Quiz 1

1. **Features of cloud computing:**
   Cloud computing is on-demand availability of computer system resources like computing power, storage, etc without having to personally deal with physical hardware. This is made possible through massive data centers that hosts thousands of computers to run user specified tasks. There are many features of cloud computing. Some of them are listed below:
   - **On demand self-service:** This is one of the vital characteristics that makes cloud computing so appealing. Anyone can literally spin up and down a server or computing power, or storage, or network bandwidth at any given time of the day or night without having to buy entire sets of hardware for it. With this service user can also monitor the computing capabilities and scale the server up and down as per the demand of the application deployed in the cloud.
   - **Broad Network Access:** Cloud computing resources can be assessed through different platforms and devices over network, be it a private network as in private clouds or worldwide network like internet. Using cloud, we can upload the data to the cloud from anywhere and we can access it with any device if we have internet connection.
   - **Resource Pooling:** Since cloud provides service to so many clients at the same time, resource pooling helps them give service to as many users as possible in cheapest price possible. Cloud providers acts like a manager who can provide access to its pooled computing resources using multi-tenant model. Multi-tenancy allows multiple customers to share common applications or physical infrastructure while retaining security and privacy over their data. On one hand resource pooling allows for the price of computing power to be much cheaper while also enabling cloud providers to become much more efficient as to not waste or create more computing than necessary.
   - **Rapid elasticity or Scalibility:** If one was to set up his own actual server to run a web server, he would have to buy actual physical hardwares. When his server hits very high load, he will have to buy more servers and also figure out tedious process to how to interconnect them so as to distribute the load. However, with cloud computing's rapid elasticity features, one can easily scale up or down instantaneously as the server hits higher/lower load. Just in time (JIT) service is the notion of requiring cloud elasticity either to provision more resources or less. This makes scaling automatic.
   - **Measured Service:** In the traditional approach of owning a server, it is really hard to keep track of the service and resources one is actually being able to leverage. However, in cloud computing each and every resource usage like: VMs, storage, processing and bandwidth can be monitored, controlled, and reported. Resource utilization can also be highly optimized by using charge-per-use capabilities.
   - **Security:** Until today, there have been no known breaches of the underlying resources of the major cloud platforms. Since big platforms has groups of security experts and engineers working to maintain the high level of security, it is always safer to use cloud to get secure services instead of trying to invent the wheels of security from scratch.

2. **Differences between IaaS, PaaS, and SaaS:**
    - **IaaS:** Typically, IaaS (Infrastructure as Service) includes cloud computing infrastructures like servers, network, operating systems, and storage. These are achieved through virtualization. IaaS provides the same technologies and capabilities like traditional data center without one having to manage or maintain if there is any problem. This is the hardest to setup than PaaS and SaaS, but has the most flexibility. Some examples of IaaS are: AWS EC2, Google Cloud Compute Engine, Linode, DigitalOcean, etc.
    - **PaaS:** PaaS (Platform as a Service) provides a platform for software creation. This platform is usually delivered through web making developers life lot easier since they don't have to worry about operating system, software updates, storage, or infrastructure. However, there is certain limitation to what developers can achieve using PaaS. Some of the limitations includes data security, integrations, vendor lock in, runtime issues, etc. Example of PaaS includes: AWS Elastic Beanstalk, Firebase, Heroku, Google App Engine, etc.
    - **SaaS:** SaaS (Software as a Service) are simply cloud application services. It uses internet to deliver the software to its user. Since the application can be used directly from web browser, using SaaS makes developer task easiest. However, there are lot more limitation to using SaaS as there is very few thing developers can tweak. Some of its limitations includes: interoperability, vendor lock in, lack of integration support, data security, customization, feature limitations, etc. Example of SaaS includes: Dropbox, Google Drive, Gmail, etc.

3. **PUE:**
PUE stands for Power Usage Effectiveness. It represents the ratio of total amount of energy used by datacenter facility to the energy delivered to the computers. It is an important industry metric that measures the energy efficiency of data center's infrastructure under normal operating conditions. It is calculated as:

```
PUE= (total energy entering data center)/(IT Equipment Power)
```

Less PUE means less energy is used for cooling or more portion of the total energy entering the data center is actually used to run IT equipment. In short, lower PUE for a data center indicates the higher power efficiency.
There are many factors that can affect PUE:
    - **Temperature:** Higher the temperature of the surrounding of datacenter, higher will be the PUE which means higher portion of the total energy entering data center will be required for air cooling and hence less portion of total energy is actually being used to run computers.

- **Newness of the facility:** Basically, newer the facility, there is more probability that more advanced technology is used which usually consumes less energy and even may require less cooling power.
- **Location of the data center:** Location of the data center affects the PUE of that datacenter. For example, a data center in New Delhi, where ambient temperature is relatively high will require more energy to cool the data center than similar datacenter located in US where temperature is relatively lower.
- **Data center's design:** Typically, data center has top charged and bottom charged. Hot air rises from bottom to top, so better PUE can be achieved by changing top charged equipment to bottom models. Another way would be to design and reposition racks in hot-aisle/cold-aisle approach.

