Chamadas de Procedimentos Remotos em Python

A. Zamberlan e G. Kurtz

UFN

www.lapinf.com.br

Agenda

1. Contexto
2. Definições e Conceitos
3. Exemplo de código Java RMI
4. Exemplo de código RPC Python
5. Considerações

Agenda

Objetivo

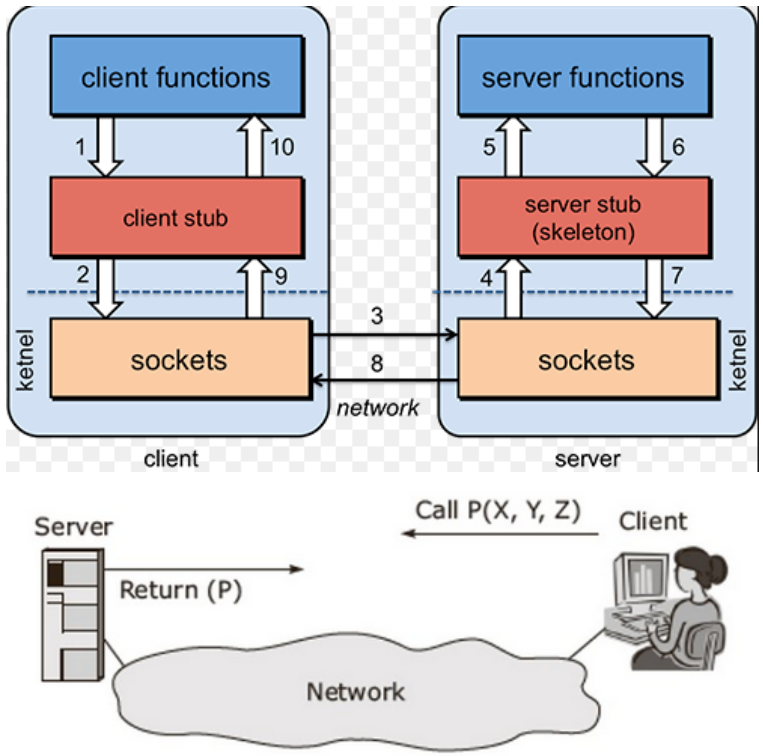
- Apresentar a dinâmica de funcionamento de programação RPC
- Discutir implementações:
  - RPC protocolo
  - RMI Java API
  - XML RPC
  - RPC Python sistema

Agenda

RPC - Remote Procedure Call

Protocolo comunicação

... que um programa utiliza para requisitar/solicitar serviço de um programa localizado em outro computador na rede sem ter que se preocupar com detalhes de rede.



RPC - Remote Procedure Call

Conceitos atrelados

- Sistemas Distribuídos + Sistema Operacional + Redes
- Arquitetura Cliente-Servidor
- Camada Transporte:
  - TCP (síncrono): Sistema bloqueante
    - servidor processa...
    - ...cliente espera
  - UDP (assíncrono): Não bloqueante
- Sockets
- Threads
- Programação Orientada a Objetos
- Serialização de objetos
- Bruce Jay Nelson cunhou **Remote Procedure Call** (1981)

RMI - Remote Method Invocation in Java

Application Program Interface

- ... fornece mecanismos para criar aplicações distribuídas em Java
- Permite um objeto chamar/invocar métodos de outro objeto rodando em outra JVM

XML RPC

Microsoft

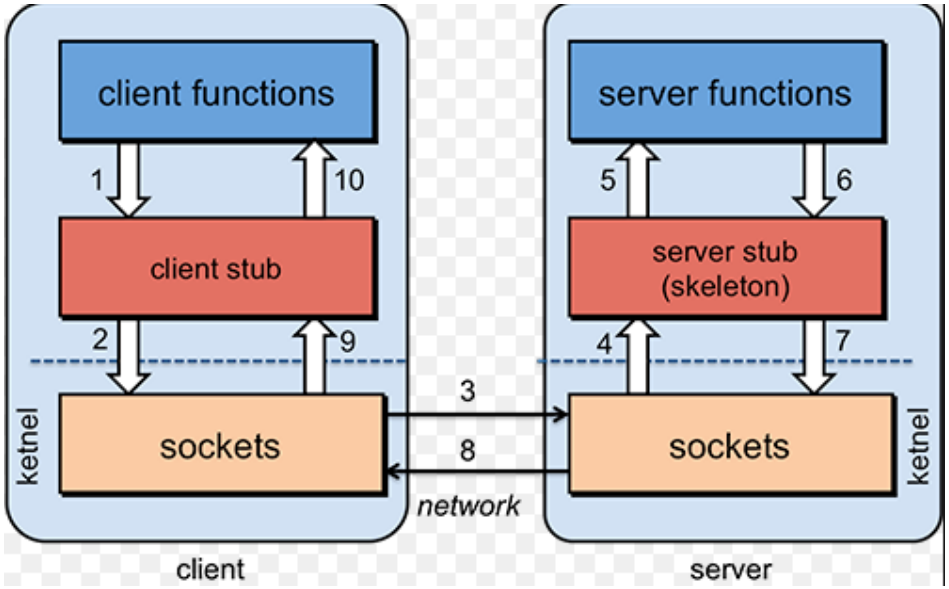
- business-to-business (e-commerce)
- ideia de ”human-readable-and-writable”
- script padrão baseado em HTML capaz de ser ’parseado’ (análise sintática

