VSYS Bianca Ploch

Verteilte Systeme Übung 01

1.1 Definition verteilter Systeme

a	Ja	Mehrere Rechner sind vorhanden mit jeweils eigenen Prozessoren und Speichern
b	Nein	Nur ein Rechner ist vorhanden
c	nein	Hat gemeinsamen Speicher, wiederspricht Schill & Springers Definition
d	ja	Verschiedene verteilte Computer die zusammen an einer Aufgabe arbeiten

1.2 Ressourcen

Hardware-Ressourcen

- Drucker
  - mehrere Computer senden verschiedene Druckaufträge
- CPU
  - ein Rechner stellt seine Rechenleistung zB. als Server anderen Rechnern im verteilten System zur Verfügung
- Festplattenkapazität
  - mehrere Rechner greifen auf eine Festplatte zu und führen Operationen darauf aus

Daten- & Software-Ressourcen

- Dateien
  - mehrere Rechner haben Lese- und Schreibzugriff auf eine Datei
- Datenobjekte
  - sind über mehrere Computer verteilt aber teilen Objektdaten und Methoden welche zB. Über *Remote Method Invocation* aufgerufen werden können
- Datenbanken
  - Inhalt kann mit mehreren Rechnern geteilt und genutzt werden

1.3 Sicherheit Chat-Applikation

- Grundlage: Nachrichten müssen in VS gesendet werden
- mögliche Sicherheitsprobleme:
  1. ist sicher, dass es sich um richtigen Server & Client handelt? (Authentizität)
  2. Nachrichten können abgefangen und verändert werden (Integrität)
  3. Nachrichteninhalt wird an mehr Leute verbreitet als vom Versender beabsichtigt (Diskretion)
  4. Verfügbarkeit des VS: Spam, DoS
- Maßnahmen:
  1. Verschlüsselung der Nachrichten
  2. Authentisierung der Nutzer
  3. Captcha, Proof of work

1.4 n-tier-Architekturen

Drei-Tier-Architektur für Client-Server-Systeme:

- Architektur aus 3 Schichten: 1 Client, 2 Server
- Client kann auf Server 1 angebotene Prozeduren aufrufen
- Server 1 holt für die aufgerufene Prozedur erforderlichen Daten von Server 2 per Aufruf ab
- Server 2 sendet Daten als Ergebnis an Server 1
- Server 1 sendet Ergebnis der aufgerufen Prozedur an Client
  → 1 Server zuständig für Anwendungslogik und der andere für Datenmanagement

1.5 Verteilung

- vertikale Verteilung
  - Verteilung erfolgt über mehrere Rechner
  - von Round-trip-time (Zeit für das Schicken eines Datenpaketes im VS und Erhalt der Antwort) abhängig
  - n-tier-Architektur
- horizontale Verteilung
  - Verteilung nur auf einem Rechner (in den logischen Schichten)
  - IP-Implementierung der versch. Schichten

1.6 Uhrensynchronisierung mit Fokus Genauigkeit

- Problem: kein gemeinsamer Speicher in VS
- Zeit wichtig für Ereignisreihenfolge im VS
- Uhren nicht synchron in allen Geräten:
  - Umfeldbedingungen nicht konsistent (Temperatur, etc.)
  - Energiequelle (Batterie schwach)
  - Physikalische Uhren: Quarz ungenau
  - logische Uhren: abweichende Zähl- /Oszillationsgeschwindigkeit

a) *in 2 Rechnern in lokalem Netzwerk ohne Verweis auf externe Zeitquelle*

- Lamport-Uhr:
  - logische Uhr
  - keine Aussage über Kausalität von Ereignissen & Konsistenz (schwache und starke Konsistenzkriterium für Uhren)
  - alle Prozesse haben einen Zähler
  - beim Verschicken von Nachrichten wird der Zähler des Sender-Prozesses um 1 erhöht und mit der Nachricht mitgeschickt
  - Empfänger setzt eigenen Zähler auf diesen Wert plus 1 wenn Empfängerzähler kleiner ist → nach zeitlichem Ablauf sortierte „Uhr“
- Berkeley Algorithmus:
  - ein Rechner wird zum Zeitserver bestimmt
  - Zeitserver fragt Zeit aller anderen Rechner(Clients) ab
  - Zeitserver berechnet *Round Trip Time* RTT und mittelt die Zeiten aller Rechner
  - Zeitserver sendet an jeden Client die Zeit-Differenz zu der lokalen Uhr des Clients
  - Client verlangsamt oder beschleunigt seine Uhr dementsprechen
  - → Genauigkeit hängt von Häufigkeit der Überprüfung ab und der Geschwindigkeit der beiden Uhren

b) *in vielen Rechnern im Internet*

- Synchronisation mittels Abfrage der UTC (Coordinated Universal Time) zB. von einem Zeitserver
  - Genauigkeit abhängig von Häufigkeit der Überprüfung
- Nutzung von GPS zur genauen realen Zeitberechnung
  - Genauigkeit abhängig von Häufigkeit der Überprüfung