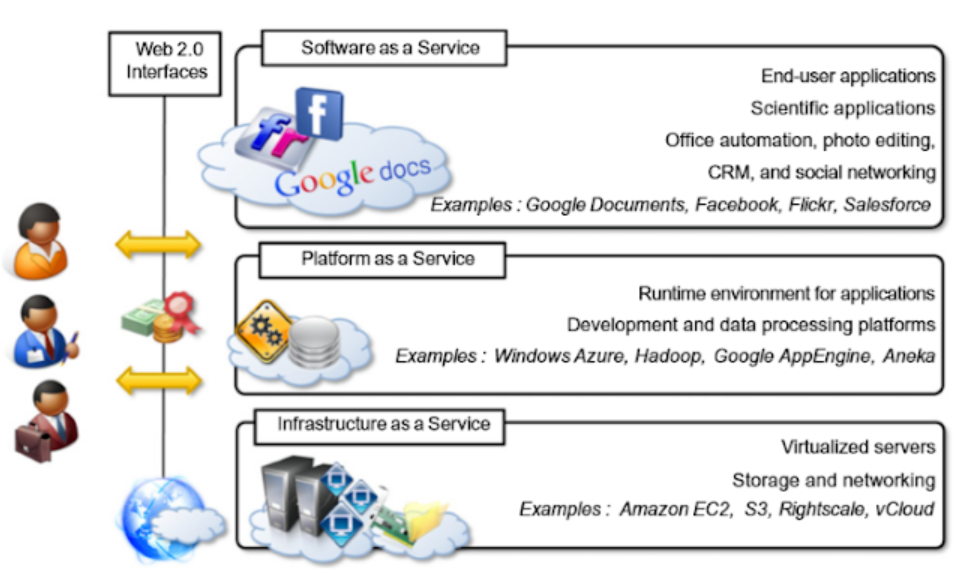
**Q. 1 Reference Model**

****

Define All IaaS, SaaS, PaaS

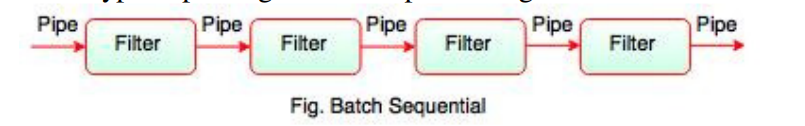
**Q. Parallel vs. distributed computing**

1. **Definition**:
   * Parallel computing involves breaking a problem into smaller tasks and processing them simultaneously on multiple processors within a single machine.
   * Distributed computing involves breaking a problem into smaller tasks and processing them on multiple computers connected via a network.
2. **Hardware**:
   * Parallel computing typically utilizes multiple processors within a single machine.
   * Distributed computing uses a network of different computers.
3. **Communication**:
   * In parallel computing, communication between processors is fast.
   * In Distributed computing relies on message passing and network communication, which can introduce latency.
4. **Scalability**:
   * Parallel computing is limited number of processors available on a single machine.
   * Distributed computing can scale horizontally by adding more machines to the network, making it more suitable for handling larger workloads.
5. **Fault Tolerance**:
   * In parallel computing, a failure in one processor can disrupt the entire process.
   * Distributed computing is more fault-tolerant because if one machine fails, others can continue processing.
6. **Examples**:
   * Parallel computing is used in high-performance computing for scientific simulations.
   * Distributed computing is used in web servers, cloud computing, and big data processing.
7. **Programming Model**:
   * Parallel computing often uses shared-memory models like OpenMP.
   * Distributed computing relies on message-passing models like MPI or higher-level abstractions like MapReduce.
8. **Latency**:
   * Parallel computing has lower latency since processors communicate directly through shared memory.
   * Distributed computing can suffer from higher latency due to network communication.
9. **Data Storage**:
   * Parallel computing may use a single, shared storage system.
   * Distributed computing use distributed databases and storage systems.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Parallel computing** | **Distributed computing** |
| **Number of computers** | **Multiple processors on a single computer, or a cluster of computers** | **Multiple computers connected over a network** |
| **Memory** | **Shared memory** | **Each computer has its own memory** |
| **Communication** | **Processors communicate with each other through shared memory, or through message passing** | **Computers communicate with each other through message passing** |
| **Tasks** | **Tasks are divided into smaller tasks that can be executed simultaneously** | **Tasks are divided into smaller tasks that are executed by different computers** |
| **Performance** | **Can significantly improve the performance of a task** | **Can be used to solve problems that are too large or complex to be solved by a single computer** |
| **Scalability** | **Can be scaled up by adding more processors or computers** | **Can be scaled up by adding more computers** |

