Java RMI

Matěj Antoš a Gabriela Novotná

ČVUT FJFI, 04.12.2012

Obsah přednášky

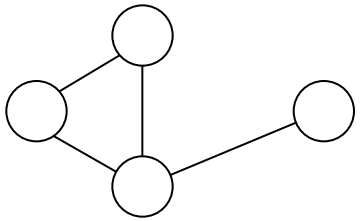
- Distribuovaný výpočet / Distribuované systémy a aplikace
- Java RMI
  - Představení technologie
  - Historie
  - Obecný postup návrhu RMI aplikace
  - Bezpečnost
  - Výhody a nevýhody
- Praktická ukázka

Distribuované systémy a aplikace

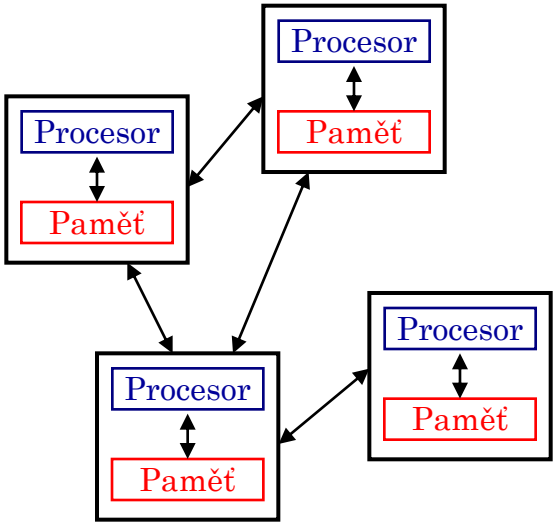
- Překrytí: paralelní – souběžný – distribuovaný

- Paralelní
  - Všechny procesory s přístupem ke sdílené paměti
- Distribuované
  - Každý procesor se svojí vlastní (distribuovanou) pamětí

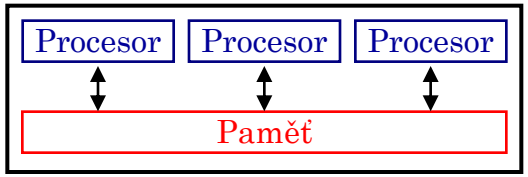
(a)



(b)


Procesor
Paměť
Procesor
Paměť
Procesor
Paměť
Procesor
Paměť

(c)


Procesor
Procesor
Procesor
Paměť

DISTRIBUOVANÉ SYSTÉMY A APLIKACE

- Rozdělení na více menších částí
- Pro rychlejší vyřízení požadavku
- Pro paralelizovatelné algoritmy

- Na úrovni operačního systému
- V režii programu

- Analýza velkého množství statistických dat

Distribuované systémy a aplikace

- 60. léta – pro vědecké účely
- 70. léta – ARPANET (email)
- 80. léta – samostatné odvětví

- Původně fyzicky, dnes i v rámci jedné jednotky
- Společný cíl: Rozsáhlý výpočet nebo pokrytí potřeb/komunikace (více) uživatelů

Distribuované systémy a aplikace

- Charakter systému
  - Několik nezávislých výpočetních jednotek s vlastní pamětí
  - Jednotky jsou schopné komunikovat zasíláním zpráv

  - Odolné vůči chybám
  - Jednotky systému nemusí znát okolí ani strukturu systému

DISTRIBUOVANÉ SYSTÉMY A APLIKACE

- Příklady využití
  - Aplikace si žádá komunikační síť spojující několik jednotek
    - Data v jedné jednotce potřebné ve druhé
  - V principu jedna jednotka – v praxi výhodnější více
    - Více jednotek nižší třídy rentabilnější než jedna nejvyšší třídy
    - Spolehlivější
    - Snadněji rozšiřitelný a spravovatelný

