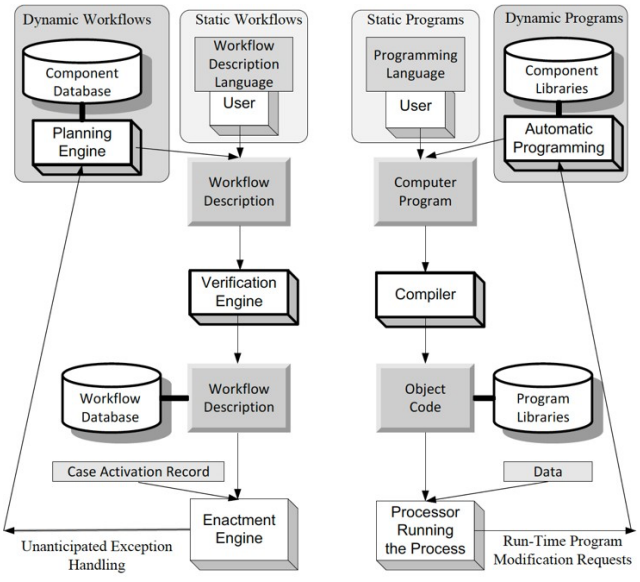
CCA Module 2 QB Solutions

1. Explain the various challenges of Cloud Computing.
   - Performance isolation - nearly impossible to reach in a real system, especially when the system is heavily loaded.
   - Reliability - major concern; server failures expected when a large number of servers cooperate for the computations.
   - Cloud infrastructure exhibits latency and bandwidth fluctuations which affect the application performance.
   - Performance considerations limit the amount of data logging; the ability to identify the source of unexpected results and errors is helped by frequent logging.

2. Describe the architectural styles for cloud computing.
   - Based on the client-server paradigm.
   - Stateless servers - view a client request as an independent transaction and respond to it; the client is not required to first establish a connection to the server.
   - Often clients and servers communicate using Remote Procedure Calls (RPCs).
   - Simple Object Access Protocol (SOAP) - application protocol for web applications; message format based on the XML. Uses TCP or UDP transport protocols.
   - Representational State Transfer (REST) - software architecture for distributed hypermedia systems. Supports client communication with stateless servers, it is platform independent, language independent, supports data caching, and can be used in the presence of firewalls.

3. Explain Workflows coordination of multiple activities.
   - Process description - structure describing the tasks to be executed and the order of their execution. Resembles a flowchart.
   - Case - an instance of a process description.
   - State of a case at time t - defined in terms of tasks already completed at that time.
   - Events - cause transitions between states.
   - The life cycle of a workflow - creation, definition, verification, and enactment; similar to the life cycle of a traditional program (creation, compilation, and execution).



4. Define the following
   a. State of a case t
   b. Primitive Task
   c. Liveliness
   d. Routing Task

   a. The state of a case at time t is defined in terms of tasks already completed at that time.
   b. A primitive task is one that cannot be decomposed into simpler tasks.
   c. The algorithm must use clocks to ensure liveliness and to overcome the impossibility of reaching consensus with a single faulty process.
   d. A routing task is a special-purpose task connecting two tasks in a workflow description.