**Q. 2 Batch Sequential**

1. Batch sequential compilation in 1970 was considered to be a sequence process.
2. In Batch sequential, Separate program systems are run sequentially and the data is transferred from one program to the next as an aggregation.
3. This is a typical paradigm for data processing.



1. The diagram above shows the batch sequential architecture flow.
2. It offers simpler sub-system divisions and each subsystem can be an

independent program which works on input and produces output data.

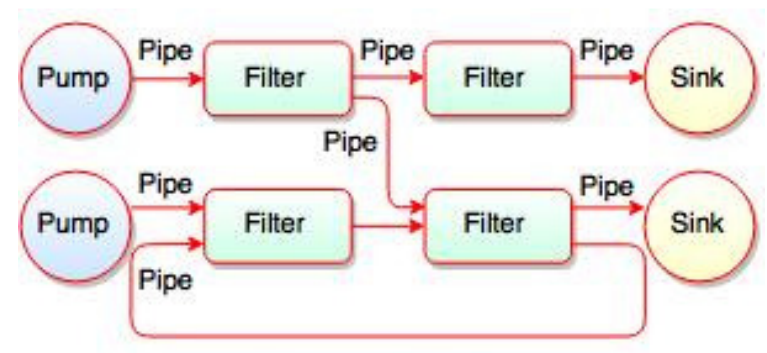
1. The biggest downside of the sequential batch architectures is the lack

of a concurrency and interactive interface. It provides high latency and

low throughput.

**Q.3 Pipe and Filter:**

1. Pipe is a connector that transfers data from one filter to another filter
2. Pipe is a directional data stream which a data buffer implements to store all data, before the following filter has time to process it.
3. It moves the data from one data source to a one data sink.
4. The stateless data stream is pipes.



1. The figure above shows the sequence of the pipe filter.
2. All filters are the processes running concurrently, which means they can run as separate threads or coroutines or be fully located on various machines.
3. Every pipe has a filter connection and has its own role in filter's operation.
4. The filters are robust, with the addition and removal of pipes on runtime.
5. Filter reads the data from their input pipes, performs its function on these data and places the result on all output pipes.
6. If the input pipes are not enough data, the filter only waits for them.
7. Filter:
8. Filter is a component.
9. The interfaces are used to flow in a variety of inputs and to flow out a
10. variety of outputs.
11. It processes and refines the data input.
12. The independent entities are filters.
13. Two ways to create a filter exist:

1. Active Filter

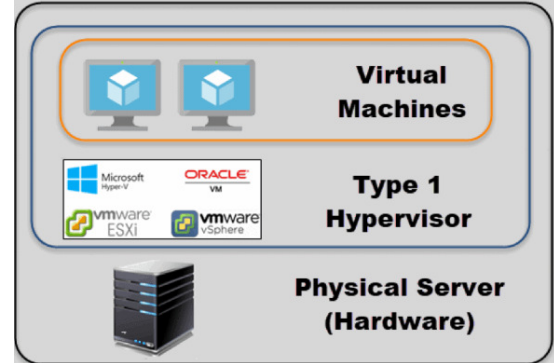
2. Passive Filter

1. The active filter creates the pipes' data flow.
2. Data flow on the pipes is driven by the passive filter.
3. Filter does not share state with other filters.
4. The identity of upstream and downstream filters is unclear.
5. Separate threads are used for filters. It may be threads or coroutines of

