Introduction

A Java-based technology called Remote Method Invocation (RMI) enables objects in a distributed network to call methods on each other. Within the Java development platform and coding language, Remote Method Invocation (RMI) is a type of application programming interface (API). RMI offers a means of generating distributed Java applications via straightforward method calls. It is used to create distributed applications that require cooperation and communication between numerous components, many of which are operating on separate machines. It enables distributed network communication between objects operating on multiple Java Virtual Machines (JVMs) and computers.

## Background and Context of RMI Application Development Project

The following are the background and context related to the Remote Method Invocation app development project:

1. Distributed Computing:

When the capabilities of a program might be enhanced by dividing it into different components, interacting components that operate on various machines, distributed computing becomes necessary. There are a few possible reasons for this, including fault tolerance, scalability, and geographic distribution. This system allows the simultaneous functioning of the application even in terms of error outbreak in the parts of the application.

1. Java and RMI:

Java is a well-liked option for developing distributed systems due to its compatibility with RMI and platform independence. With the use of RMI in Java, it allows a constant communication among the different components of the application with one being the server and the other client. Java objects can communicate transparently over a network with each other just like local objects thanks to RMI.

1. Use Cases:

The use of RMI can be applied in making programs where one of the device functions as server and the other functions as the client. Developing client-server applications, creating distributed databases, developing remote administration and control systems, and other uses are all possible using RMI. It would be necessary to define the particular use case in the context of the RMI project.

1. The Object-Oriented Approach:

The RMI Java Approach helps to enable an object-oriented functioning of the application. Remote entities in a dispersed network can access their methods using an object-oriented paradigm, which forms the basis of Remote Machine Interface (RMI). This facilitates the design and conceptualization of systems that are distributed.

1. Security and Authentication:

The terms of security the RMI is equipped with several security policies for authentication of the users. Security is a critical issue in a system that is distributed. The project ought to take into account the best ways to protect RMI interaction using SSL and authentication methods.

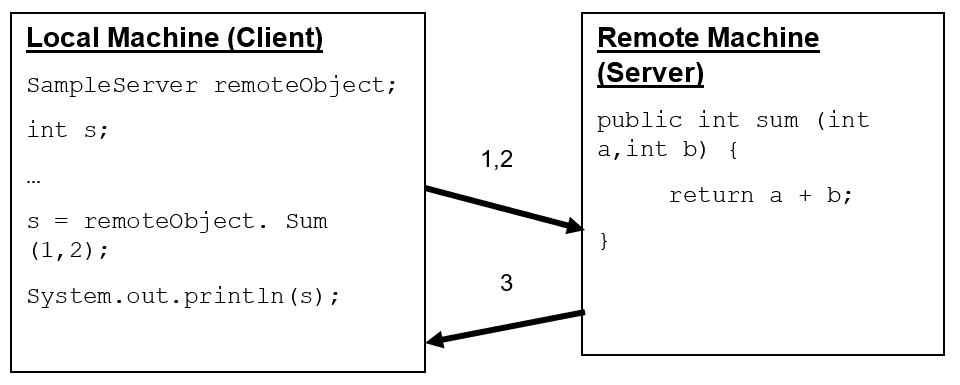


Figure : RMI

# Cloud computing and virtualization's importance in distributed systems

Both virtualization and cloud computing plays a crucial role in the distributed systems and has changes linked to the operation and view of the application based on RMI. The following are the significance of them:

1. Scalability:

Scalability is one of the crucial features that get available with the use of virtualization and cloud computing. Depending on demand, cloud computing offers the flexibility to scale resources up or down.

1. Geographic Distribution:

With the application of virtualization, the distribution of the program among different geographical area gets enabled. Generally, services related to cloud computing are accessible from a number of data centers worldwide.

1. Elasticity:

Elasticity is another feature which is unlocked with the use of virtualization. This enables auto correction of resources during use and rest. Auto-scaling capabilities in cloud-based RMI systems allow them to adjust to shifting traffic patterns.

1. Resource Isolation:

The concept of resource isolation can be defined as the separation of the resources based on the needs of the application. In a distributed system, resource isolation is crucial to prevent interaction between various components and applications causing malfunctioning. Virtualization technologies like as Virtual Machines and containers are utilized for this.

1. Disaster Recovery:

Options for disaster recovery are available with the help of cloud services. Cloud providers implement the data replication and failover techniques which are useful for RMI applications. This helps to backup important resources in terms of system failure for the application. Disaster Recovery is one of the crucial features which is provided by virtualization of the application.

