JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Ausarbeitung

DezSys

Systemtechnik-Matura
5BHIT 2015/16

Michael Weinberger

Version 1.0
Begonnen am 21. April 2016
Beendet am 16. Mai 2016

Betreuer: Graf/Borko

Inhaltsverzeichnis

Kompetenzen für Dezentrale Systeme

- **Lastenverteilung auf Applikationsebene**
  'können Lastverteilung auf Applikationsebene realisieren'
- **Sicherheitskonzepte**
  'können Sicherheitskonzepte für verteilte, dezentrale Systeme entwickeln'
- **Durchführung von Transaktionen in verteilten Systemen**
  'können in dezentralen Systemen Transaktionen durchführen'
- **Programmiertechniken zur Realisierung von entfernten Prozeduren, Methoden und Objekten**
  'können Programmiertechniken in verteilten Systemen zur Realisierung von entfernten Prozeduren, Methoden und Objekten anwenden sowie webbasierte Dienste und Messaging-Dienste in solchen Systemen implementieren'

1 Cloud Computing und Internet of Things

1.1 Einführung

1.1.1 Was versteht man unter Cloud Computing?

Heutzutage setzen viele Hersteller auf Cloud Computing und bieten dementsprechende Plattformen an, der Trend geht immer mehr in diese Richtung.

Konkret geht es bei Cloud Computing um die Auslagerung von Anwendungen, Daten und Rechenvorgängen ins Web.

Dies könnte zum Beispiel die Auslagerung von Bürosoftware wie Tabellenkalkulation, Textverarbeitung oder CRM-Systemen in die Cloud sein. Diese Auslagerung bietet einige Vorteile. Die Synchronisation zwischen mehreren Rechnern wird nicht mehr relevant und gemeinsame Arbeit an Dokumenten durch die zentrale Ablage vereinfacht. Für das Absichern der Datensätze ist die Cloud Computing-Plattform und die Anwendung selbst verantwortlich. Natürlich ist auch der Datenspeicher in der 'Wolke' begrenzt, jedoch in jeder Hinsicht größer als der eines einzelnen Rechners oder Festplattenverbundes im normalen Stil. Verglichen mit traditionellen Systemumgebungen sind Cloud Computing-Plattformen wesentlich einfacher zu verwalten. Der Grund dafür ist der hohe Abstraktionsgrad der Plattformen, denn um typische Administrationsaufgaben wie Load Balancing oder Serverwartung kümmert sich der Anbieter. Bei Rechenvorgängen ist der Vorteil einer besseren Skalierbarkeit gegeben, dass man auf einen großen Pool von Instanzen zurückgreifen kann. Dank Cloud Computing und dessen hoher Flexibilität kann man diese Server beispielsweise auch für wenige Stunden oder Tage, in denen sie benötigt werden, *on demand* mieten und somit Betriebskosten einsparen. Die Bereitstellung erfolgt innerhalb von Minuten, und kommt ohne komplexe Verträge aus.[1]

1.1.2 Was versteht man unter IoT?

Das Internet der Dinge (Internet of Things / IoT) ist ein Gebilde, bei dem Objekte, Tiere oder Menschen mit einem einzigartigen Identifikator ausgestattet sind. Weiterhin ist damit die Möglichkeit verbunden, Daten über ein Netzwerk ohne Interaktionen Mensch-zu-Mensch oder Mensch-zu-Computer zu übertragen. Ein Ding im Internet der Dinge kann zum Beispiel eine Person mit einem Herzschrittmacher, ein Nutztier auf einem Bauernhof mit einem Biochip-Transponder oder ein Automobil mit eingebauten Sensoren sein. Letzteres könnte eine Warnung auslösen, wenn der Reifendruck zu niedrig ist. Im Prinzip ist jedes vom Menschen geschaffene Objekt ein Kandidat, das sich mit einer IP-Adresse ausstatten lässt und Daten via Netzwerk übertragen kann. Bisher wurde das Internet der Dinge am häufigsten mit Maschine-zu-Maschine-Kommunikation bei einer Fertigungsstraße in Verbindung gebracht. Sind Produkte mit M2M-Kommunikation ausgestattet, werden sie häufig als *intelligent* oder *smart* bezeichnet. Der durch IPv6 wesentlich größere Adressraum ist ein wichtiger Faktor bei der Entwicklung des Internets der Dinge. Durch die wachsende Anzahl an intelligenten Knoten (Nodes) erwartet man, dass es neue Bedenken für Privatsphäre, Datenhoheit und Sicherheit gibt. [2]

1.2 Realisierung von entfernten Prozeduren, Methoden, Objekten zur Interkommunikation

Dieses Kapitel bezieht sich auf die Implementierungen für die Programmiersprache Java.