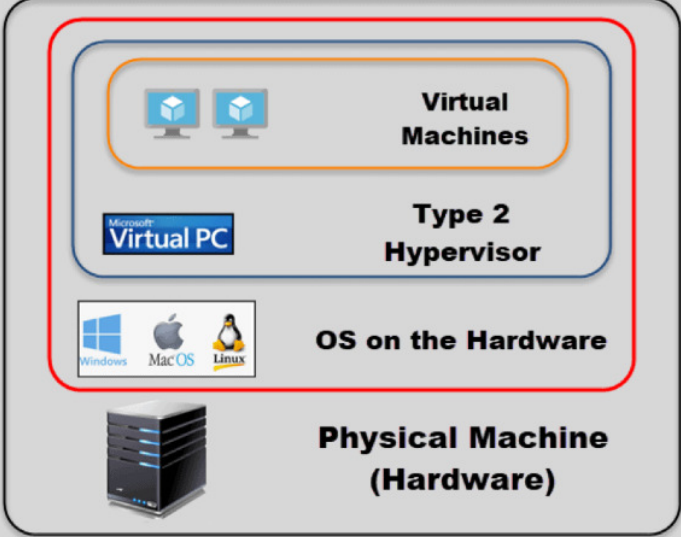
hardware or software

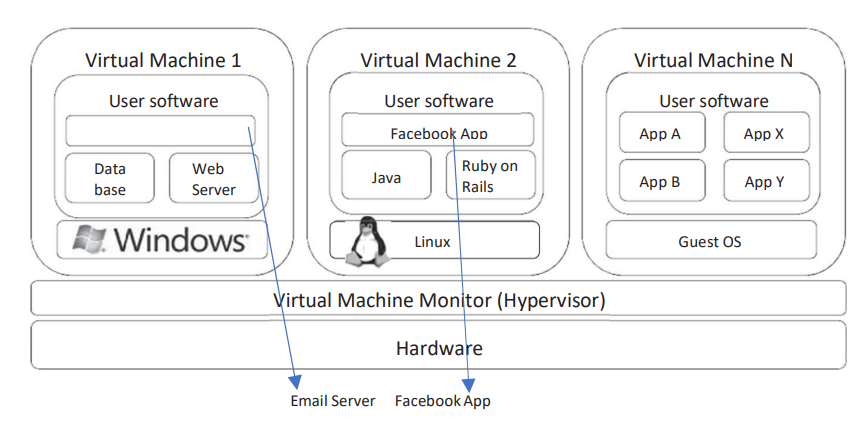
**Q.4 Types Of Hypervisor**

1. **Type 1 Hypervisor (Bare-Metal Hypervisor)**:



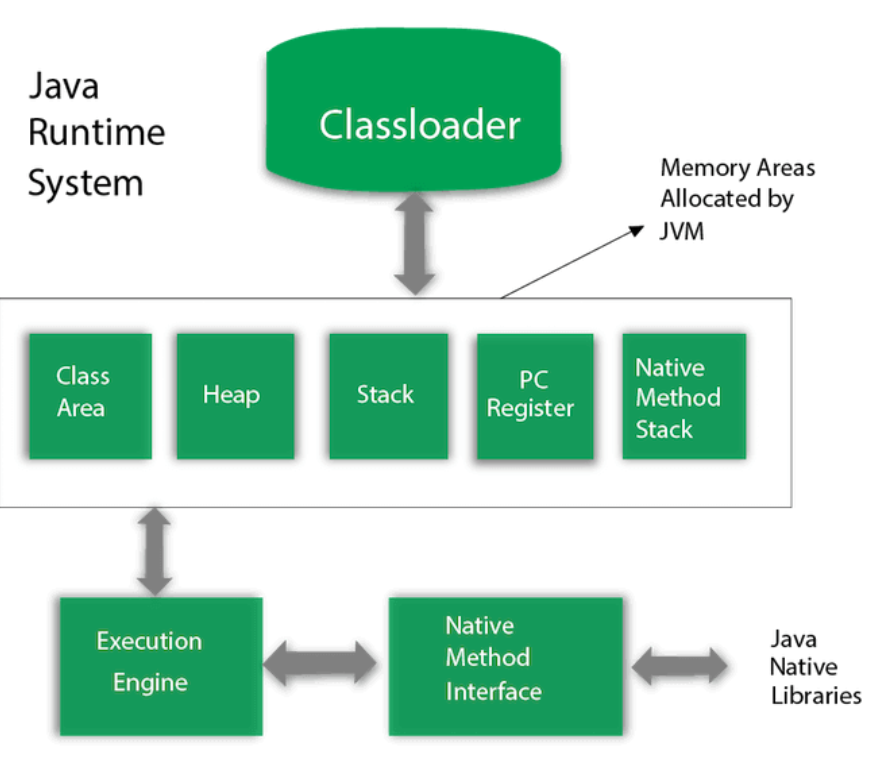
* + This hypervisor runs directly on the physical hardware, without the need for a host OS.
  + It is highly efficient and offers better performance because it has direct access to physical resources.
  + Examples include VMware vSphere/ESXi, Microsoft Hyper-V (when installed without Windows Server), and Xen.

