SSC150 – Sistemas Computacionais Distribuídos

Comunicação em Sistemas Distribuídos
Java RMI

5ª aula
08/04/10

Profa. Sarita Mazzini Bruschi
sarita@icmc.usp.br

Material elaborado pelo aluno de mestrado Geraldo Guiesi

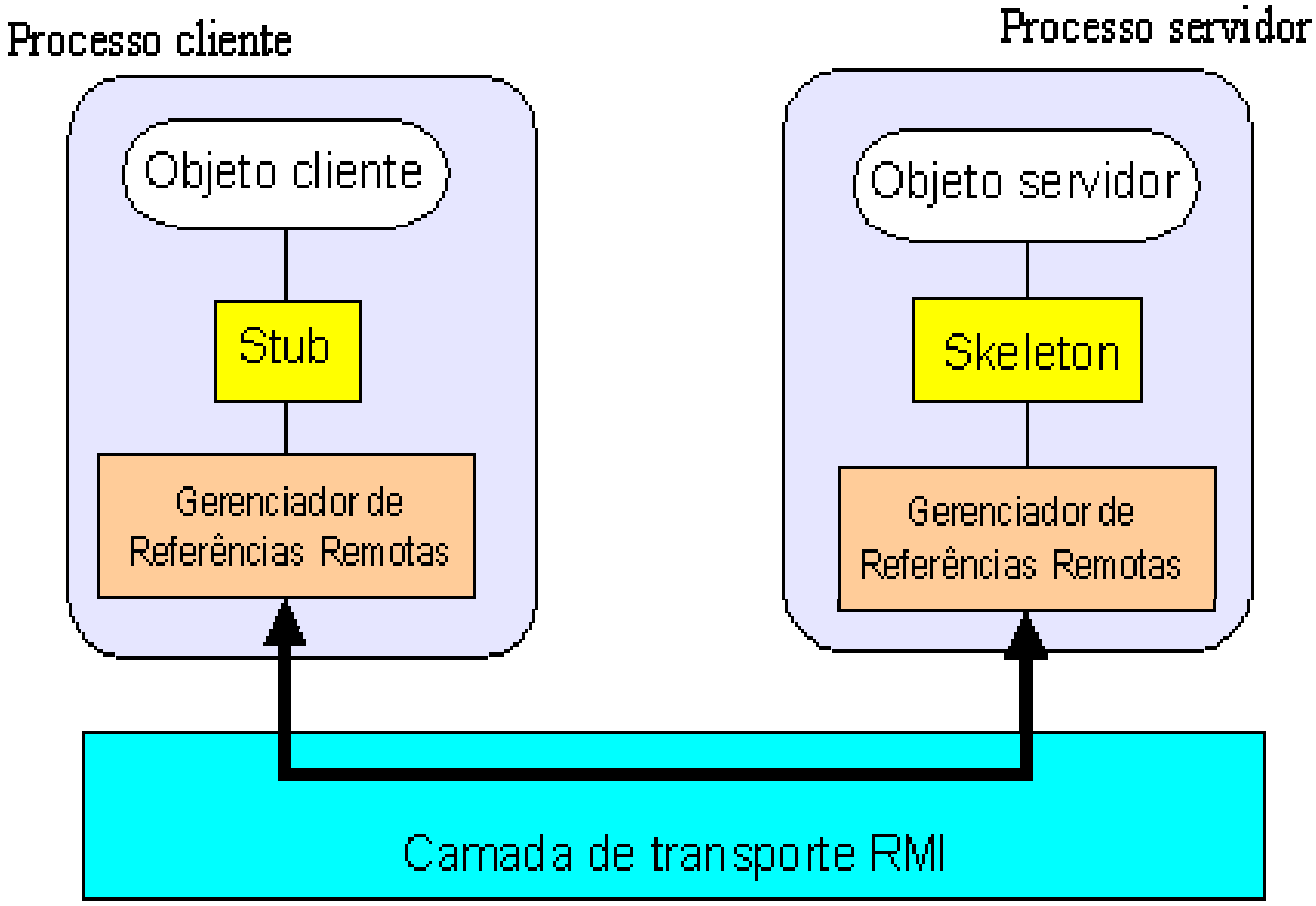
Introdução

- **Proposta:**
  - **Programação distribuída com a mesma sintaxe e semantica usada nos programas Java locais;**
- **Mapeamento de funcionamento de objetos Java em uma JVM (*Java Virtual Machine*) para funcionarem em ambiente distribuído (múltiplas JVM);**

Arquitetura RMI

- Um objeto ativo em uma máquina virtual Java interage com objetos de outras máquinas virtuais Java, independentemente da localização.