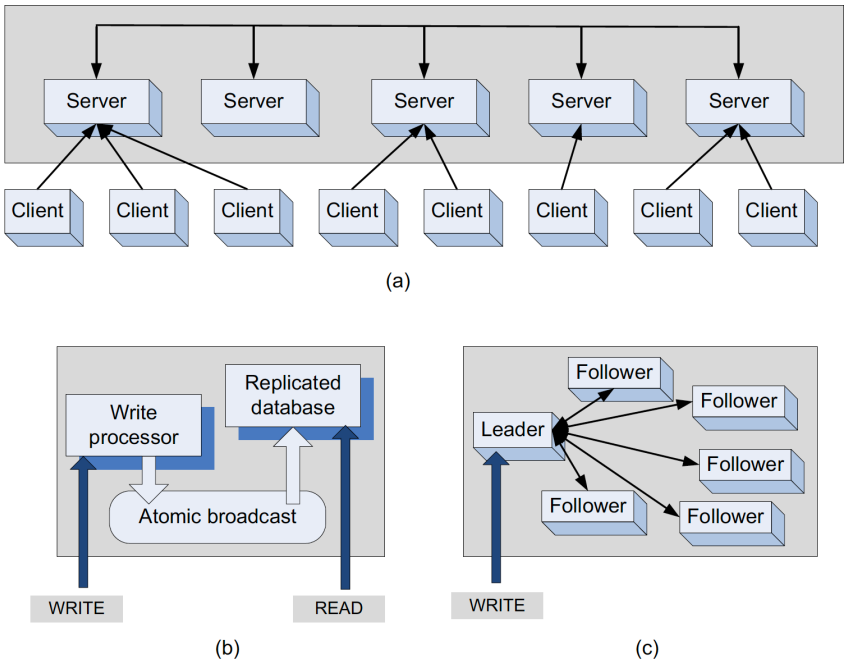
5. Describe the various attributes of a task.
   Task is the central concept in workflow modeling; a task is a unit of work to be performed on the cloud, and it is characterized by several attributes, such as:
   - Name. A string of characters uniquely identifying the task.
   - Description. A natural language description of the task.
   - Actions. Modifications of the environment caused by the execution of the task.
   - Preconditions. Boolean expressions that must be true before the action(s) of the task can take place.
   - Post-conditions. Boolean expressions that must be true after the action(s) of the task take place.
   - Attributes. Provide indications of the type and quantity of resources necessary for the execution of the task, the actors in charge of the tasks, the security requirements, whether the task is reversible, and other task characteristics.
   - Exceptions. Provide information on how to handle abnormal events. The exceptions supported by a task consist of a list of <event, action> pairs. The exceptions included in the task exception list are called anticipated exceptions, as opposed to unanticipated exceptions. Events not included in the exception list trigger replanning. Replanning means restructuring of a process or redefinition of the relationship among various tasks.

6. Define with a neat diagram the workflows and program.

Same answer as Q3

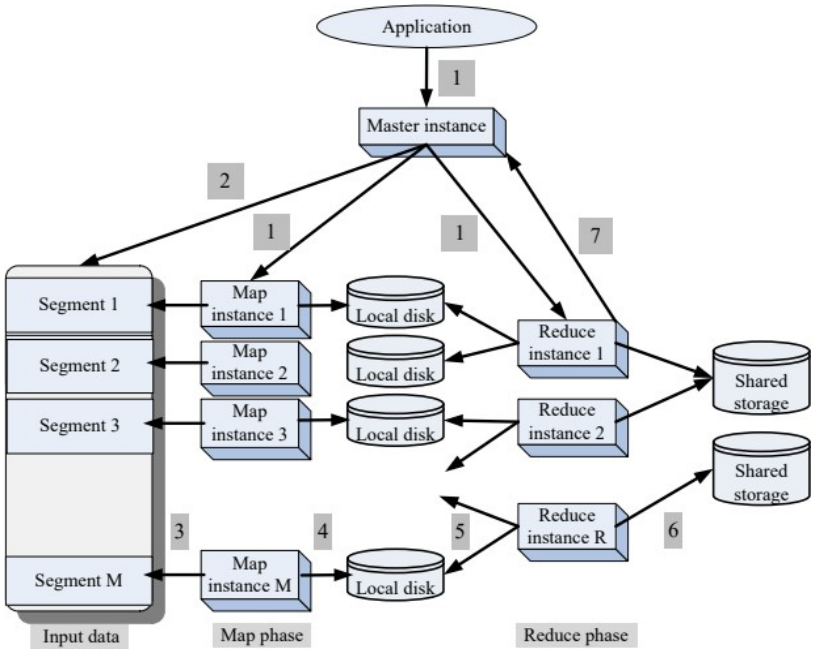
7. Explain coordination based on state machine model with a diagram.
   Coordination based on a state machine model: The ZooKeeper
   - Cloud elasticity distribute computations and data across multiple systems; coordination among these systems is a critical function in a distributed environment.
   - ZooKeeper:
     - Distributed coordination service for large-scale distributed systems.
     - High throughput and low latency service.
     - Implements a version of the Paxos consensus algorithm.
     - Open-source software written in Java with bindings for Java and C.
     - The servers in the pack communicate and elect a leader.
     - A database is replicated on each server; consistency of the replicas is maintained.