1. **Type 2 Hypervisor (Hosted Hypervisor)**:  
    
   * Type 2 hypervisors run on top of an existing host operating system.
   * They are easier to set up and use for testing and development.
   * Performance may be slightly lower compared to Type 1 due to the additional layer.
   * Examples include VMware Workstation, Oracle VirtualBox, and Parallels Desktop.

**Q.5 Hardware Virtualization:**

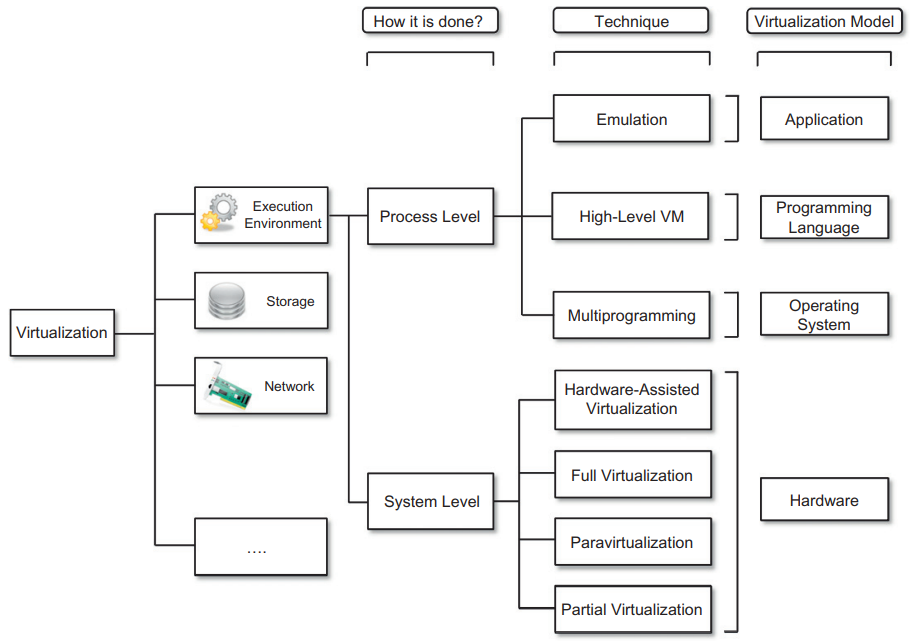
1. **Isolation**: Hardware virtualization provides strong isolation between virtual machines. Each VM operates as an independent entity with its own dedicated resources, including CPU, memory, and storage.
2. **Efficiency**: It maximizes resource utilization by allowing multiple VMs to share the same physical hardware while maintaining isolation. This enhances the efficiency of server utilization.
3. **Hypervisor**: A hypervisor, which can be either a Type 1 (bare-metal) or Type 2 (hosted) hypervisor, is used to manage and allocate physical resources to VMs.
4. **Compatibility**: Hardware virtualization is compatible with a wide range of operating systems, making it versatile for various workloads and applications.
5. **Security**: It enhances security by isolating VMs from each other, reducing the risk of one VM affecting others in case of a security breach.
6. **Migration**: Live migration of VMs is possible, allowing them to be moved between physical servers without significant downtime.
7. **Cloud Computing**: Hardware virtualization is fundamental to cloud computing, enabling cloud providers to allocate virtual resources to customers while maintaining physical separation.

