UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 3
Musterlösung

Vorlesung Verteilte Systeme
Sommersemester 2020

Aufgabe 1: RPC und Unions

Falls das Laufzeitsystem nicht erkennt, welcher Wertetyp sich in dem Feld befindet, kann es ihn nicht korrekt verpacken. Damit können Unions in einem RPC-System nicht toleriert werden, es sei dann, es gibt ein kennzeichnendes Feld, das eindeutig beschreibt, was in dem varianten Feld enthalten ist. Das kennzeichnende Feld darf nicht der Kontrolle des Benutzers unterliegen.

Aufgabe 2: Entfernte Objektreferenzen

Unter Verwendung von Java können wir eine solche Implementierung z.B. durch die folgende Klasse ausdrücken:

```
public class Object_reference {
  InetAddress server_address;     // Netzwerkadresse des Objekt-Servers
  int server_port;                // Port, der Server idnetifiziert
  int object_id;                  // Identifier für dieses Objekt
  URL client_code;                // (entfernte) Datei, die den Code des
                                  // Client-Stubs enthält
  byte[] init_data;               // Möglicherweise zusätzliche
                                  // Initialisierungsdaten für den Stub
}
```

Die Objektreferenz sollte mindestens die Transportschichtadresse des Servers enthalten, auf dem sich das Objekt befindet, hier ebenso auch den Port. Außerdem brauchen wir eine Objekt-ID, weil der Server mehrere Objekte enthalten kann. In unserer Implementierung verwenden wir eine URL, um auf eine (entfernte) Datei zu verweisen, die den gesamten erforderlichen Client-seitigen Code enthält. Eine alternative Implementierung wäre es, direkt einen serialisierten Client-Stub als Objektreferenz zu verwenden. Dieser Ansatz wird beispielsweise in Java RMI befolgt, wo Proxies als Referenz übergeben werden.

Aufgabe 3: Exceptions

Weil die Exceptions ursprünglich auf der Server-Seite geworfen werden, kann der Server-Stub die Ausnahme nur abfangen, als spezielle Fehlerantwort verpacken und an den Client zurückgeben. Der Client-Stub dagegen muss die Nachricht entpacken und die selbe Exception werfen, um den Zugriff auf den Server transparent zu halten. Demzufolge müssen auch die Ausnahmen in einer Schnittstellendefinitionssprache beschrieben werden.

Aufgabe 4: Threads

a) Gesamtzeit = Berechnung der Argumente + Marshalling Argumente + Bearbeitungszeit Betriebssystem Senden + Netzwerkzeit für Übertragung + Bearbeitungszeit Betriebssystem Empfangen + Unmarshalling Argumente +

Verarbeitungszeit Server + Marshalling Ergebnisse + Bearbeitungszeit Betriebssystem Senden + Netzwerkzeit für Übertragung + Bearbeitungszeit Betriebssystem Empfangen + Unmarshalling Argumente

= 5 + 4 * Marshalling/Unmarshalling + 4* Bearbeitungszeit Betriebssystem + 2 * Netzwerkzeit für Übertragung + Verarbeitungszeit Server
= 5 + 4* 0,5 + 4* 0,5 + 2 * 3 + 10ms = 25ms

Für zwei Anfragen benötigt der Client also 2*25ms = 50ms.

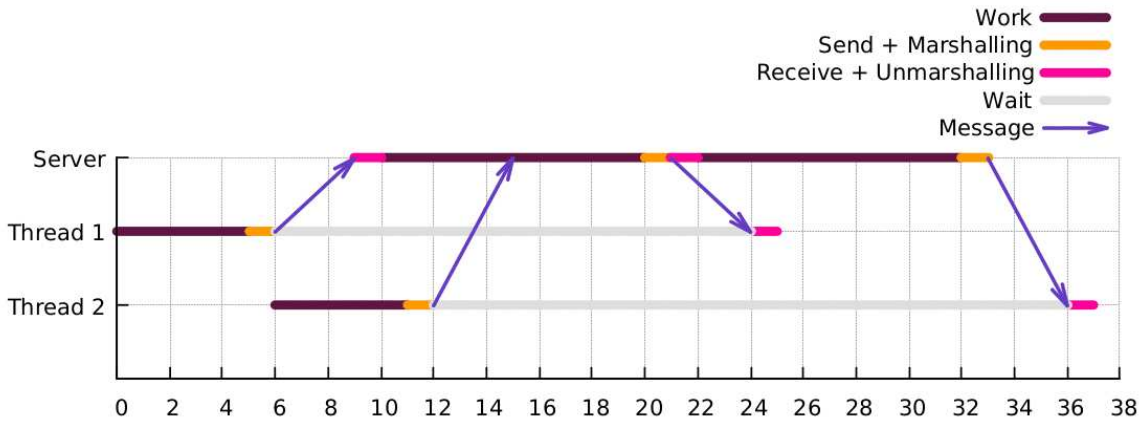
b) Client:
1. Anfrage: Berechnung der Argumente + Marshalling Argumente + Bearbeitungszeit Betriebssystem Senden = 5 + 0,5 + 0,5 = 6, dann
2. Anfrage: Berechnung der Argumente + Marshalling Argumente + Bearbeitungszeit Betriebssystem Senden = 5 + 0,5 + 0,5 = 6 = 12ms, danach wird auf Antwort vom 1. Aufruf gewartet

Server:
Die 1. Anfrage erreicht den Server-Rechner nach 6 + 3 = 9 ms.
Es folgt Bearbeitungszeit Betriebssystem Empfangen + Unmarshalling Argumente = 0,5 + 0,5 = 1ms, so dass der Server-Prozess die Anfrage 10ms nach dem Start erhält.
Es folgen 10ms Verarbeitungszeit des Servers sowie je 0,5ms für Marshalling der Ergebnisse und Bearbeitungszeit Betriebssystem Senden, daher wird die Antwort nach 21ms verschickt. Der Server-Rechner erhält die 2. Anfrage in der Zwischenzeit, jedoch verarbeitet er sie erst nach den 21ms, also sendet er die Antwort wieder 12ms später (je 0,5ms für Bearbeitungszeit Betriebssystem Empfangen + Unmarshalling Argumente, 10ms Verarbeitungszeit, je 0,5ms für Marshalling der Ergebnisse und Bearbeitungszeit Betriebssystem Senden) , d.h. 33ms nach dem Start.

Client:
Der Client erhält die Antworten 3 + 1 = 4ms später.
(Netzwerkzeit für Übertragung + Bearbeitungszeit Betriebssystem Empfangen + Unmarshal Argumente)

Damit beträgt die Gesamtzeit 37ms.

Der Ablauf ist hier nochmals als Gantt-Diagramm dargestellt:



Aufgabe 5: RMI- Grundlagen und Architektur

Siehe Kap. 2 und 3 der Vorlesung.

Aufgabe 6: Programmierung: Java-RMI - Hello World!

Die Lösung dieser Aufgabe finden Sie im Archiv `l03Files.zip`[1] auf der Vorlesungswebseite.

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/l03Files.zip`