Arquitetura RMI



Arquitetura RMI

- A arquitetura RMI oferece três camadas entre os objetos cliente e servidor:
  - A camada de **stub/skeleton** oferece as interfaces que os objetos da aplicação usam para interagir entre si;
  - A camada de **referência remota** é o *middleware* entre a camada de stub/skeleton e o protocolo de transporte. É nesta camada que são criadas e gerenciadas as referências remotas aos objetos;
  - A camada do **protocolo de transporte** oferece o protocolo de dados binários que envia as solicitações aos objetos remotos pela rede.

Estrutura RMI

- Interface Remota
- Implementação dos serviços
- Criação de cliente RMI
- Criação de Servidor RMI

Interface Remota

- **Mesmas especificações de uma interface comum**
- **Interface deve estender a interface Remote**
- **Todo método da interface deverá declarar que a exceção RemoteException**

Conceitos RMI

- Interface remota
  - Exemplo:

```
import java.rmi.*;
public interface Count extends Remote {
    void set(int val) throws RemoteException;
    int get() throws RemoteException;
}
```

Implementaçao de serviços

- É preciso incluir as funcionalidades para que um objeto dessa classe possa ser acessado remotamente como um servidor.
- As funcionalidades de um servidor remoto são especificadas na classe abstrata **RemoteServer**, do pacote **java.rmi.server**.
- Uma subclasse concreta de **RemoteServer** oferecida no mesmo pacote é **UnicastRemoteObject**, que permite representar um objeto que tem uma única implementação em um servidor.

Conceitos RMI

- Implementação de serviços
  - Exemplo:

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class CountImpl extends UnicastRemoteObject
    implements Count {
    private int sum;
    public CountImpl() throws RemoteException {  super();  }
     public void set(int val) throws RemoteException { sum = val; }
    public int get() throws RemoteException { return sum; }
}
```

Criação de Cliente RMI

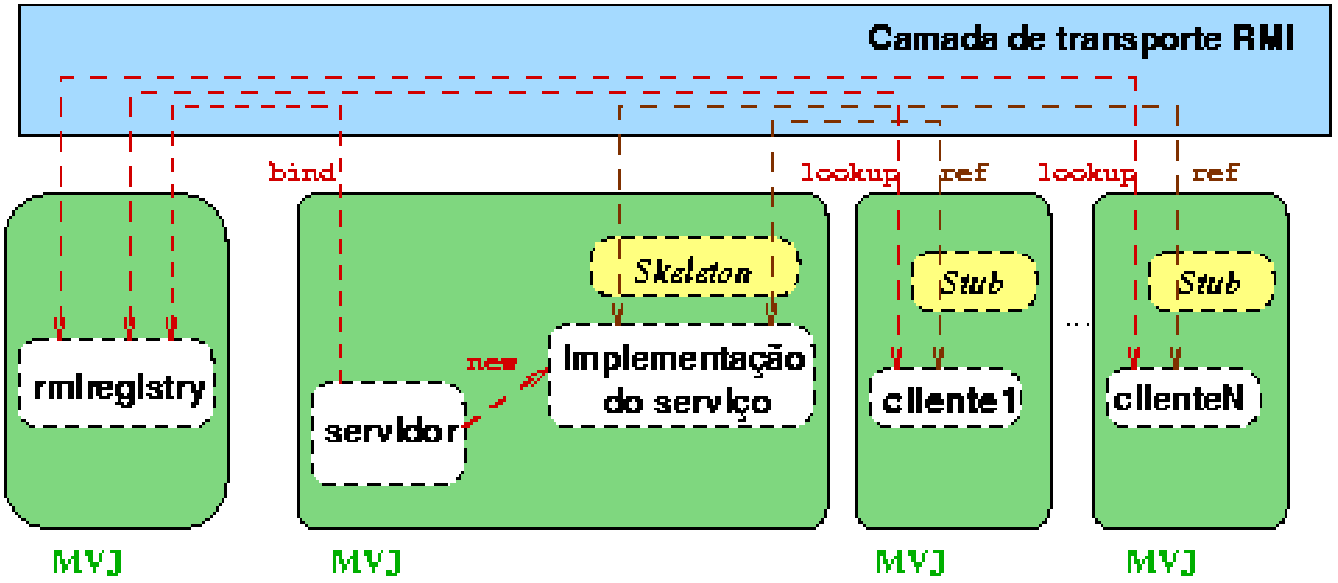
- O cliente RMI requer obtenção de uma referência remota para o objeto que implementa o serviço.
- Referência ocorre através do cadastro realizado pelo servidor. A operação com o objeto remoto é indistingüível da operação com um objeto local.

Definição Servidor

- Criar uma instância do objeto que implementa o serviço
- Disponibilizar o serviço através do mecanismo de registro

Execução

- A execução da aplicação cliente-servidor em RMI requer:
  - execução da aplicação cliente, execução da aplicação servidor, a execução do serviço de registro de RMI.



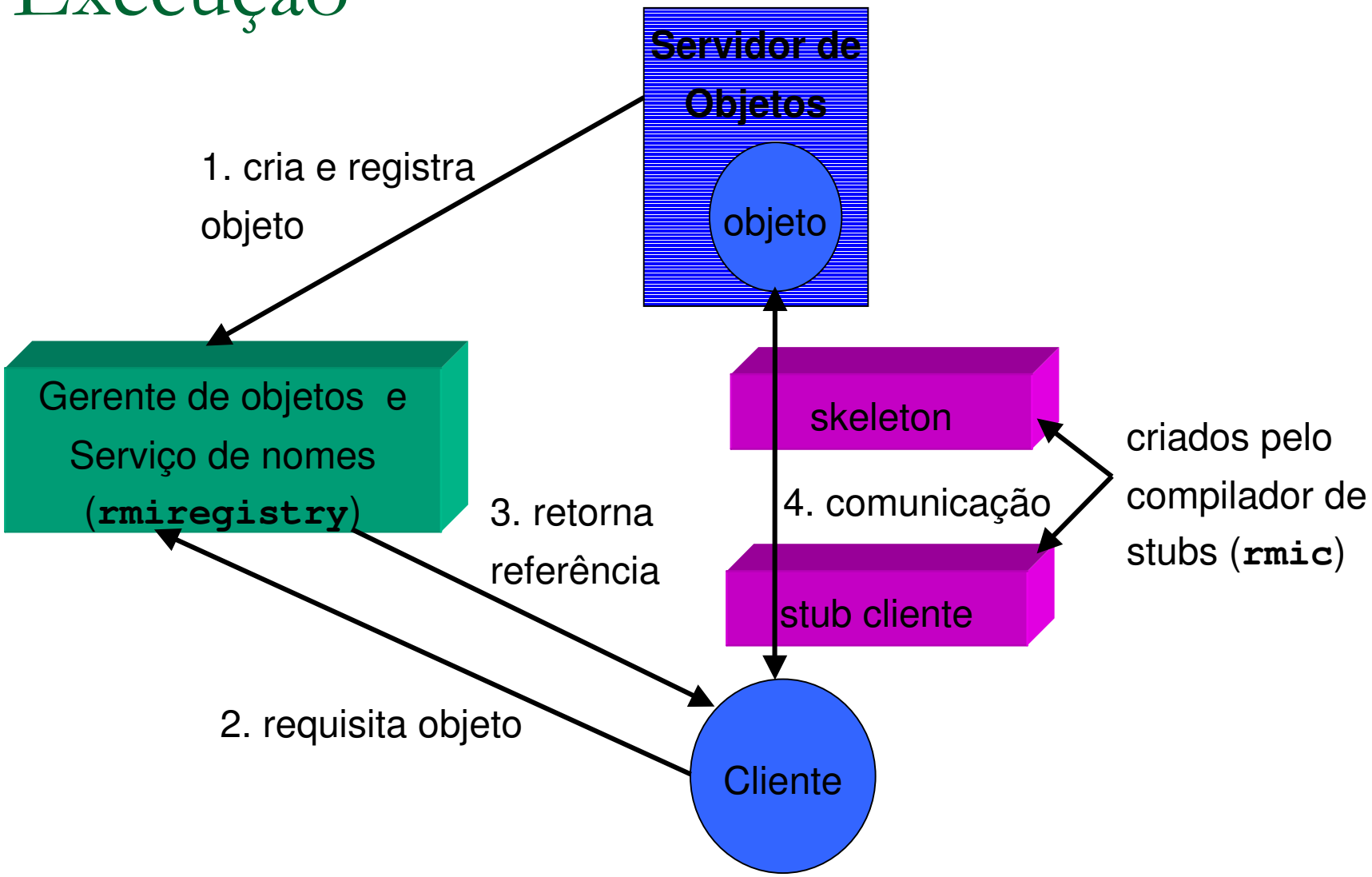
Execução

- Compilar os arquivos .java
  - Javac *.java
- Compilar RMI
  - rmic <Classe que implementa serviços>
- Registrar RMI
  - rmiregistry & // unix
  - start rmiregistry // dos

Execução

- Execução do Servidor
  - Java <servidor>
    - O Servidor regista no rmiregistry os serviços disponíveis
- Execução do Cliente
  - Java <cliente>

Execução


Servidor de Objetos
objeto
1. cria e registra objeto
Gerente de objetos e Serviço de nomes (rmiregistry)
3. retorna referência
2. requisita objeto
skeleton
4. comunicação
stub cliente
criados pelo compilador de stubs (rmic)
Cliente

Referências

- http://www.dca.fee.unicamp.br/cursos/PooJava/objdist/javarmi.html
- http://java.sun.com/products/jdk/rmi/reference/docs/index.html
- http://www.portaljava.com/home/modules.php?name=Content&pa=showpage&pid=8