**Q6. JVM Architecture:**

****

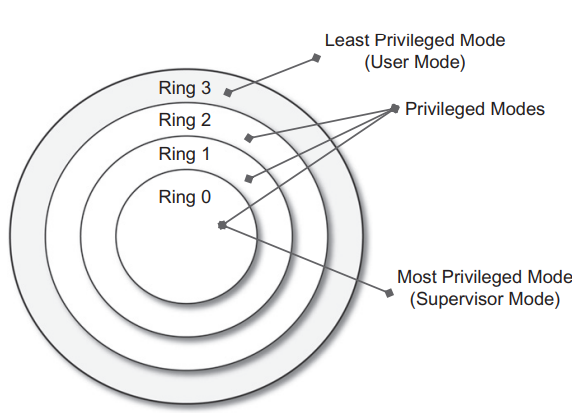
1. **Class Loader Subsystem**:
   * Responsible for loading class files into memory.
   * Class files contain bytecodes generated from Java source code.
2. **Method Area**:
   * Stores class-level data, including method code, field data, and static variables.
3. **Heap**:
   * The JVM heap is where objects are allocated and deallocated during runtime.
   * It's divided into the Young Generation and Old Generation, managing object lifecycle efficiently.
4. **Java Stacks**:
   * Each thread running in the JVM has its own Java Stack.
   * It stores method call frames and local variables.
5. **Native Method Stacks**:
   * Similar to Java Stacks but used for native methods (methods written in languages like C/C++).
6. **Execution Engine**:
   * Interprets bytecode or compiles it into native machine code.
   * Includes the Just-In-Time (JIT) compiler for performance optimization.
7. **Native Interface**:
   * Allows interaction between Java code and native libraries through the Java Native Interface (JNI).
8. **Native Method Libraries**:
   * Contains native libraries required for Java applications.
9. **Direct Memory**:
   * Memory outside the JVM heap used for Direct Byte Buffers.
10. **PC Registers**:
    * Each thread has its Program Counter (PC) register, storing the address of the currently executing instruction.
11. **Native Method Interface**:
    * Enables interaction with native libraries using Java code.

**Q.7 Taxonomy of virtualization techniques.**

****

1. **Hardware Virtualization**:
   * **Full Virtualization**: It allows multiple virtual machines to run different operating systems on the same physical hardware. Hypervisors, like VMware ESXi, enable this.
   * **Para-virtualization**: Here, the guest OS is aware of the virtualization, which can improve performance. Xen is an example.
2. **Operating System Virtualization**:
   * **Containers**: They share the host OS kernel but isolate user spaces, making them lightweight and efficient. Docker and Kubernetes are popular container platforms.
3. **Storage Virtualization**:
   * **Storage Area Network (SAN) Virtualization**: It abstracts multiple storage devices into a single pool, simplifying management and enhancing scalability.
   * **Network Attached Storage (NAS) Virtualization**: Similar to SAN, but focused on file-level access.
4. **Network Virtualization**:
   * **Software-Defined Networking (SDN)**: Decouples network control and data planes, offering flexibility in network management and automation.
   * **Virtual LANs (VLANs)**: Segments a physical network into multiple virtual networks.
5. **Application Virtualization**:
   * **Application Streaming**: It delivers applications on-demand to end-user devices, reducing installation and compatibility issues.
   * **Application Containers**: Isolates applications and their dependencies for portability.
6. **Cloud Computing Virtualization**:
   * **Infrastructure as a Service (IaaS)**: Virtualizes computing resources like servers and storage in cloud environments.
   * **Platform as a Service (PaaS)**: Provides a platform for developing and deploying applications without managing underlying infrastructure.

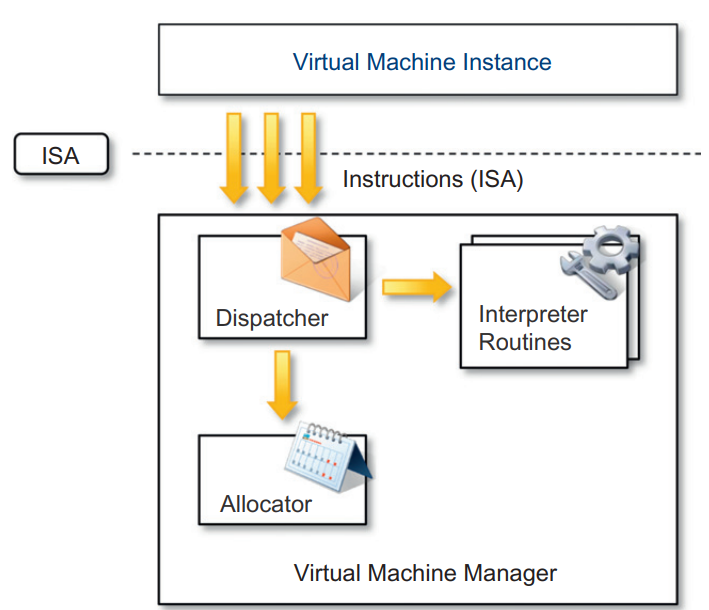
**Q.8 Security rings and privilege modes.**