- A client connect to a single server, synchronizes its clock with the server, and sends requests, receives responses and watch events through a TCP connection.



The ZooKeeper coordination service. (a) The service provides a single system image. Clients can connect to any server in the pack. (b) Functional model of the ZooKeeper service. The replicated database is accessed directly by read commands; write commands involve more intricate processing based on atomic broadcast. (c) Processing a write command: (1) A server receiving the command from a client forwards the command to the leader; (2) the leader uses atomic broadcast to reach consensus among all followers.

8. Explain MapReduce programming model.



MapReduce philosophy

- An application starts a master instance, M worker instances for the Map phase and later R worker instances for the Reduce phase.
- The master instance partitions the input data in M segments.
- Each map instance reads its input data segment and processes the data.
- The results of the processing are stored on the local disks of the servers where the map instances run.
- When all map instances have finished processing their data, the R reduce instances read the results of the first phase and merge the partial results.
- The final results are written by the reduce instances to a shared storage server.
- The master instance monitors the reduce instances and when all of them report task completion the application is terminated.

9. Case study: Grep the web application

Case study: GrepTheWeb

The application illustrates the means to

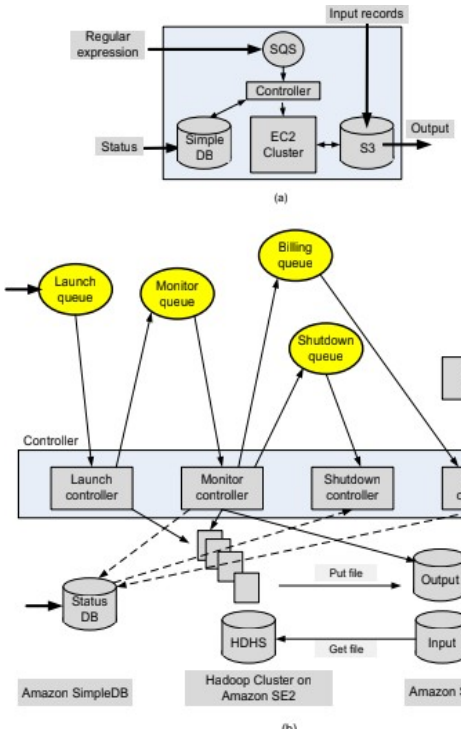
- create an on-demand infrastructure.
- run it on a massively distributed system in a manner that allows it to run in parallel and scale up and down, based on the number of users and the problem size.

GrepTheWeb

- Performs a search of a very large set of records to identify records that satisfy a regular expression.
- It is analogous to the Unix grep command.
- The source is a collection of document URLs produced by the Alexa Web Search, a software system that crawls the web every night.
- Uses message passing to trigger the activities of multiple controller threads which launch the application, initiate processing, shutdown the system, and create billing records.

(a) The simplified workflow showing the inputs:
- the regular expression.
- the input records generated by the web crawler.
- 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.



10. Explain the importance of layering with a neat diagram
    - 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.
    - Thus, we are able to design, implement, and modify the individual subsystems independently.
    - The instruction set architecture (ISA) defines a processor's set of instructions.
    - The hardware supports two execution modes,
        - a privileged/ kernel,
        - mode a user mode.
    - The instruction set consists of two sets of instructions, privileged 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.