1.2.1 Interprozesskommunikation

Interprozesskommunikation (IPC) ist eine Sammlung von Programmierinterfaces, die einem erlauben koordinierte Aktivitäten zwischen verschiedenen Prozessen nebenläufig laufen zu lassen. Dies ermöglicht etwa, dass ein Programm viele Benutzeranfragen gleichzeitig verarbeiten kann. Jede einzelne Useranfrage kann auch mehrere Prozesse generieren, die allesamt miteinander kommunzieren müssen. IPC nimmt sich dieser Problematik an. Jede einzelne Umsetzungsmethode hat seine eigenen Vor- und Nachteile, so ist es nicht selten, dass auch mehrere Methoden gleichzeitig zum Einsatz kommen. [3]

1.2.2 Wie werden Informationen ausgetauscht?

In der Interprozesskommunikation empfiehlt es sich, ein 'Protokoll' zu definieren, sprich, wie Nachrichten untereinander weitergegeben werden. Beispielsweise ist SOAP ein standardisiertes Verpackungsprotokoll für Nachrichten. Die Spezifikation definiert einen XML-basierten Umschlag (Envelope) für die zu übertragenden Informationen sowie Regeln für die Umsetzung von Anwendungs- und plattformspezifischen Datentypen in XML-Darstellung. Der Nachrichtenaustausch über XML stellt eine flexible Methode zur Kommunikation zwischen Anwendungen dar. Eine Nachricht kann alles mögliche enthalten z.B. Warenbestellungen, Suchanfragen oder ähnliches. Da XML an keine bestimmte Programmiersprache oder ein bestimmtes Betriebssystem gebunden ist, können XML-Nachrichten in allen Umgebungen verwendet werden. Aufgrund der Einfachheit empfiehlt sich in Java der Einsatz sogenannter POJOs. Das sind schlicht und einfach 'plain old java objects' mit Methoden und Attributen. Dieses Objekt kann in verteilten Systemen serialisiert übertragen werden. [4]

1.2.3 Sockets

Sockets sind wahrscheinlich die einfachste Möglichkeit, um Prozesse 'miteinander reden zu lassen'. Ein Socket ist ein Kommunikationsendpunkt, und wird über eine IP-Adresse und einen Port eindeutig identifiziert. Er nimmt eingehende Verbindungen entgegen, und verbindet sich mit seinem Gegenstück. Unter einer IP-Adresse können mehrere Sockets erreichbar sein! Sowohl Server als auch Client benötigen einen eindeutigen Kommunikationsendpunkt, wir unterscheiden zwischen Server-Socket und Client-Socket. Der Server-Socket wartet und 'horcht' auf eingehende Verbindungen durch Clients, wird daher auch passiver Socket genannt. Er muss einer IP-Adresse und einem Port zugeteilt sein ('Binding'). Er empfängt Daten von Clients, führt Berechnungen durch und sendet das Ergebnis zurück, der Aufbau geschieht typischerweise multithreaded. Der Client-Socket verbindet sich mit Server-Socket und initiiert den Verbindungsaufbau, wird daher aktiver Socket genannt. Er muss IP und Port des Servers wissen, und sendet single- oder multithreaded Daten an den Server und erhält Resultate. [5]

1.2.4 RPC

Der Remote Procedure Call (RPC) ist ein Protokoll, das die Implementierung verteilter Anwendungen - also Netzwerkdienste - vereinfachen soll. Die dahinter steckende Idee ist, dass ein Programm eine Funktion eines Programms, das auf einem anderen Rechner läuft, nutzen kann, ohne sich um die zu Grunde liegenden Netzwerkdetails kümmern zu müssen. Der RPC arbeitet nach dem Client-Server-Modell. Ein RPC-Aufruf arbeitet in den meisten Fällen synchron. Das lokale Programm, der Client, sendet eine Anforderung an das entfernte Programm, den Server, und unterbricht seine Arbeit bis zum Eintreffen der Antwort. In Verbindung mit Threads ist allerdings eine asynchrone Realisierung eines entfernten Funktionsaufrufs möglich. Die Programmierung eines Programms, das RPC-Aufrufe verwendet, gestaltet sich recht einfach. In dem der Sprache C sehr ähnlichen Programmcode wird eine so genannte Stub-Routine verwendet, die als Platzhalter für den kompletten Code zur Realisierung des Netzwerkzugriffs dient. Später wird mit Hilfe des Programms rpcgen aus der Datei ein vollständiges C-Programm erzeugt. [6]