DISTRIBUOVANÉ SYSTÉMY A APLIKACE

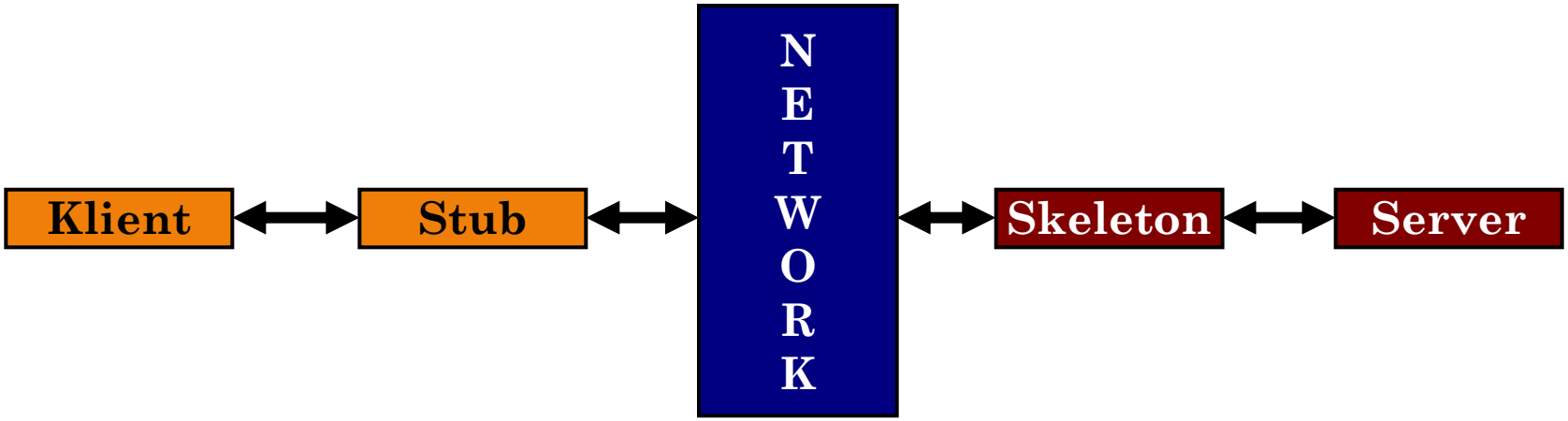
- Příklady využití
  - Telekomunikační sítě
  - Síťové aplikace
  - Řízení procesů v reálném čase
  - Paralelní programování
    - Biofyzikální projekty (POEM@home)
    - Simulace vývoje klimatu (ClimatePrediction.net)
    - Řešení matematických problémů (GFPS)

