DPC END TERM NOTES

JAVA RMI

It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM.

- Builds distributed applications
- Remote communications between java programs

Architecture of an RMI Application

- Server program: a remote object is created and reference of that object is made available for the client (using the registry).
- Client program: requests the remote objects on the server and tries to invoke its methods.

RMI uses **stub and skeleton** object for communication with the remote object.

STUB

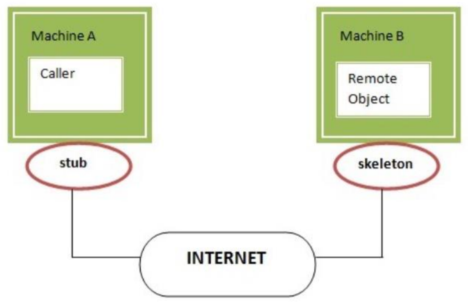
Gateway for client side and also resides in it.
When the caller invokes method on the stub object, it does the following tasks:

- It initiates a connection with remote Virtual Machine (JVM),
- It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM)
- It waits for the result
- It reads (unmarshals) the return value or exception
- It finally, returns the value to the caller

Skeleton

Gateway for the server-side object. When the skeleton receives the incoming request, it does the following tasks:

- It reads the parameter for the remote method
- It invokes the method on the actual remote object, and
- It writes and transmits (marshals) the result to the caller.



Server Application

Naming class provides 5 methods:

Lookup: It returns the reference of the remote object.

Bind: It binds the remote object with the given name.

Unbind: It destroys the remote object which is bound with the given name.

Rebind: It binds the remote object to the new name.

List: It returns an array of the names of the remote objects bound in the registry.

STEPS:

1) **Creating the remote interface**
```
import java.rmi.*;
public interface Adder extends Remote{
public int add(int x,int y)throws RemoteException;}
```
2) **Implementing the Remote interface**
```
import java.rmi. *;
import java.rmi.server.*;
public class AdderRemote extends UnicastRemoteObject implements Adder{
AdderRemote()throws RemoteException {
super();}
public int add(int x int y) {return x+y;}
```
3) **Create Stub and skeleton**
```
rmic AdderRemote
```
4) **Start the registry Service**
```
rmiregistry 5000
```
5) **Creating and running server application**
```
import java.rmi. *
import java.rmi.registry. * ;
public class MyServer{
public static void main(String args[]) {
try {
Adder stub=new AdderRemote();
Naming.rebind("rmi://localhost:5000/sonoo",stub);
} catch(Exception e) {System.out.println(e);}}}
```
6) **Creating and running the client application**
```
import java.rmi.*;
public class MyC1ient{
public static void main(String args[]){
try {
```

```
Adder stub=(Adder)Naming.lookup("rmi: localhost:5000/sonoo");
System.out.print1n(stub.add(34,4));
} catch(Exception e) { }}}
```

- Compiling: javac *.java
- Stub: rmic AdderRemote
- Registry: rmiregistry 5000
- Server start: java MyServer
- Client start: java MyClient

OpenMP

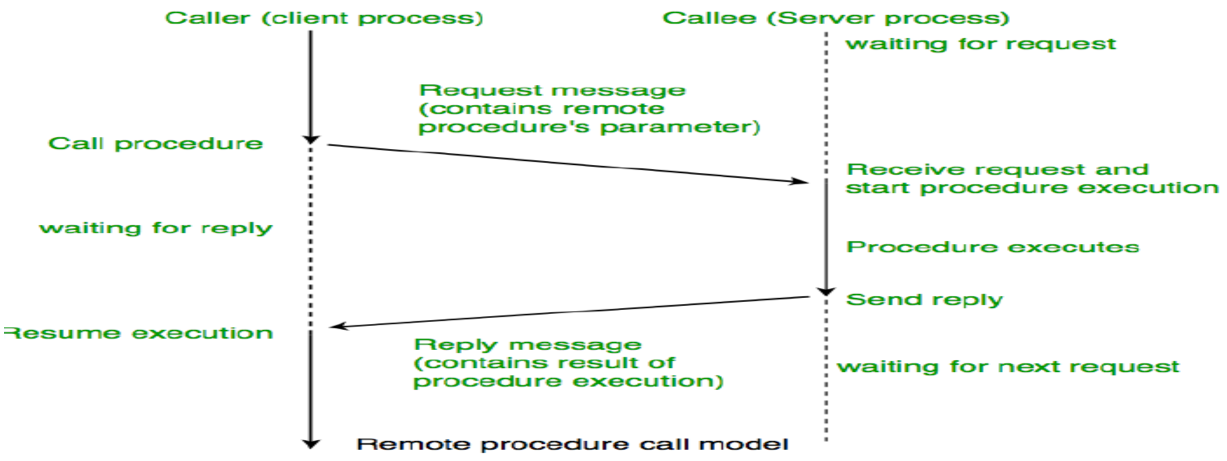
- OpenMP is a standard parallel programmmg API for shared memory environments, written in C, C++, or FORTRAN.
- First of all, OpenMP provides a cross-platform, cross-compiler solution.
- Secondly, using OpenMP can be very convenient and flexible to modify the number of threads.
- Thirdly, using OpenMP to create threads is considered to be convenient and relatively easy.