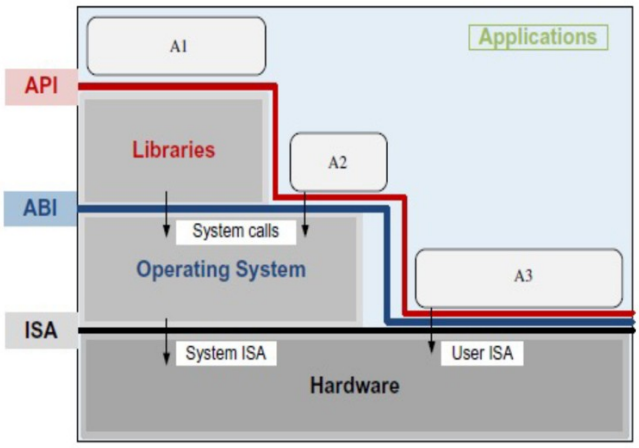


FIGURE 5.1

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).

11. Describe virtualization. Explain the types of virtualization.
Virtualization abstracts the underlying resources and simplifies their use, isolates users from one another, and supports replication, which, in turn, increases the elasticity of the system.
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.

There are side effects of virtualization, notably the:
    - performance penalty
    - the hardware costs.

Types of Virtualization
    - 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: VMW are ESX, ESXi Servers, Xen, OS370, and Denali.
    - Hybrid. The VMM shares the hardware with the existing OS [see Figure 5.3(c)]. Example: VMW are 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. A price to pay for this simplicity is the increased overhead and associated performance penalty; indeed, the I/O operations, page faults, and scheduling requests from a guest OS are not handled directly by the VMM. Instead, they are passed to the host OS. Performance as well as the challenges to support complete isolation of VMs make this solution less attractive for servers in a cloud computing environment. Example: User-mode Linux.

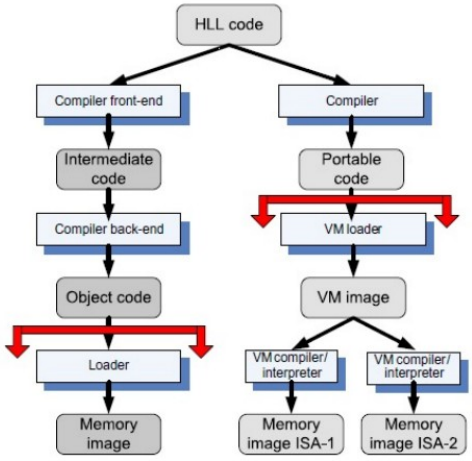
12. What is a virtual machine? List the different types of virtual machines along with an example for each.
A virtual machine (VM) is an isolated environment that appears to be a whole computer but actually only has access to a portion of the computer resources. Each VM appears to be running on the bare hardware, giving the appearance of multiple instances of the same computer, though all are supported by a single physical system.
Two types of VM: process and system
    - 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]).

13. What is virtual machine monitor? Briefly explain the various tasks done by VMM.
    - 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.
    - Sometimes the hardware supports a third mode of execution for the guest OS. [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. At the same time, the VMM monitors system performance and takes corrective action to avoid performance degradation.]

- 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.
- A VMM virtualizes the CPU and memory.
- 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, 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.
- VMMs use a range of optimization techniques; for example, VMware systems avoid page duplication among different virtual machines; they maintain only one copy of a shared page and use copy-on-write policies whereas Xen imposes total isolation of the VM and does not allow page sharing.
- VMMs control the virtual memory management and decide what pages to swap out; for example, when the ESX VMware server wants to swap out pages, it uses a balloon process inside a guest OS and requests it to allocate more pages to itself, thus swapping out pages of some of the processes running under that VM.
- Then it forces the balloon process to relinquish control of the free page frames.

14. With a help of a neat diagram explain the High-level language (HLL) code can be translated for a specific architecture and operating system.
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.
platform-specific compiler
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 binaries resulting from the translation of HLL programs are targeted to a specific hardware architecture.



appllication and library modules issur system calls to os, that decides whether the priviledged operations required by the application don't break system safety rules