PAPER

"Advancements in Java RMI: Enhancing Scalability and Performance through Modern Implementations"

Disusun oleh :

1. Afif Dhaifulloh	(2211102040)
2. Fatimah Az Zahra	(2211102160)
3. Kharisma Nurhayati	(2211102027)
4. Laura Anisa Mafatikhur Rizqi	(2211102029)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

ABSTRAK

Java Remote Method Invocation (RMI) merupakan teknologi yang memfasilitasi panggilan objek secara remote, menyerupai panggilan lokal[1]. Secara luas digunakan dalam sistem terdistribusi dan aplikasi berbasis Internet, implementasi Java RMI tradisional dapat mengalami kekurangan ketika kompleksitas sistem dan permintaan akan skalabilitas dan kinerja meningkat. Paper ini mengulas kemajuan yang telah dilakukan dalam Java RMI untuk meningkatkan skalabilitas dan kinerjanya. Introduksi dilakukan terhadap implementasi-implementasi Java RMI kontemporer yang dirancang untuk mengatasi tantangan yang melekat pada pendekatan tradisional. Dengan memanfaatkan teknologi dan teknik terkini, implementasi-implementasi ini bertujuan untuk meningkatkan efisiensi dan efektivitas Java RMI dalam sistem terdistribusi. Bukti empiris disajikan untuk menegaskan peningkatan yang signifikan dalam kinerja dan skalabilitas yang diperoleh oleh implementasi-implementasi modern ini. Selain itu, paper ini membahas hambatan-hambatan yang dihadapi dalam mengimplementasikan kemajuan ini dan memberikan wawasan tentang arah masa depan Java RMI. Sebagai kesimpulan, paper ini menyajikan survei komprehensif tentang kemajuan Java RMI dan dampaknya terhadap kinerja dan skalabilitas sistem terdistribusi. Ini berfungsi sebagai sumber daya vital bagi pengembang dan peneliti yang tertarik untuk menjelajahi perkembangan terbaru dalam Java RMI dan aplikasi potensialnya dalam sistem terdistribusi kontemporer.

BAB I

PENDAHULUAN

Java Remote Method Invocation (RMI) adalah teknologi yang dibangun oleh Sun Microsystems (sekarang Oracle Corporation) untuk memungkinkan objek Java di dalam sistem berbeda untuk mengirimkan panggilan metode (invocation) ke objek Java di sistem lain[2]. Java RMI memungkinkan programmer untuk mengembangkan aplikasi berbasis jaringan yang dapat melakukan panggilan metode antar-proses tanpa perlu mengimplementasikan mekanisme komunikasi jaringan sendiri.

Java RMI digunakan untuk memanggil metode pada objek yang berada di mesin yang berbeda dalam jaringan dengan cara membuat objek client yang menangani panggilan metode dari objek server yang berada di sistem lain. Dalam implementasi RMI, objek client dapat mengirimkan panggilan metode ke objek server melalui jaringan, dan objek server akan mengirimkan hasil panggilan metode kembali ke objek client.

Pada dasarnya, RMI menggunakan protokol RPC (Remote Procedure Call) untuk melakukan panggilan metode antar-proses. Dalam proses ini, objek client mengirimkan permintaan panggilan metode ke objek server, yang kemudian mengirimkan hasil panggilan metode kembali ke objek client[3]. RMI memungkinkan programmer untuk menangani proses ini dengan cara yang lebih efektif dan efisien, karena objek server dapat mengirimkan hasil panggilan metode ke objek client tanpa perlu mengirimkan data secara langsung.

Java RMI telah digunakan dalam berbagai aplikasi berbasis jaringan, termasuk aplikasi perbankan, sistem manajemen, dan aplikasi berbasis web. Java RMI memungkinkan programmer untuk membangun aplikasi yang dapat bekerja dengan objek di dalam sistem berbeda, memungkinkan aplikasi yang lebih efektif dan efisien.