Code snippet:

```
#include<omp.h>
#pragma omp parallel{
printf("Hello World... from thread = %d\n", omp_get thread num()); (/Parallel region code)
}
```

Remote Procedure Call (RPC)

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications.



Remote procedure call model

NOTE: RPC is especially well suited for client-server (e.g. query-response) interaction in which the flow of control alternates between the caller and callee.
Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again.
Key Considerations for Designing and Implementing RPC Systems are:
Security: authentication, encryption, and authorization implemented to prevent unauthorized access and protect sensitive data.
Scalability: As the number of clients and servers increases, the performance of the RPC system must not degrade.
Fault tolerance: The RPC system should be resilient to network failures, server crashes, and other unexpected events.
Standardization:
Performance tuning: Fine-tuning the RPC system for optimal performance is important. This may involve optimizing the network protocol, minimizing the data transferred over the network, and reducing the latency and overhead associated with RPC calls.
ADVANTAGES :

- RPC provides ABSTRACTION
- RPC often omits many of the protocol layers to improve performance.
- RPC enables the usage of the applications in the distributed environment, not only in the local environment.
- With RPC code re-writing re-developing effort is minimized.
- Process-oriented and thread-oriented models supported by RPC.

Synchronization in Distributed Systems

Synchronization in distributed systems is achieved via clocks. The physical clocks are used to adjust the time of nodes.
Each node in the system can share its local time with other nodes in the system.
The time is set based on UTC (Universal Time Coordination).
Clock synchronization can be done by 2 ways: **External and Internal Clock Synchronization**.
External: external reference clock present
Internal: Each node share time with other.
2 Clock synchronization algorithms: Centralized and Distributed.

- **Centralized** is the one in which a time server is used as a reference. It is dependent on a single time-server, so if that node fails, the whole system will lose synchronization.
    - **Examples** of centralized are-**Berkeley the Algorithm, Passive Time Server, Active Time Server**
- **Distributed** is the one in which there is no centralized time-server present. Instead, the nodes adjust their time by using their local time and then, taking the average of the differences in time with other nodes.

- Distributed algorithms overcome the issue of centralized algorithms like scalability and single point failure.
    - **Examples** of Distributed algorithms are — **Global Averaging Algorithm, Localized Averaging Algorithm, NTP (Network time protocol)**

Logical Clock in Distributed System

Logical Clocks refer to implementing a protocol on all machines within your distributed system, so that the machines are able to maintain consistent ordering of events within some virtual timespan.

A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system.

Ex: there are 10 pc working on own clocks. Logical clock is used to make them work together in one clock.

What is causality ?

Causality is fully based on HAPPEN BEFORE RELATIONSHIP.

For 2 events occuring one after another then TS(A) <TS(B)

Distributed Algorithm is an algorithm that runs on a distributed system

Election Algorithm

- Election algorithms choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on other processor.
- It determines where a new copy of the coordinator should be restarted.
- It ensures each active process to have a unique priority number. The process with highest priority will be chosen as a new coordinator. Then this number is sent to every active process in the distributed system.

2 election algorithms for two different configurations of a distributed system.

1. **The Bully Algorithm** — This algorithm applies to system where every process can send a message to every other process in the system.
2. **The Ring Algorithm** — This algorithm applies to systems organized as a ring(logically or physically). In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its right only. Data structure that this algorithm uses is active list, a list that has a priority number of all active processes in the system.

Types of cloud computing

Public cloud

- Public clouds are owned and operated by third-party cloud service providers, which deliver computing resources like servers and storage over the internet.
- Microsoft Azure is an example of a public cloud.
- With a public cloud, the cloud provider owns and manages all hardware, software, and other supporting infrastructure.
- You access these services and manage your account using a web browser.

Private cloud

- A private cloud refers to cloud computing resources used exclusively by a single business or organization.
- A private cloud can be physically located on the company's onsite data center.
- Some companies also pay third-party service providers to host their private cloud.
- A private cloud is one in which the services and infrastructure are maintained on a private network.

Hybrid cloud

- Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them.
- By allowing data and applications to move between private and public clouds, a hybrid cloud gives your business greater flexibility and more deployment options and helps optimize your existing infrastructure, security, and compliance.

Characteristics of SaaS

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users are not responsible for hardware and software updates. Updates are applied automatically.
- The services are purchased on a pay-as-per-use basis

Example: BigCommerce, Google Apps, Salesforce, Dropbox, ZenDesk, Cisco WebEx, Slack, and GoToMeeting.

