

Verteilte Systeme

Prof. Dr. Martin Becke

CaDS - HAW Hamburg

Version 0.9

Inhalt

1 RPC

- Remote Procedure Call
- Remote Method Invocation

2 Chord

Remote Procedure Call

Idee

- ▶ Ein Kommunikationsmodell
- ▶ Prozeduraufrufe zwischen Prozessen auf unterschiedlichen Systemen oder Maschinen
- ▶ Entfernte Funktionen oder Methoden aufrufen wie lokale

Remote Procedure Call

Ablauf

- 1 Der Client ruft die entfernte Prozedur über den Client Stub auf, als wäre es eine lokale Funktion.
- 2 Der Client Stub wandelt die Parameter der Anfrage in ein standardisiertes Format um (Marshalling) und sendet die Anfrage über das Kommunikationsprotokoll an den Server.
- 3 Der Server Stub empfängt die Anfrage, demarshalled die Parameter und ruft die entsprechende Funktion auf dem Server auf.
- 4 Die Funktion wird auf dem Server ausgeführt, und das Ergebnis wird an den Server Stub zurückgegeben.
- 5 Der Server Stub marshallt das Ergebnis und sendet es über das Kommunikationsprotokoll an den Client Stub zurück.
- 6 Der Client Stub empfängt die Antwort, demarshalled das Ergebnis und gibt es an den Client zurück.

Remote Procedure Call

Naiver Entwurf

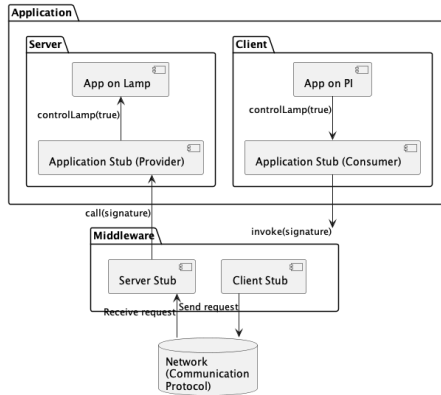


Figure – Erster Ansatz einer RPC Architektur

Remote Procedure Call

Naiver Code I

```
public interface Lamp {  
    void controllamp(Boolean b);  
}
```

Listing 1 – RPC Interface

Remote Procedure Call

Naiver Code II

```
public class LampRemote implements Lamp {  
    @Override  
    public void controllLamp(boolean b){  
        List<Boolean> list = new ArrayList<Boolean>();  
        Collections.addAll(list, b);  
        Middleware.invoke("controllLamp", list);  
    }  
}
```

Listing 2 – RPC Remote Implementation

Remote Procedure Call

Naiver Code III

```
public class LampFactory {  
    public static Lamp createLamp() {  
        // Hier koennte die Middleware-Verbindung  
        hergestellt werden.  
        // In diesem Beispiel verwenden wir eine  
        einfache lokale Implementierung.  
        return new LampRemote();  
    }  
}
```

Listing 3 – Application Stub mit Factory Pattern

Remote Procedure Call

Naiver Code IV

```
public class LampController {  
    public static void main(String[] args) {  
        // Erstelle eine Lampe ueber die Factory-Methode  
        // (Application-Stub)  
        Lamp lamp = LampFactory.createLamp();  
  
        // Verwende die Lampe in der Anwendung  
        lamp.controlLamp(true);  
    }  
}
```

Listing 4 – Controller ruft Middleware

Remote Procedure Call

Marshalling Process - Beispiel

- ▶ Konvertierung Parameter (z.B. String und Integer) in ein plattformunabhängiges Format, z. B. Byte-Arrays.
- ▶ Hinzufügen von Metadaten, wie Typinformationen, zur Identifizierung der Parameter auf der Empfängerseite.
- ▶ Zusammenpacken der Byte-Arrays und Metadaten in einer Nachricht, die über das Netzwerk gesendet werden kann

Remote Procedure Call

Copy in, Copy out

- ▶ Semantik zur Übergabe von Parametern
- ▶ Kopiert Parameter in eine Nachricht (Ohne Kenntnis der Daten)
- ▶ Benötigt ein generisches Regelwerk für Datentypen
- ▶ Alternative : Call by Reference - Direkter Speicherzugriff fremder Systeme

Remote Procedure Call

Copy in, Copy out Optimierung

- ▶ Datenkompression
- ▶ Effiziente Serialisierungsformate (Beispiel MessagePack)
- ▶ Selektive Übertragung

Remote Procedure Call

Interface Definition Language (IDL)

- ▶ IDL beschreibt die Signatur der entfernten Funktionen
- ▶ Generierung des Stub- und Skeleton-Code aus der IDL-Definition
- ▶ Kann mit Code-Generator verbunden werden.