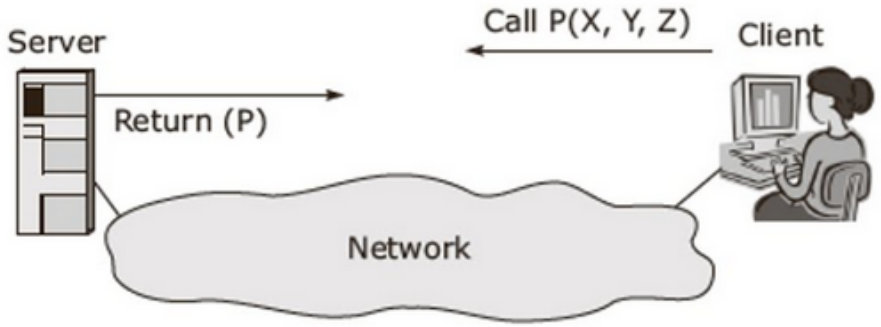
Python RPC

Sistema

- ... apresenta mesma funcionalidade RPC tradicional e RMI Java.
- CONTUDO, é mais "simples tratar tudo isso!!!
- MAS, e o desempenho?????

Remote Procedure Call





Agenda

RMI Java

```java
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface IHoje_eh extends Remote{
    public DataHora pegaDataHora() throws RemoteException;
}
```

```java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;

public class Hoje_eh extends UnicastRemoteObject implements IHoje_eh {

    public Hoje_eh() throws RemoteException{

    }

    @Override
    public DataHora pegaDataHora() throws RemoteException {
        DataHora obj = new DataHora();
        return obj;
    }
}
```

RMI Java

```java
import java.io.Serializable;
import java.sql.Time;
import java.util.Date;

public class DataHora implements Serializable {
    Date data;
    Time hora;

    public DataHora() {
        this.data = new Date();
        this.hora = new Time(1000);
    }
}
```

RMI Java

```java
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Server {
    String HOST_URL = "rmi://localhost/Hoje_eh";

    public Server(){
        try {
            LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
            Hoje_eh objetoRemoto = new Hoje_eh();
            System.out.println("servidor retornando a data/hora...");
            Naming.bind(HOST_URL, objetoRemoto);
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }

    public static void main(String args[]){
        new Server();
    }
}
```

```java
import java.rmi.Naming;

public class Client {
    public static void main(String args[]) {
        try {
            IHoje_eh d = (IHoje_eh) Naming.lookup("rmi://localhost/Hoje_eh");
            System.out.println("Data e hora vinda do servidor!!");
            System.out.println("Data: " + d.pegaDataHora().data);
            //System.out.println("Hora: " + d.pegaDataHora().hora);
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }
}
```

RMI Java - Saída

Prompt de Comando - java Server

```
C:\Users\usrlab25\Downloads\sistemasDistribuidos\7-RMI-Java\exemplo\src>java Server
servidor retornando a data/hora...
```

Prompt de Comando

```
C:\Users\usrlab25\Downloads\sistemasDistribuidos\7-RMI-Java\exemplo\src>java Client
Data e hora vinda do servidor!!
Data: Fri May 31 21:19:48 BRT 2019

C:\Users\usrlab25\Downloads\sistemasDistribuidos\7-RMI-Java\exemplo\src>java Client
Data e hora vinda do servidor!!
Data: Fri May 31 21:19:53 BRT 2019

C:\Users\usrlab25\Downloads\sistemasDistribuidos\7-RMI-Java\exemplo\src>java Client
Data e hora vinda do servidor!!
Data: Fri May 31 21:19:56 BRT 2019

C:\Users\usrlab25\Downloads\sistemasDistribuidos\7-RMI-Java\exemplo\src>java Client
Data e hora vinda do servidor!!
Data: Fri May 31 21:19:59 BRT 2019
```