4. **Over Subscription in the traditional data center network:**
   Traditional data center network uses switches of higher bandwidth at higher level in order to perform data transfer at reliable speed. However, it is not necessary for higher level switches bandwidth to be total sum of bandwidths supported by all servers together. Over subscription occurs when the overall internal switching bandwidth of the switches is less than the total bandwidth of all ingress ports. For instance: if we have a 48-port switch each supporting 10 GbE, we will have 480GbE required if all servers were to operate at full speed. If the ports in upper level does not support 480GbE, that is where we have oversubscription. If the internal bandwidths on the switch can only switch 240 GbE of traffic at any given time, then we have 2:1 oversubscription ratio. This is done mainly because of economic reasons and also because not every server will be operating at their full network speed simultaneously thereby taking advantage of network patterns. It was one of the reason why traditional data center was also able to add more devices to the network.

5. **How many servers in Fat-tree topology when k=64?**
   Fat Tree adopts a special instance of Clos topology. It is used in order to solve over subscription problem in traditional data center network. Not only does it solve problem like over subscription, but also brings down the expense, and is easier to scale and much fault tolerant.
   For k-ary fat tree, there are three layers: edge, agregation and core. There are $(k/2)^2$ core switches, k pods and each pods consists of $(k/2)^2$ servers. In this topology, each of the core switch connects to all the k pods.

```
When k = 64,
Number of Core Switches= (k/2)² = 4096
Number of Aggregation and Edge Switches = (k/2) = 36
Number of Servers per pod = (k/2)² = 4096
Total number of pods = k = 64

Hence,  Maximum number of servers = (k^3)/4 = 65536
```

6. **Difference between type 1 and type 2 hypervisor:**
A hypervisor or Virtual Machine Monitor (VMM) is a software that creates and runs virtual machines (VMs). With hypervisor, it is possible to support multiple guest VMs on one host computer by sharing the resource of the host computer. There are two kinds of hypervisor. They are:

Type 1/ Native/ Bare Metal	Type 2/ Hosted
This kind of hypervisor runs on top of the bare metal or hardware.	These hypervisors run on top of conventional operating system as a software layer or application.
This is more like an Operating System that can run another OS on top of it.	This is purely a software that emulates OS and works by abstracting guest operating systems from the host operating system.
It runs in privileged mode.	It runs in unprivileged mode.
This is more common in an enterprise data center or other server-based environment.	This is more common for individual users who want to run multiple operating systems on a single device.
It is much efficient and scalable.	It is not as scalable and efficient because of its reliance on underlying OS.
It is more secure to use.	It is less secure since any problem in the base OS could affect the entire hypervisor.
Example: KVM, VMware ESXI, Microsoft Hyper-V, etc.	Example: VMWare, Virtualbox, QEMU, Microsoft Virtual PC, etc.

7. **Why VMs cannot execute privileged instructions directly?**
VM runs on top of hypervisor, which is usually responsible for running multiple VMs. Any VM running on top of that hypervisor will be running on user mode since they are like a software or application. They cannot execute privileged instruction themselves in order to provide protection to kernel and CPU from any sort of malicious behavior that user of VM could execute which in turn could affect other VMs running on the same hypervisor. Attempting a privileged instruction in user mode by VM causes an error -> trap. That is where VMM gains control, analyzes error and executes the operation attempted by the guest if possible. Then the control and return value is sent back to user mode. Another reason why VM cannot run privileged instruction directly is because if they could, they can disable the

interrupts which will affect all other VMs that are supposed to be isolated from one another and therefore performance of all VMs will be affected.

8. **Hadoop program using map() and reduce() to calculate sum of cubes:**

```
package hadoop;

import java.util.*;

import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class sumOfCubes {
    // Mapper class
    public static class Map extends MapReduceBase implements Mapper<LongWritable, /* Input
key Type */
            Text, /* Input value Type */
            Text, /* Output key Type */
            IntWritable> /* Output value Type */
    {
        // Map function
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter)
                throws IOException {

            int num = parseInt(value);
            output.collect(new Text(key), new IntWritable(num * num * num));
        }
    }

    // Reducer class
    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {

        // Reduce function
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output,
                Reporter reporter) throws IOException {

            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    // Main function
    public static void main(String args[]) throws Exception {
        JobConf conf = new JobConf(sumOfCubes.class);

        conf.setJobName("sumOfCubes");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0])); //n numbers is located in the file
        FileOutputFormat.setOutputPath(conf, new Path(args[1])); // we get back the sum of cubes in this output file

        JobClient.runJob(conf);
    }
}
```

9. **Difference between RDD transformation and actions:**

Transformation	Actions
Transformation is a function that produces RDD from existing RDDs.	Actions returns a result based on an RDD, it can either be returned or saved to an external storage system.
Transformations are lazy, i.e., their result RDD is not computed immediately. This helps limit network call.	Actions are eager, i.e., result is immediately computed. An action
Transformations will create a DAG (Directed Acyclic Graph) using the applied operation, source RDD and function used for transformation.	An action will send data from executer to driver. Here executors are agents that are responsible for running a task.
Optimization is not implemented during this phase.	While performing every actions, spark tries to implement optimizations by making a execution plan.
Example: map, flatMap, filter, groupByKey, distinct, etc.	Example: collect, count, take, reduce, foreach, etc.

10. Spark Programming to calculate sum of Cubes of n numbers:

```python
from pyspark import SparkContext
sc = SparkContext("local", "sumOfCubes")

def sumOfCubes(n):
    l = list(range(n))

    #Creating a RDD
    nums = sc.parallelize(l)

    #Pass each element through a function
    cubes = nums.map(lambda x: x * x * x)

    #Applying reduce action on RDD produced above
    result = cubes.reduce(lambda x, y: x + y)

    return result
```