2) and b(t1, t2), respectively, minimize the expression where wa and wb are the weights of the threads a and b, respectively.

$$\left| \frac{\Omega_a(t_1, t_2)}{w_a} - \frac{\Omega_b(t_1, t_2)}{w_b} \right|,$$

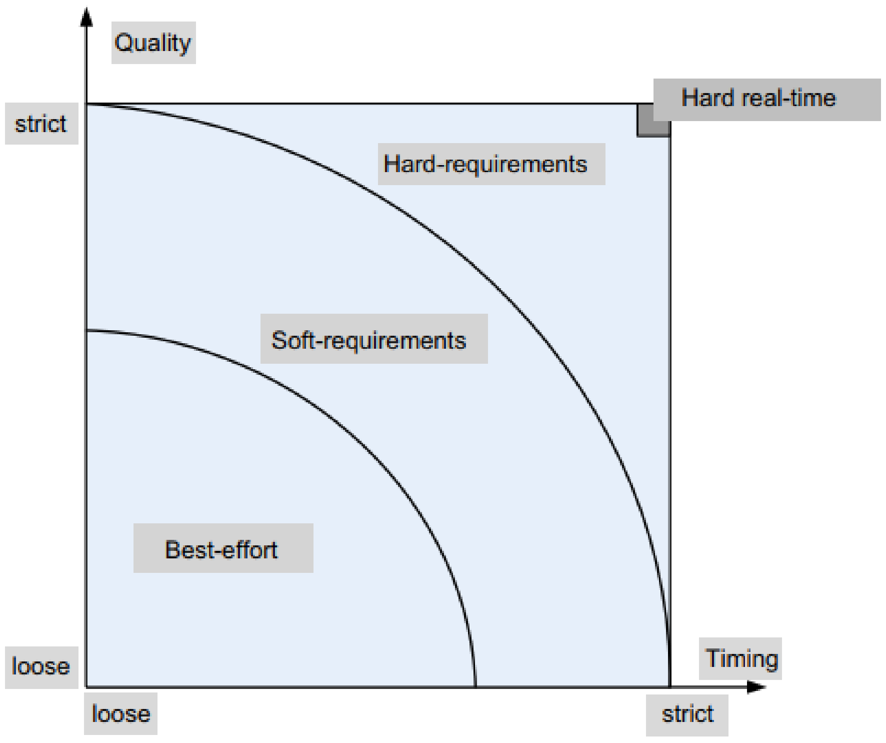


**FIGURE 6.7**

Best-effort policies do not impose requirements regarding either the amount of resources allocated to an application or the timing when an application is scheduled. Soft-requirements allocation policies require statistically guaranteed amounts and timing constraints; hard-requirements allocation policies demand strict timing and precise amounts of resources.

**49) What are the computing resources in demand that requires management and mention the criteria that affect evaluation of system? What does scaling, its type and elasticity has to do with it. Explain.**

The demand for computing resources, such as CPU cycles, primary and secondary storage, and network bandwidth, depends heavily on the volume of data processed by an application.

The demand for resources can be a function of the time of day, can monotonically increase or decrease in time, or can experience predictable or unpredictable peaks.

**two scaling modes:**

**vertical and horizontal.** Vertical scaling keeps the number of VMs of an application constant, but increases the amount of resources allocated to each one of them. This can be done either by migrating the VMs to more powerful servers or by keeping the VMs on the same servers but increasing their share of the CPU time. The first alternative involves additional overhead; the VM is stopped, a snapshot of it is taken, the file is transported to a more powerful server, and, finally, the VM is restated at the new site.

**Horizontal scaling** is the most common mode of scaling on a cloud; it is done by increasing the number of VMs as the load increases and reducing the number of VMs when the load decreases. Often, this leads to an increase in communication bandwidth consumed by the application. Load balancing among the running VMs is critical to this mode of operation. For a very large application, multiple load balancers may need to cooperate with one another. In some instances the load balancing is done by a front-end server that distributes incoming requests of a transaction-oriented system to back-end servers.