BAB II

ISI

A. Sejarah Perkembangan Java RMI

Sejarah perkembangan Java Remote Method Invocation (RMI) dimulai sejak Java pertama kali diperkenalkan oleh Sun Microsystems (sekarang Oracle Corporation) pada tahun 1995[5]. Java Remote Method Invocation (RMI) adalah teknologi untuk memungkinkan objek Java di dalam sistem berbeda untuk mengirimkan panggilan metode (invocation) ke objek Java di sistem lain. Java RMI telah mengalami beberapa evolusi dan peningkatan sejak saat itu. Berikut adalah evolusi Java RMI dari awal hingga saat ini:

1. **Java 1.1 (1997) :** Java RMI pertama kali diperkenalkan dalam Java Development Kit (JDK) 1.1. Pada versi ini, RMI digunakan untuk memungkinkan objek Java berkomunikasi satu sama lain melalui jaringan dengan menggunakan protokol Remote Procedure Call (RPC).
2. **Java 2 Platform, Standard Edition (J2SE) 1.2 (1998) :** Pada versi ini, RMI mengalami beberapa peningkatan performa dan peningkatan keamanan. Sun Microsystems memperkenalkan dukungan untuk objek-objek yang lebih kompleks, seperti objek yang dapat digunakan kembali dan objek yang mengandung referensi ke objek lain.
3. **Java 2 Platform, Enterprise Edition (J2EE) 1.2 (1999) :** Java RMI menjadi bagian integral dari J2EE, yang dirancang untuk memfasilitasi pengembangan aplikasi enterprise yang kompleks. Pada versi ini, dukungan untuk komponen-komponen Enterprise JavaBeans (EJB) yang menggunakan RMI sebagai mekanisme panggilan jarak jauh diperkenalkan[6].
4. **Java 2 Platform, Enterprise Edition (J2EE) 1.3 (2001) :** Pada versi ini, RMI mengalami peningkatan dalam kinerja dan kemampuan skalabilitas. Dukungan untuk pengelolaan objek-objek yang lebih besar dan kompleks diperbaiki.
5. **Java 2 Platform, Standard Edition (J2SE) 1.4 (2002) :** Dalam JDK 1.4, RMI memperoleh dukungan untuk pengiriman objek yang dioptimalkan. Peningkatan keamanan juga diperkenalkan, termasuk dukungan untuk sertifikat digital dan enkripsi.

6. **Java 2 Platform, Enterprise Edition (J2EE) 1.4 (2003) :** Pada versi ini, RMI diperbarui dengan fitur-fitur baru untuk mendukung pengembangan aplikasi enterprise yang lebih kompleks dan efisien.
7. **Java Platform, Standard Edition (Java SE) 6 (2006) :** Dalam Java SE 6, ada beberapa perbaikan kecil dalam RMI untuk meningkatkan performa dan keamanan. Pada versi ini dukungan untuk SSL (Secure Sockets Layer) diperkenalkan, yang memungkinkan RMI untuk beroperasi secara aman di atas jaringan yang tidak aman.
8. **Java SE 7 (2011) :** Java SE 7 membawa beberapa perbaikan kecil dalam RMI, tetapi tidak ada peningkatan besar dalam arsitektur atau fitur.
9. **Java SE 8 (2014) :** Java SE 8 juga tidak memiliki perubahan yang signifikan dalam Java RMI.
10. **Java SE 9, 10, 11, 12, 13, 14, 15, dan seterusnya:** Pada versi-versi Java selanjutnya setelah Java SE 8, tidak ada perubahan besar dalam Java RMI yang dicatat secara khusus. Peningkatan keamanan, kinerja, dan stabilitas mungkin telah dilakukan, tetapi tidak ada perubahan signifikan dalam arsitektur atau fitur yang dilaporkan secara luas.

Selama perkembangannya, Java RMI telah mengalami perubahan dalam arsitektur dan penambahan fitur untuk meningkatkan kinerja, keamanan, dan fleksibilitas. Beberapa perubahan dan penambahan fitur yang signifikan termasuk:

- Peningkatan Keamanan: Dengan penambahan fitur-fitur keamanan seperti penggunaan sertifikat digital, keamanan transport layer, dan manajemen izin akses, Java RMI telah menjadi lebih aman dari versi awalnya.
- Peningkatan Performa: Perbaikan performa dan peningkatan dalam manajemen koneksi jaringan telah diperkenalkan dalam versi-versi terbaru untuk mengoptimalkan kinerja RMI.
- Dukungan SSL: Perkenalan dukungan untuk SSL telah meningkatkan keamanan komunikasi RMI di atas jaringan yang tidak aman.

Dengan evolusi ini, Java RMI terus berkembang untuk tetap relevan dalam lingkungan komputasi terdistribusi yang terus berubah, dan memberikan pengembang dengan alat yang lebih kuat dan aman untuk membangun aplikasi terdistribusi di platform Java.