JAVA RMI – ÚVOD

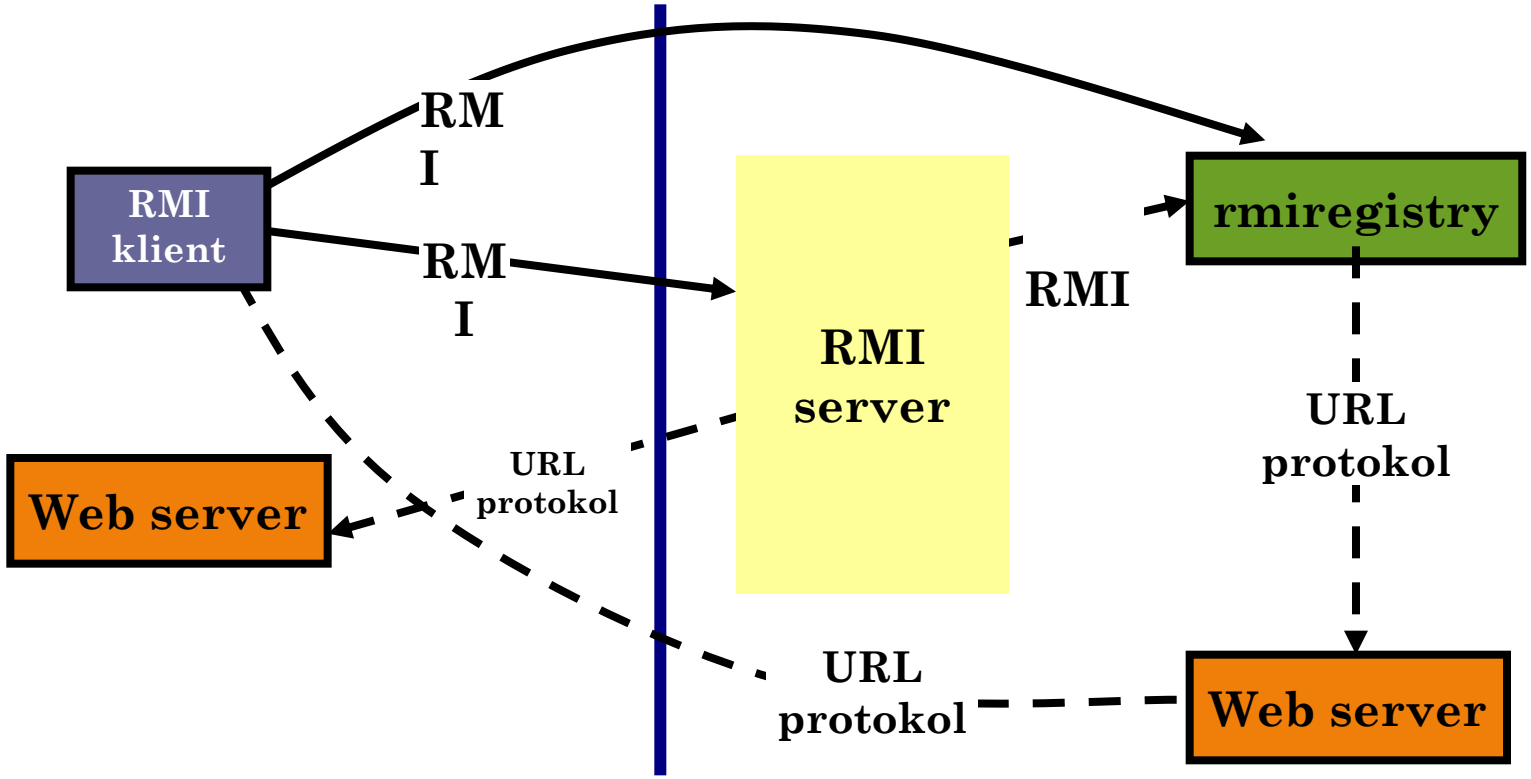
- **R**emote **M**ethod **I**nvocation
- API (Application Programming Interface) pro platformu JAVA

- Založeno na architektuře Klient – Server
- Umožňuje z jednoho virtuálního stroje (JVM) volat metody objektů na jiném JVM
- Využívá serializace objektů na platformě JAVA

Původní RMI model



Princip technologie RMI



Historie RMI

- Součástí JDK od verze 1.1
- JDK 1.2 odstranila nutnost Skeletona
- Od JDK 1.3 používán protokol RMI/IIOP (Internet-InterORB Protocol)
- JDK 1.5 a 1.6 výrazně zjednodušení syntaxe
- Pro spolupráci s aplikacemi mimo platformu JAVA byla vyvinuta i CORBA verze RMI

Popis API Java RMI

- Balíček java.rmi:
  - `java.rmi.Remote`
  - `java.rmi.RemoteException` a různé jiné vyjímky
  - `java.rmi.server.UnicastRemoteObject`
- Pomocná třída:
  - `java.lang.SecurityManager`

Postup implementace RMI aplikace

- Obecný postup:
  1. Definice RMI rozhraní (interfejsu)
  2. Implementace RMI serveru
  3. Implementace RMI klienta
  4. Kompilace zdrojů
  5. Zpřístupnění zdrojů na síti
  6. Spuštění aplikace

Doprovodná RMI aplikace

- Součást tutorialu od Oraclu
- Generická výpočetní aplikace
- Plně využívá serializace JAVY
- Volně dostupná na: http://docs.oracle.com/javase/tutorial/rmi/overview.html