Agenda

Python RPC

servidor.py

```python
from xmlrpc.server import SimpleXMLRPCServer
import datetime, xmlrpc.client

def hoje_eh():
    hoje = datetime.datetime.today()
    return xmlrpc.client.DateTime(hoje)

servidor = SimpleXMLRPCServer(("localhost", 8000))
print("Ouvindo a porta 8000...")
servidor.register_function(hoje_eh, "hoje")
servidor.serve_forever()
```

cliente.py

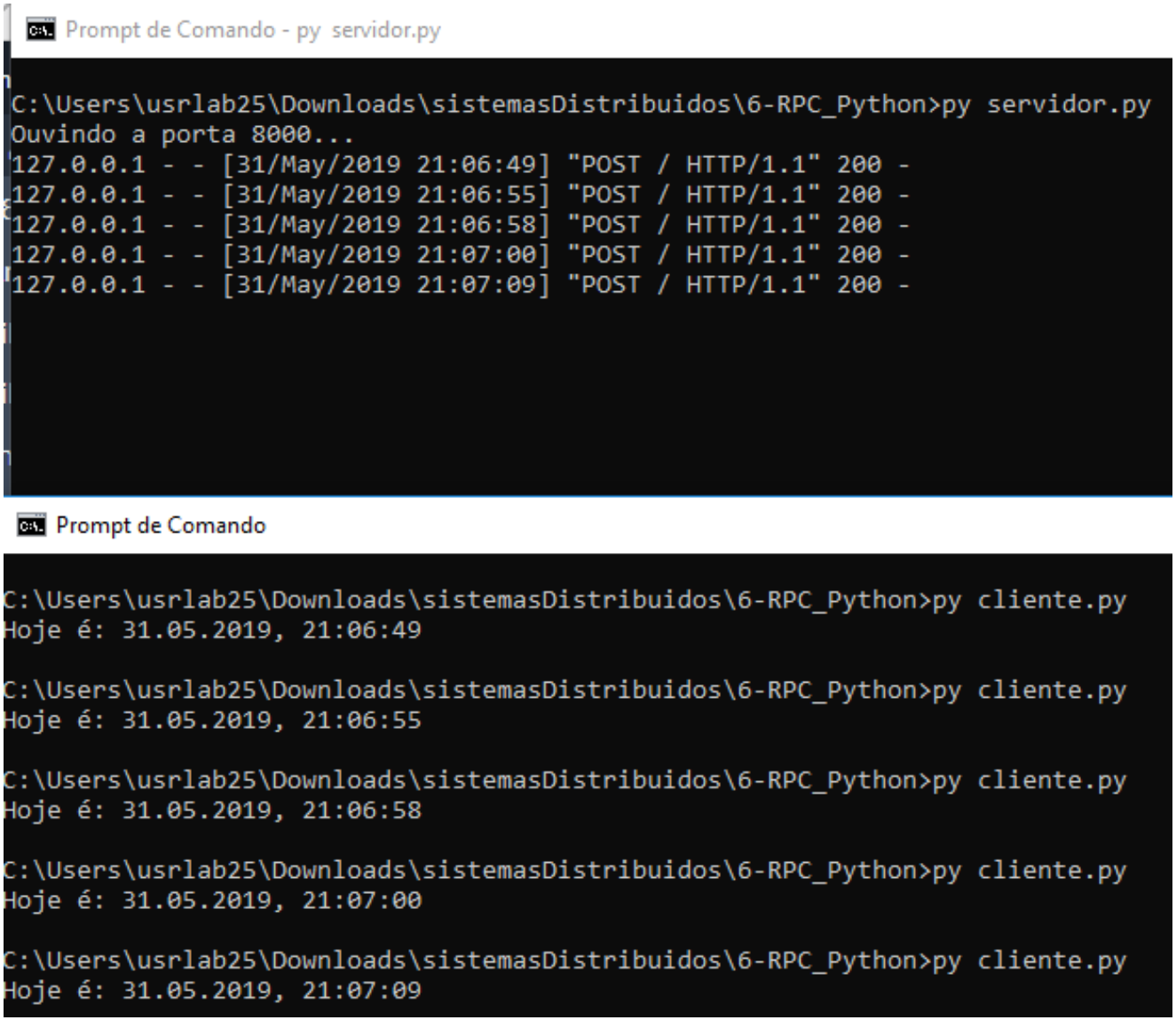
```python
import xmlrpc.client, datetime

servidor = xmlrpc.client.ServerProxy("http://localhost:8000/")

hoje = servidor.hoje()

# convert the ISO8601 string to a datetime object
data_hora_convertida = datetime.datetime.strptime(hoje.value, "%Y%m%dT%H:%M:%S")
print("Hoje é: %s" % data_hora_convertida.strftime("%d.%m.%Y, %H:%M:%S"))
```

Python RPC - Saída



Agenda

Considerações

- Simples e rápido para 'produção' e 'entrega'
- Desempenho é pior do que sockets
  - simplificar o desenvolvimento X melhorar desempenho
  - 'serviços' simples não faz diferença

Considerações

Abstração para o programador

- Sistemas Distribuídos + Sistema Operacional + Redes
- Arquitetura Cliente-Servidor
- Camada Transporte:
  - TCP (síncrono): Sistema bloqueante - server processa... cliente espera
  - UDP (assíncrono): Não bloqueante
- Sockets
- Threads
- Programação Orientada a Objetos
- Serialização de objetos

Contato

Alexandre Zamberlan e Guilherme Kurtz
alexz@ufn.edu.br e guilhermechagaskurtz@ufn.edu.br

https://github.com/alexandrezamberlan/sistemasDistribuidos

www.lapinf.com.br