Remote Procedure Call

IDL Example

```
interface MathService {  
    int add(in int a, in int b);  
}
```

Listing 5 – IDL Example

Remote Procedure Call

RPC IDL Service Impl

```
class MathServiceImpl(MathService):  
    def add(self, a, b):  
        return a + b
```

Listing 6 – RPC IDL Service Impl

Remote Procedure Call

RPC IDL Service Impl

```
math math_service_client =  
    MathServiceStub(server_address)  
result = math_service_client.add(5, 7)  
print("Result of 5 + 7:", result)
```

Listing 7 – RPC IDL Service Impl

Remote Procedure Call

Framework Beispiele

- ▶ XML-RPC <http://xmlrpc.com/>
- ▶ JSON-RPC <https://www.jsonrpc.org/>
- ▶ gRPC <https://grpc.io/>

Remote Procedure Call

Beispiele XML-RPC

```
from xmlrpc.server import SimpleXMLRPCServer

def add(a, b):
    return a + b

server = SimpleXMLRPCServer(("localhost", 8080))
server.register_function(add, "add")
server.serve_forever()
```

Listing 8 – XML-RPC in Python

Remote Procedure Call

Server-Stub, Skeleton und Server-Proxy

- ▶ Skeleton ist die reine Funktion
- ▶ Server-Stub ist das Skeleton aus der IDL abgeleitet
- ▶ Server-Proxy übernimmt weitere Aufgaben wie diskutiert

Remote Method Invocation (RMI)

Idee

- ▶ Speziell für eine Objekt-orientierte Programmiersprache
- ▶ Proprietäre Protokolle
- ▶ In Java ist RMI eingebettet (Beispiel Script)

Inhalt

1 RPC

- Remote Procedure Call
- Remote Method Invocation

2 Chord

P2P Chord

Idee

- ▶ Chord-Architektur ist ein verteiltes Hashtabelle-System
- ▶ Basiert auf strukturierten Overlay-Netzwerk
- ▶ Bekannt seit 2001
- ▶ Grundlage für die Entwicklung von Blockchain-Technologien
- ▶ Grundlage dezentralisierten autonomen Organisationen (DAOs)

P2P Chord

Struktur und Funktionsweise

- ▶ Identifier Space : Default Ring
- ▶ Key-Node Mapping
- ▶ Finger Tables
- ▶ Lookup-Operationen
- ▶ Dynamik und Fehlertoleranz

P2P Chord

Einsatzmöglichkeiten

- ▶ Dateifreigabe
- ▶ Distributed Domain Name System (DDNS)
- ▶ Content Distribution Networks (CDN)
- ▶ Distributed Databases
- ▶ etc

P2P Chord

Beispiel

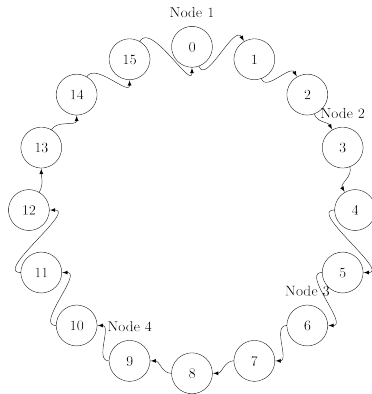


Figure – Chord Setup

P2P Chord

Fingertable

- ▶ $\log(n)$ Einträge
- ▶ $(n + 2^{(i-1)}) \bmod 2^m$
- ▶ Lässt schnelle Suche zu (im Vergleich zur linearen)

P2P Chord

Cassandra

- ▶ NoSQL-Datenbanksystem
- ▶ Partitionierungsstrategie
 - ▶ Range-based Sharding
 - ▶ Hash-based Sharding
- ▶ Replikationsstrategie
- ▶ Datenmodell
- ▶ Gossip-Protokoll (statt Fingertable für Routing-Aufbau)
- ▶ Hinted Handoff
- ▶ Repair und Anti-Entropy