1.2.5 Java RMI

Während RPC C-basiert ist, liegt bei RMI (Remote Method Invocation) der Fokus voll und ganz auf Java, und setzt dieses objektorientiert um. Objekte, die sich auf unterschiedlichen Rechnern befinden, können mit Hilfe von RMI über Methodenaufrufe miteinander kommunizieren. Das Prinzip von RMI ist denkbar einfach: Ein Client-Objekt sendet dabei eine Nachricht an den Server, die die aufzurufende Methode sowie die dafür benötigten Parameter enthält. Das Server-Objekt führt die entsprechende Methode aus und schickt das Resultat wieder an den Client zurück. Zur Realisierung dieses Konzepts kapselt Java die Daten, die über das Netzwerk übermittelt werden sollen, auf dem Client-Rechner in sogenannten 'Stubs'. Die Parameter müssen dafür zuerst in einem passenden Format zusammengefasst werden. Komplizierter ist diese Aktion bei Objekten, beispielsweise bei Strings oder eigenen Klassen. Da Objektreferenzen im Grunde nichts anderes sind als Zeiger auf bestimmte lokale Speicherstellen, kann der Server damit natürlich nicht viel anfangen. Es muß also das komplette Objekt übermittelt werden, wozu Object Serialization, also der gleiche Mechanismus verwendet wird, der mit dem auch Objekte beziehungsweise Objektreferenzen auf einer Diskette oder Festplatte gespeichert werden. Um das zu ermöglichen, müssen Objekte, die als Parameter für RMI-Aufrufe dienen, das Interface 'Serializable' aus dem Package 'java.io' implementieren. [7]

1.2.6 CORBA

1.3 Grundlagen Messaging-Dienste

mom, ...

1.4 Anwendung mit webbasierten Diensten

JEE, REST Grundlagen, Frameworks, ...

2 Automatisierung, Regelung und Steuerung

Themengebiet wird ausgelassen (1 von 1)

3 Security, Safety, Availability

3.1 Grundlegende Sicherheitskonzepte

Intrusion Detection, Honey Pot, Application Firewall, ... sowie Unterschiede

3.2 Risiken & Gefahren von dezentralen Systemen

Beschreibung und Lösungsansätze

3.3 Frameworks

Funktionsweisen, verschiedene Ansätze

4 Authentication, Authorization, Accounting

4.1 Beschreibung der Grundlagen

Auth, Aut, Audit

4.2 Was ist LDAP?

Erklärung der Funktionsweise

4.3 Benutzerverwaltung mit LDAP

Serverarten, Zugriff darauf

4.4 Möglichkeiten zur Implementierung/alternative verteilte Authentifizierungsdienste

KDC, kerberos, Single Sign On, ...

5 Disaster Recovery

5.1 Backup-Strategien

Häufigkeit, wo werden sie aufbewahrt, sicherheitslevels

5.2 Disaster Recovery Plan

Aufstellen, was sollte drin sein, ...

5.3 Best Practice für die vorgegebenen Anforderungen

hot standby, cold standby, cluster

5.4 Rollback

Wie wird zentrales Image verteilt?

6 Algorithmen und Protokolle

6.1 Techniken zur Prüfung und Erhöhung der Sicherheit von dezentralen Systemen

Vorgehensweisen, symm. Verschlüsselung, SSL/TLS-Protokoll, ...

6.2 Grundlagen Lastverteilung

Hervorheben der Notwendigkeit, erste ansätze

6.3 Load Balancing-Algorithmen

round robin, weighted distr, least conn, ...

6.4 Session-basiertes Load Balancing

Erklärung der Vorteile und Idee zur Verwirklichung

6.5 Load-Balancing-Frameworks

Einführung und Unterschiede

7 Konsistenz und Datenhaltung

7.1 Grundlagen & Erklärung

cap-theorem, hervorheben der notwendigkeit, vermeidung von inkonsistenzen, ...

7.2 Verschiedene Transaktionsprotokolle

2-phase, 3-phase, 2-phase-lock, long-duration, transaction, jta

7.3 Lösungsansätze bei Transaktionskonflikten

Aufzählen und erklären der verschiedenen vorgehensweisen

Literatur

[1] Burkhard Hampl und Simon Wortha. Cloud computing-ausarbeitung systemtechnik 2015/16. https://elearning.tgm.ac.at/mod/resource/view.php?id=48953. zuletzt besucht: 16.05.2016.

[2] Margaret Rouse. Definition - internet der dinge (internet of things, iot). http://www.searchnetworking.de/definition/Internet-der-Dinge-Internet-of-Things-IoT. zuletzt besucht: 16.05.2016.

[3] whatis.com. Definition - interprocess communication (ipc). http://whatis.techtarget.com/definition/interprocess-communication-IPC. zuletzt besucht: 16.05.2016.

[4] Selina Brinnich und Niklas Hohenwarter. Service-oriented architecture. https://elearning.tgm.ac.at/mod/resource/view.php?id=45143. zuletzt besucht: 16.05.2016.

[5] Dominik Dolezal. Sockets. https://elearning.tgm.ac.at/mod/resource/view.php?id=48863. zuletzt besucht: 16.05.2016.

[6] TU Clausthal. Remote procedure call. http://zach.in.tu-clausthal.de/teaching/programming_0506/literatur/linuxfibel/rpc.htm. zuletzt besucht: 16.05.2016.

[7] Anja Austermann. Rmi – verteilte programmierung unter java. http://www.cul.de/data/jbuilderpr.pdf. zuletzt besucht: 16.05.2016.

Listings

Abbildungsverzeichnis