Characteristics of PaaS

- Accessible to various users via the same development application.
- Integrates with web services and databases.
- Builds on virtualization technology, so resources can easily be scaled up or down as per the organization's need.
- Support multiple languages and frameworks.
- Provides an ability to "**Auto-scale**".

Example: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, Magento Commerce Cloud, and OpenShift.

Characteristics of IaaS

- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible
- GUI and API-based access
- Automated administrative tasks

Example: DigitalOcean, Linode, Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE), Rackspace, and Cisco Metacloud.

IaaS	**Paas**	**SaaS**
It provides a virtual data center to store information and create platforms for app development, testing, and deployment.	It provides virtual platforms and tools to create, test, and deploy apps.	It provides web software and apps to complete business tasks.
It provides access to resources such as virtual machines, virtual storage, etc.	It provides runtime environments and deployment tools for applications.	It provides software as a service to the end-users.
It is used by network architects.	It is used by developers.	It is used by end users.
IaaS provides only Infrastructure.	PaaS provides Infrastructure+Platform.	SaaS provides Infrastructure+Platform +Software.

IaaS Advantages:

- Shared infrastructure
- Web access to the resources
- Pay-as-per-use model
- Focus on the core business
- On-demand scalability

Disadvantages: Security, maintenance and upgrade, interoperability issues

Platform as a Service (PaaS)

- Programming languages: Java, PHP Ruby, Perl, and Go.
- Application frameworks: Node.js, Drupal, Joomla, WordPress, Spring, Play, Rack, Zend.
- Databases: ClearDB, PostgreSQL, MongoDB, and Redis.

Paas advantages:

- Simplified Development
- Lower risk
- Prebuilt business functionality
- Instant community
- Scalability

Disadvantages: vendor lock-in, data privacy, integration with rest of system.

Salesforce-SalesForce.com, Google-Google App Engine, Cloud Foundary from VMware Openshift, AppFog, Windows Azure.

Software as a Service [SaaS]

SaaS is also known as "On-Demand Software". It is a software distribution model in which services are hosted by a cloud service provider.

Advantages:

- SaaS is easy to buy
- One to Many
- Less hardware required for SaaS
- Low maintenance required for SaaS
- No special software or hardware versions required.

Disadvantages: security, latency issue, totally dependent on internet.,difficulty in switching bwtn vendors

Virtualization in Cloud Computing

Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".

Creation of a virtual machine over existing operating system and hardware is known as Hardware Virtualization.

The machine on which the virtual machine is going to create is known as Host Machine and that virtual machine is referred as a Guest Machine.

Types of Virtualization

1) Hardware Virtualization:

Virtual machine manager (VMM) is directly installed on the hardware system

done for the server platforms, because controlling virtual machines is much easier than controlling a physical server.

Advantages:

1) More Efficient Resource Utilization
2) Lower Overall Costs Because Of Server Consolidation
3) Increased Uptime Because Of Advanced Hardware Virtualization Features

2) Operating System Virtualization:

the virtual machine software or virtual machine manager (VMM) is installed on the Host operating system Instead.

used for testing the applications on different platforms of OS.

3) Server Virtualization:

the virtual machine software or virtual machine manager (VMM) is directly installed on the Server system is known as server virtualization.

single physical server can be divided into multiple servers on the demand basis and for balancing the load.

Types of Server Virtualization

1. Hypervisor

In the Server Virtualization, Hypervisor plays an important role. It is a layer between the operating system (OS) and hardware. There are two types of hypervisors.

- **•Type 1 hypervisor (also known as bare metal or native hypervisors)**
- **•Type 2 hypervisor (also known as hosted or Embedded hypervisors)**

2. Full Virtualization

Full Virtualization uses a hypervisor to directly communicate with the CPU and physical server.

4) Storage Virtualization:

Storage virtualization is the process of grouping the physical storage from multiple network storage devices so that it looks like a single storage device.

Storage virtualization is a major component for storage servers, in the form of functional RAID levels and controllers. Operating systems and applications with device can access the disks directly by themselves for writing.

The main usage of Virtualization Technology is to provide the applications with the standard versions to their cloud users, they provide money on monthly/annual basis.

Data Virtualization

Process of retrieve data from various resources without knowing its type and physical stored location

collects heterogeneous data from different resources and allows access as per requirement.

We can use Data Virtualization in the field of data integration, business intelligence, and cloud computing.

Uses of Data Virtualization

- Analyze performance
- Search and discover interrelated data
- Agile Business Intelligence
- Data Management

Data Virtualization Tools

1. Red Hat JBoss data virtualization

Red Hat Virtualization is the best choice for developers and those who are using microservices and containers. It is written in Java.