Definice RMI rozhraní

- Určuje typy objektů v parametrech a návratové hodnoty
- Rozhraní musí rozšiřovat rozhraní `java.rmi.Remote`
- Každá metoda musí „vyhazovat“ vyjímku typu `java.rmi.RemoteException`

- Implementující RMI objekty musí:
  - Deklarovat implementaci příslušného RMI rozhraní pomocí klíčového slova implements
  - Deklarovat konstruktor pro každý vzdálený objekt
  - Implementovat metody uvedené v rozhraní

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

public interface Compute extends Remote
{
    <T> T executeTask(Task<T> t)  throws RemoteException;
}
```

Implementace RMI serveru

- Musí implementovat RMI rozhraní
- Pro jeho spuštění je nutné
  - Vytvořit a instalovat `SecurityManager`
  - Zpřístupnit RMI objekt klientům
  - Zaregistrovat „*stub*“ RMI objektu v registru vzdálených objektů

```
public class ComputeEngine implements Compute
{
    public ComputeEngine() {
        super();
    }

    public <T> T executeTask(Task<T> t) {
        return t.execute(); }

    public static void main(String[] args) {
       if (System.getSecurityManager() == null)
            { System.setSecurityManager(new  SecurityManager()); }
       try {
          String name = "Compute";
          Compute engine = new ComputeEngine();
          Compute stub=(Compute)UnicastRemoteObject.exportObject(engine, 0);
          Registry registry = LocateRegistry.getRegistry();
          registry.rebind(name, stub);
       }
       catch (Exception e) { … }
    }
}
```

Implementace RMI klienta

- Pro úspěšnou komunikaci se serverem je nutné:
  - Vytvořit a instalovat `SecurityManager`
  - Vyhledat daný server a připojit se na jeho registr vzdálených objektů
  - V registru vyhledat daný *„stub“*
- Dále lze s daným objektem pracovat jako kdyby byl na lokálním JVM

```
public class ComputePi
{
    public static void main(String args[])
    {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager()); }
        try {
            String name = "Compute";
            Registry registry=LocateRegistry.getRegistry(args[0]);
            Compute comp = (Compute) registry.lookup(name);
            Pi task = new Pi(Integer.parseInt(args[1]));
            BigDecimal pi = comp.executeTask(task);
            System.out.println(pi);
        }
        catch (Exception e){
            System.err.println("ComputePi exception:");
            e.printStackTrace();
        }
     }
}
```

Bezpečnost RMI

- Implicitně kódované, ale ne šifrované
- Bezpečnost lze vylepšit:
  - Použitím SSL socketů
    - Pomocí knihovny `javax.net.ssl`
    - Umožňují autentizaci jak na straně serveru, tak i klienta
    - Šifruje data pomocí algoritmu dle výběru (DES, 3DES, RS4 apod.)

- Využitím možností třídy `SecurityManager`
  - Využívá balíku java.security.manager
  - Bez jeho instalace žádná RMI aplikace nefunguje
  - Lze využít systém individuálních omezení pro jednotlivé RMI aplikace
  - Omezení na „*stuby*“, které lze aplikovat:
    - resolve – převod jména na číselnou IP adresu
    - accept – přijetí připojení
    - connect – vytvoření socketového připojení
    - listen – naslouchání na určitém portu

Výhody a nevýhody RMI

- Výhody:
  - Jednoduchá a čistá implementace vedoucí k robustnějšímu lépe udržovatelnému kódu
  - Není třeba instalace kromě JVM
  - Možnost naprogramovat klienta, který nevyžaduje žádné nastavování
  - Při změně funkčnosti na straně serveru je třeba kompilovat jen daný server

- Nevýhody:
  - Kompatibilní jen s platformou JAVA
  - Pomalejší komunikace než při využití socketů
  - Větší pozornost se musí věnovat bezpečnosti
  - Nutná identifikace klienta – není zaručeno, že klientské spojeni bude na stejném vlákně

Následuje ukázka implementace „Hello World“ pomocí Java RMI

Děkujeme za pozornost

Rádi odpovíme na případné dotazy