The **elasticity** of a public cloud, the fact that it can supply to an application precisely the amount of resources it needs and that users pay only for the resources they consume are serious incentives to migrate to a public cloud.

**50) Use the start-time fair queuing (SFQ) scheduling algorithm to compute the virtual start-up and the virtual finish time for two threads a and b with weights wa = 1 and wb = 5 when the time quantum is q = 15 and thread b blocks at time t = 24 and wakes up at time t = 60. Plot the virtual time of the scheduler function of the real time.**

*Refer 45*

**51) Examine cloud scheduling subject to deadlines, its task characterization, system model and the scheduling policies.**

**Task Characterization and Deadlines.** Real-time applications involve periodic or aperiodic tasks with deadlines. A task is characterized by a tuple $(A_i, \sigma_i, D_i)$, where $A_i$ is the arrival time, $\sigma_i > 0$ is the data size of the task, and $D_i$ is the *relative deadline*. Instances of a *periodic task*, $\Pi_i^q$, with period $q$ are identical, $\Pi_i^q \equiv \Pi^q$, and arrive at times $A_0, A_1, \dots A_i, \dots$, with $A_{i+1} - A_i = q$. The deadlines satisfy the constraint $D_i \leqslant A_{i+1}$ and generally the data size is the same, $\sigma_i = \sigma$. The individual instances of *aperiodic tasks*, $\Pi_i$, are different. Their arrival times $A_i$ are generally uncorrelated, and the amount of data $\sigma_i$ is different for different instances. The *absolute deadline* for the aperiodic task $\Pi_i$ is $(A_i + D_i)$.

We distinguish *hard deadlines* from *soft deadlines*. In the first case, if the task is not completed by the deadline, other tasks that depend on it may be affected and there are penalties; a hard deadline is strict and expressed precisely as milliseconds or possibly seconds. Soft deadlines play more of a guideline role and, in general, there are no penalties. Soft deadlines can be missed by fractions of the units used to express them, e.g., minutes if the deadline is expressed in hours, or hours if the deadlines is expressed in days. The scheduling of tasks on a cloud is generally subject to soft deadlines, though occasionally applications with hard deadlines may be encountered.

**System Model.** In our discussion we consider only aperiodic tasks with arbitrarily divisible workloads. The application runs on a partition of a cloud, a virtual cloud with a *head node* called $S_0$ and *n worker nodes* $S_1, S_2, \dots, S_n$. The system is homogeneous, all workers are identical, and the communication time from the head node to any worker node is the same. The head node distributes the workload to worker nodes, and this distribution is done sequentially. In this context there are two important problems:

1. The order of execution of the tasks $\Pi_i$.
2. The workload partitioning and the task mapping to worker nodes.

**Scheduling Policies.** The most common scheduling policies used to determine the order of execution of the tasks are:

- First in, first out (FIFO). The tasks are scheduled for execution in the order of their arrival.
- Earliest deadline first (EDF). The task with the earliest deadline is scheduled first.
- Maximum workload derivative first (MWF).

The *workload derivative* $DC_i(n^{min})$ of a task $\Pi_i$ when $n^{min}$ nodes are assigned to the application, is defined as

$$DC_i(n^{min}) = W_i\left(n_i^{min} + 1\right) - W_i\left(n_i^{min}\right), \qquad (6.69)$$

with $W_i(n)$ the workload allocated to task $\Pi_i$ when $n$ nodes of the cloud are available; if $\mathcal{E}(\sigma_i, n)$ is the execution time of the task, then $W_i(n) = n \times \mathcal{E}(\sigma_i, n)$. The MWF policy requires that:

1. The tasks are scheduled in the order of their derivatives, the one with the highest derivative $DC_i$ first.
2. The number $n$ of nodes assigned to the application is kept to a minimum, $n_i^{min}$.

**52) d**