2. TIBCO data virtualization

TIBCO helps administrators and users to create a data virtualization platform for accessing multiple data sources and data sets

3. Oracle data service integrator

It is a very popular and powerful data integrator tool that mainly works with Oracle products. It allows organizations to quickly develop and manage data services to access a single view of data.

Software Virtualization

Software virtualization is just like a virtualization but able to abstract the software installation procedure and create virtual software installations.

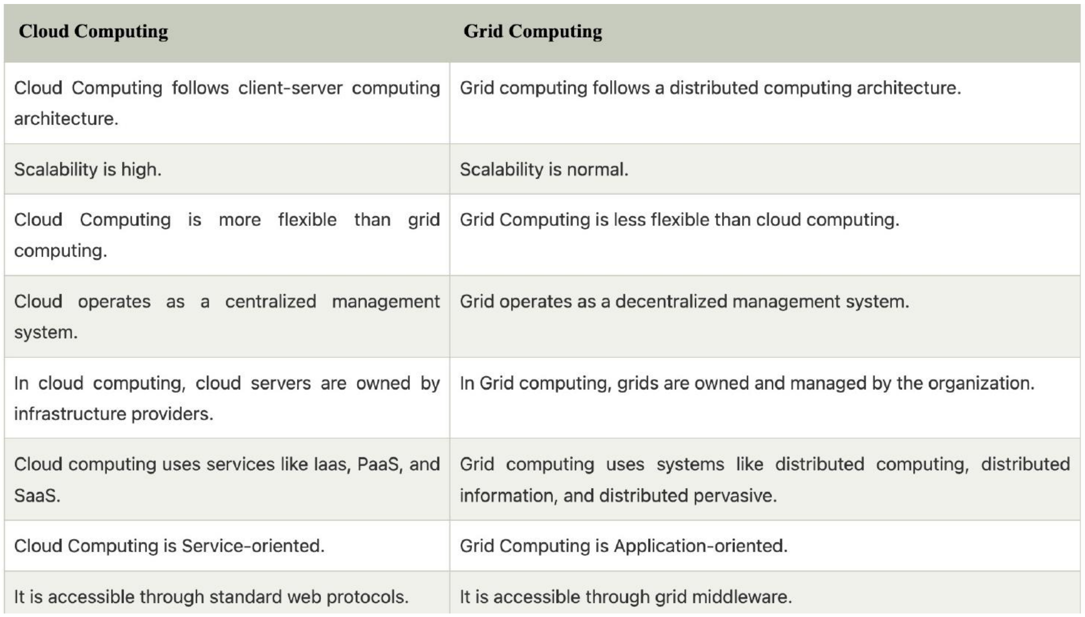
Example of software virtualization is **VMware software, virtual box** etc.

Advantages :

1) Client Deployments Become Easier

2) Easy to manage

3) Software Migration



Cloud Computing	Grid Computing
Cloud Computing follows client-server computing architecture.	Grid computing follows a distributed computing architecture.
Scalability is high.	Scalability is normal.
Cloud Computing is more flexible than grid computing.	Grid Computing is less flexible than cloud computing.
Cloud operates as a centralized management system.	Grid operates as a decentralized management system.
In cloud computing, cloud servers are owned by infrastructure providers.	In Grid computing, grids are owned and managed by the organization.
Cloud computing uses services like Iaas, PaaS, and SaaS.	Grid computing uses systems like distributed computing, distributed information, and distributed pervasive.
Cloud Computing is Service-oriented.	Grid Computing is Application-oriented.
It is accessible through standard web protocols.	It is accessible through grid middleware.

WEB SERVICES

A web service is a

- standardised method for propagating messages between client and server applications on the World Wide Web.
- set of open protocols and standards that allow data exchange between different applications or systems

A client invokes a web service by submitting an XML request, to which the service responds with an XML response. [Extensible Markup Language]

WEB SERVICE COMPONENTS

XML and HTTP is the most fundamental web service platform.

SOAP (Simple Object Access Protocol)

1. transport-independent messaging protocol.
2. XML data sent in form of SOAP messages.
3. In XML document root element is the 1st element.

The "envelope" is divided into two halves.

- Header: contains routing data, info of xml document
- Body: contains real message.

UDDI (Universal Description, Search, and Integration)

- a standard for specifying, publishing and searching online service providers.
- provides a specification that helps in hosting the data through web services.
- provides a repository where **WSDL(Web Service Description Language)** files can be hosted

WSDL (Web Services Description Language)
A WSDL file is another XML-based file that describes what a web service does with a client application. It helps client application to understand where the web service is located and how to access it.

Features of Web Service

- **XML-based**
- **Loosely Coupled**
- **Ability to be synchronous or asynchronous**
- **Supports remote procedural calls**
- **Supports document exchanges**

GOOGLE COLAB: Software as a service.