B. Arsitektur Java RMI

Java RMI (Remote Method Invocation) memungkinkan pemanggilan metode dari objek yang berjalan di mesin virtual Java (JVM) lain, memungkinkan komunikasi antara objek-objek yang berjalan di mesin yang berbeda. Arsitektur Java RMI terdiri dari beberapa komponen utama yang bekerja sama untuk memungkinkan komunikasi antara client dan server.

a. Komponen Utama Java RMI

1. Remote Object: Objek yang berada di server dan dapat dipanggil dari mesin yang berbeda. Objek ini harus mengimplementasikan interface java.rmi.Remote dan metode yang dapat dipanggil dari jarak jauh[4].
2. Stub: Stub adalah representasi lokal dari objek yang berada di mesin yang berbeda (server). Ketika client ingin memanggil metode pada objek yang berada di server, ia sebenarnya memanggil stub lokal ini.
3. Registry: Registry adalah layanan direktori yang memetakan referensi objek ke lokasi fisik mereka di mesin server. Objek-objek yang siap dipanggil dari jarak jauh harus didaftarkan di registry.
4. Transporter: Transporter bertanggung jawab untuk mentransfer parameter dan hasil pemanggilan metode antara client dan server. Ini melibatkan serialisasi objek, transfer data melalui jaringan, dan deserialisasi objek.

b. Komunikasi antara Komponen-Komponen dalam Implementasi Java RMI

1. Registrasi Objek: Objek yang siap dipanggil dari jarak jauh harus didaftarkan di registry menggunakan *'java.rmi.registry.LocateRegistry.createRegistry() atau java.rmi.Naming.rebind()'*.
2. Pemanggilan Metode dari Client: Ketika client ingin memanggil metode pada objek yang berada di server, ia menggunakan *`java.rmi.Naming.lookup()`* untuk mendapatkan referensi objek dari registry. Referensi ini sebenarnya adalah stub lokal yang merepresentasikan objek di server.
3. Pemanggilan Metode dari Stub: Ketika client memanggil metode pada stub, stub ini mengatur transfer pemanggilan tersebut melalui jaringan ke objek yang sebenarnya di server.
4. Transportasi dan Eksekusi: Parameter pemanggilan metode diserialisasi dan dikirim melalui jaringan ke objek di server. Objek di server menerima

pemanggilan, menjalankan metode yang diminta, dan mengembalikan hasilnya.

5. Pengiriman Balik Hasil: Hasil dari pemanggilan metode, beserta objek-objek yang mungkin disertakan, diserialisasi dan dikirim kembali ke stub di client. Stub di client kemudian mengembalikan hasilnya kepada pemanggil asli.

Melalui langkah-langkah ini, komunikasi terjadi antara komponen-komponen Java RMI, memungkinkan pemanggilan metode dari jarak jauh dengan transparan seolah-olah objek tersebut berada di mesin lokal.

C. Implementasi Java RMI Pada Aplikasi Remote Calculator

Java RMI dapat diimplementasikan pada banyak aplikasi. Berikut adalah salah satu pengimplementasian aplikasi remote calculator sederhana menggunakan Java RMI yang memungkinkan client untuk memanggil metode pada remote objek, dalam hal ini, sebuah kalkulator sederhana yang berada di server.

Langkah-langkah Implementasi Java RMI

Langkah 1: Pembuatan Antarmuka Remote Objek

Buat sebuah antarmuka yang menggambarkan metode yang akan dipanggil dari jarak jauh. Misalnya, *CalculatorInterface.java*:

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

public interface CalculatorInterface extends Remote {
    int add(int a, int b) throws RemoteException;
    int subtract(int a, int b) throws RemoteException;
}
```

Langkah 2: Implementasi Remote Objek

Buat kelas yang mengimplementasikan antarmuka remote objek. Misalnya, `*CalculatorImpl.java*`:

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

```
public class CalculatorImpl extends UnicastRemoteObject implements CalculatorInterface {
    protected CalculatorImpl() throws RemoteException {
        super();
    }

    public int add(int a, int b) throws RemoteException {
        return a + b;
    }

    public int subtract(int a, int b) throws RemoteException {
        return a - b;
    }
}
```

Langkah 3: Pembuatan Server

Buat kelas server yang akan membuat objek kalkulator dan mendaftarkannya di registry. Misalnya, `*CalculatorServer.java*`:

```
import java.rmi.Naming;

public class CalculatorServer {
    public static void main(String[] args) {
        try {
            CalculatorInterface calculator = new CalculatorImpl();
            Naming.rebind("CalculatorService", calculator);
            System.out.println("Calculator server is ready.");
        } catch (Exception e) {
            System.out.println("Calculator server failed: " + e);
        }
    }
}
```

Langkah 4: Pembuatan Client

Buat kelas client yang akan mencari objek kalkulator di registry dan memanggil metodenya. Misalnya, `*CalculatorClient.java*`:

```
import java.rmi.Naming;

public class CalculatorClient {
    public static void main(String[] args) {
        try {
            CalculatorInterface calculator = (CalculatorInterface)
Naming.lookup("rmi://localhost/CalculatorService");
            int a = 10;
            int b = 5;
            System.out.println("Addition result: " + calculator.add(a, b));
            System.out.println("Subtraction result: " + calculator.subtract(a, b));
        } catch (Exception e) {
            System.out.println("Calculator client failed: " + e);
        }
    }
}
```

Menjalankan Aplikasi :

1. Compile semua file java dengan perintah `javac *.java`.
2. Jalankan `rmiregistry` untuk memulai registry.
3. Jalankan server dengan java `CalculatorServer`.
4. Jalankan client dengan java `CalculatorClient`.

Setelah langkah-langkah tersebut dilakukan, client akan terhubung ke server menggunakan Java RMI, memanggil metode pada objek kalkulator yang berada di server, dan mencetak hasilnya ke konsol.

Dengan langkah-langkah di atas, kita telah berhasil mengimplementasikan Java RMI untuk membuat aplikasi sederhana yang memanfaatkan komunikasi antara objek-objek di mesin yang berbeda menggunakan Java RMI.

BAB III

PENUTUP

A. KESIMPULAN

Kesimpulan dari "Advancements in Java RMI: Enhancing Scalability and Performance through Modern Implementations" adalah bahwa dengan memanfaatkan kemajuan dalam teknologi Java RMI dan menerapkan implementasi modern, dapat meningkatkan skalabilitas dan kinerja sistem secara signifikan. Dengan pendekatan yang tepat, perusahaan dapat memperluas kapabilitas sistem mereka dan meningkatkan efisiensi operasional, sehingga dapat bersaing lebih baik di pasar yang terus berubah dan memenuhi tuntutan pengguna yang semakin tinggi.

DAFTAR PUSTAKA

[1] Adrian Nathaniel Wikana. 2024. Implementasi Remote Method Invocation (Rmi) Untuk Tes Online Interaktif Multiuser Pada Local Area Network(Lan) from neliti.com website: [Online]. Available: https://www.neliti.com/id/publications/68812/implementasi-remote-method-invocation-rmi-untuk-tes-online-interaktif-multiuser

[2] Java. (2015). Retrieved March 4, 2024, from SlideShare website: [Online]. Available: https://www.slideshare.net/agus248/java-43317164

[2] Sri Lestari. 2011. Client/Server dengan Java Remote Method Invocation (Java RMI): Sebuah Tutorial. Retrieved March 4, 2024, from ResearchGate website: [Online]. Available: https://www.researchgate.net/publication/279500548_ClientServer_dengan_Java_Remote_Med_Invocation_Java_RMI_Sebuah_Tutorial

[3] Aakash Ojha. 2023. Remote Method Invocation in Java - GeeksforGeeks. Retrieved March 4, 2024, from GeeksforGeeks website: [Online]. Available: https://www.geeksforgeeks.org/remote-method-invocation-in-java/

[4] Nandri Marsan Sitinjak, Frans Ikorasaki. 2023. Implementasi Remote Method Invocation (RMI) pada Sistem Pemesanan Tiket Kereta Api Berbasis Web. Publikasi Kegiatan Pengabdian Masyarakat Widya (PUNDIMASWID), [Online]. Vol 2 No 1. Available: https://www.amikwidyaloka.ac.id

[5] Unknown Author. (Year not provided). Microsoft Word - hardcover_skripsi. Retrieved March 4, 2024, from Binus University website: [Online]. Available: Microsoft Word - hardcover_skripsi (binus.ac.id)

[6] Oracle Corporation. (Year not provided). Getting Started Using Java RMI. Retrieved March 4, 2024, from Oracle website: [Online]. Available: Getting Started Using Java RMI (oracle.com)