1. Cost Effectiveness:

Another crucial advantage of virtualization is cutting down unnecessary expenses and using the resources efficiently. Since the payment is needed for only the resources used, cloud computing can be financially beneficial. Based on demand in real time, the optimization of the resources and its allocation is available for RMI applications.

# RMI Framework and its Role in Distributed Systems

Remote Method Invocation (RMI) is a Java technology for the development of distributed systems which allows the object in Java to communicate among one another and call methods to invoke the objects which are located remotely on different Virtual Machines or environments. RMI works as a suitable infrastructure for the development of distributed system applications. It is a technology which works by the invocation methods or objects remotely.

The role of RMI is crucial in distributed systems as it acts as a bridge which connects the objects in the server to the different machines allowing them to call the methods. The RMI is important in the development of applications where there is necessity of various different components distributed across different computer systems. The network communication details can be easily handled, which allows easy and quick design and deployment of distributed systems.

# Virtualization and Cloud Computing

Virtualization is defined as the process which allows the system to operate two or more operation systems and allocate the existing hardware resources among them. This helps to create a multiple computer by hosting them from a single computer system. Each system has their own independent operating system and applications. This helps to create an efficient usability of the available resources and separate the applications from one another.

Cloud Computing helps to enhance the virtualization and make it more efficient and effective. This enables the user to have access to the resources without the need of the actual physical hardware resources. The cloud computing allows the host to rent out the resources from a cloud. The providers have their own distinct services such as allow for the creation of the virtual servers which have their own operating system and necessary software. Along with that, cloud computing makes virtualization resource efficient as well as cost efficient as the payment of only the resources which is actually used is needed.

# Objective and Goals

# Overview of RMI application architecture and components

Initially, the server has to connect its name to the register. To create remote references, the client looks for the server’s name in the registry. The skeleton calls the remote method, serializes the result back to the stub, and the stub serializes the arguments to the skeleton.

The following are the components and concepts which are considered in the framework of RMI:

**Remote Interface:**

Remote interface is the component which is created to specify the invocation of remote methods in the system. In RMI, the declaration of methods which is to be called remotely is defined in these interfaces. Within the interface, they are extended from the ‘java.rmi.Remote’ interface for parent properties. This contains only the signatures of the method without their implementations.

**Remote Objects:**

The Java objects which are used for the implementation of remote interface are called remote objects. These remote objects help to provide the actual implementation of the remote methods and are hosted on the servers. The remote object provides the execution of the method on the server and a response is provided to the client when an RMI call is initiated to the remote object.

**Stubs and Skeletons:**

The client-side proxies linked to remote objects are known as stubs. These stubs are responsible for the implementation of remote interface workings as remote objects which serializes the calls of the methods and directs them to the server, and deserializes the output. Whereas, the server-side components which handles the clients’ RMI requests are known as skeletons. The skeletons are responsible for calling methods and redirecting them based on remote object with the proper output delivered back to the client.

**RMI Registry:**

RMI Registry is a directory registry which helps to assign a url or name to the remote objects. This registry helps the users to search for the particular remote objects with the help of the name or url of them for receiving the references of the object. The obtained references are responsible for the creation of method calls.

**RMI API:**

The RMI API stands for Remote Method Invocation Application Programming Interface which is a collection of different interfaces and classes available to the users from Java’s package called ‘java.rmi’. The package includes the remote object management, naming and classes for registry. This RMI API is responsible for managing the parameter marshalling, serialization of object, and communication in network.

**Java Virtual Machine**

Java Virtual Machine is the crucial component of RMI which is responsible for the execution of the codes of Java in both the client and the server section. The component is responsible for creating and executing the objects remotely and ensuring for the execution of the RMI method calls properly.

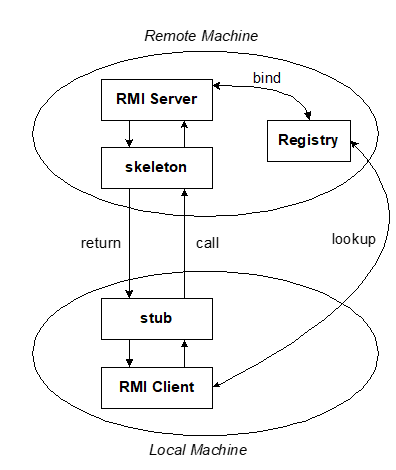


Figure : Architecture of RMI