****

Security rings and privilege modes are concepts used in computer systems to manage and control access to hardware and software resources. They are often implemented in modern processors and operating systems to ensure system stability and security.

1. **Protection Rings**:
   * Protection rings, also known as privilege levels or CPU privilege rings, are hierarchical levels of privilege in a computer system. Typically, there are four rings, numbered from 0 (most privileged) to 3 (least privileged).
   * Ring 0 is the highest privilege level and is often reserved for the operating system kernel. It has unrestricted access to hardware and can execute privileged instructions.
   * Rings 1 and 2 are less privileged and are not commonly used in modern systems.
   * Ring 3 is the least privileged and is where user-level applications run. It has limited access to hardware and cannot execute certain privileged instructions.
2. **Purpose**:
   * The primary purpose of protection rings is to prevent unauthorized access and ensure the stability of the system. Ring 0 can execute critical operations while isolating user-level processes in Ring 3.
   * If a process in Ring 3 attempts to perform a privileged operation, the hardware generates an exception, and the operating system can decide how to handle it, often terminating the offending process.

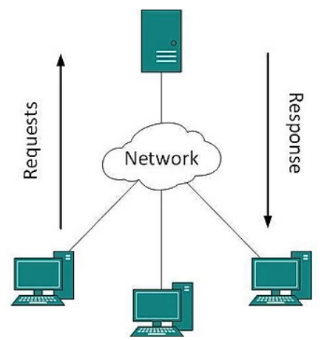
**Q.9 Hypervisor reference architecture.**

****

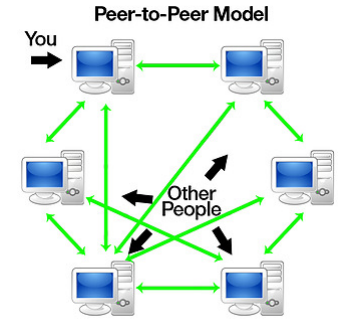
1. The dispatcher is the entry point of the monitor and redirects instructions from the VM instance to one of the other two modules.
2. The allocator is responsible for allocating resources to VMs. The dispatcher calls the allocator when a VM needs more resources.
3. The interpreter module executes interpreter routines whenever a VM executes a privileged instruction.

**Q.10 System architectural styles:**

Client-server and peer-to-peer are two common network architectures. They are often confused, but there are key differences between them.

****

1. **Client Server Architecture:**
2. Two major components are in the client server architecture.
3. The server and the client.
4. The server stores and processes data, while the client accesses it.
5. The server allows clients to make requests, and the server will reply.
6. In general, the remote side is managed only by a computer.
7. Load balancing distributes traffic across multiple servers for better performance and reliability.
8. This database includes information on security, such as credentials and access details.
9. Absent security keys, users can't sign in to a server.
10. This architecture bit more stable and secure than Peer to Peer.
11. A secure database can improve resource efficiency, which leads to stability
12. A server can only handle a limited amount of work at a time, so if too much work is requested, the system could crash.
13. **Peer to Peer (P2P):**



1. There is no central control in a distributed system behind peer topeer.
2. The fundamental idea is that at a certain time each node can be a client or a server.
3. If something is asked from the node, it could be referred
4. to as a client and if something arrives from a node it could be referred to as a server.
5. Usually every node is called a peer.
6. Any new node will first join this network.
7. Upon joining, they may either request or provide a service.
8. A node's initiation phase (joining a node) can vary based on network's implementation.
9. There are two ways a new node can learn what other nodes provide.

