

Organisation

Vorlesung

Übung

Prüfung

Einführung

Verteilte Systeme

Eigenschaften und Herausforderungen

Vom lokalen zum verteilten System



■ Verantwortliche

- **Tobias Distler** Raum 0.039 distler@cs.fau.de
- Jürgen Kleinöder Raum 0.043 jkl@cs.fau.de

■ Termin

- Dienstag, 12:15 – 13:45 Uhr
- Raum 0.031-113

■ Web-Seiten

- Skript: https://www4.cs.fau.de/Lehre/SS19/V_VS/Vorlesung/
- Literatur: https://www4.cs.fau.de/Lehre/SS19/V_VS/Literatur/

■ Fragen und Rückmeldungen sind erwünscht!



■ **Client-Server-Systeme**

- Grundlagen
- Fernaufrufe
- Effizienz
- Fehlertoleranz

■ **Replizierte Systeme**

- Replikationstechniken
- Kommunikation innerhalb einer Replikatgruppe
- Georeplikation

■ **Verteilte Algorithmen**

- Synchronisation von Uhren
- Gegenseitiger Ausschluss



■ Verantwortliche

- **Michael Eischer** Raum 0.045 eischer@cs.fau.de
- **Laura Lawniczak** Raum 0.041 lawniczak@cs.fau.de
- Christopher Eibel Raum 0.042 ceibel@cs.fau.de
- Tobias Distler Raum 0.039 distler@cs.fau.de

■ Termine

- Tafelübung: Mittwoch, 12:15 – 13:45 Uhr, 0.031-113 (ab 24.04.)
- Rechnerübung: Mittwoch, 14:00 – 16:00 Uhr, 02.151-113 (ab 24.04.)

■ Web-Seite

- https://www4.cs.fau.de/Lehre/SS19/V_VS/Uebung/

■ Anmeldung

- Web-Anmeldesystem *Waffel*
- <https://waffel.informatik.uni-erlangen.de>



- Tafel- und Rechnerübung
 - Ergänzende und vertiefende Informationen zur Vorlesung
 - Hilfestellungen zur **Bearbeitung der Übungsaufgaben**
 - Klärung von Fragen
 - Abgabe der Übungsaufgaben

- Themen
 - Anwendung von Java Remote Method Invocation (RMI)
 - Entwicklung eines eigenen Fernaufrufsystems
 - Fehlertolerante Fernaufrufe
 - Aktive Replikation von Diensten
 - Verteilte Synchronisation
 - Lesen und Begutachten von Fachliteratur



- Informatik (**Bachelor und Master**)
 - Vertiefung „Verteilte Systeme und Betriebssysteme“
 - 5 ECTS- oder 7,5 ECTS-Modul
- Informations- und Kommunikationstechnik
 - Bachelor: „Wahlmodule aus EEI und INF“ (5 ECTS-Modul)
 - Master: „Wahlpflichtmodul aus INF“ (5 ECTS- oder 7,5 ECTS-Modul)
 - Eingebettete Systeme
 - Kommunikationsnetze
 - Kommunikationsnetze und Übertragungstechnik
- Varianten
 - **5 ECTS: Vorlesung + Übung**
 - Erfolgreiche Bearbeitung aller abzugebenden Übungsaufgaben
 - Mündliche Prüfung
 - **7,5 ECTS: Vorlesung + erweiterte Übung**
 - Erfolgreiche Bearbeitung aller abzugebenden Übungsaufgaben
 - Erfolgreiche Bearbeitung aller Zusatzaufgaben
 - Mündliche Prüfung



- Anmeldung
 - Registrierung in *mein campus*
 - Bitte vom Prüfungsamt vorgegebenen **Anmeldezeitraum** beachten!
- Vereinbarung des Prüfungstermins
 - Gegen Ende des Semesters
 - Alle Angemeldeten erhalten eine **E-Mail mit einem Doodle-Link**
 - Doodle: Terminvorschläge für Anfang, Mitte und Ende der Semesterferien
 - **Auswahl des eigenen Prüfungstermins** in der Doodle-Umfrage
 - Erscheinen des Prüfungstermins in *mein campus* dient als Bestätigung
- Prüfung
 - Gespräch über den Stoff der Vorlesung und (erweiterten) Übung
 - Diskussion behandelte Probleme und möglicher Lösungsansätze
 - Anwendung bekannter Konzepte auf weitere Problemstellungen
 - **Verstehen ist entscheidend**, nicht das Auswendiglernen!



Organisation

Vorlesung

Übung

Prüfung

Einführung

Verteilte Systeme

Eigenschaften und Herausforderungen

Vom lokalen zum verteilten System



- Typische Merkmale verteilter Systeme
 - **Mehrere Rechner**, die unabhängig voneinander ausfallen können
 - **Verbindung durch ein Netzwerk**
 - Interaktion nur durch Nachrichtenaustausch möglich
 - Netzwerk unzuverlässig und mit variablen Latenzen
 - Moderate Übertragungsgeschwindigkeit im Vergleich zu Mehrkernsystemen
 - Unterschied zu Parallelrechnern
 - **Kooperation von Knoten** zur Lösung einer gemeinsamen Aufgabe
- Definition von [Tanenbaum et al.]

„Ein verteiltes System ist eine **Kollektion unabhängiger Computer**, die ihren Nutzern wie ein einzelnes **einheitliches System** erscheint.“

■ Literatur



Andrew S. Tanenbaum and Maarten van Steen
Distributed systems: Principles and paradigms (2nd edition)
Prentice-Hall, Inc., 2006.



- Ausprägungen
 - Physische Verteiltheit
 - Hardware: **Erhöhte Latenzen** aufgrund von Entfernungen
 - Software: Unabhängige Prozesse auf verschiedenen Rechnern
 - Logische Verteiltheit
 - Aufteilung von Aufgaben auf **mehrere eigenständige Komponenten**
 - Kein Alleinstellungsmerkmal physisch verteilter Systeme
 - Problemstellungen verteilter Systeme auch in Mehrkernrechnern zu finden
- Konsequenzen
 - Knoten haben **nicht zwingend dieselbe Sicht** auf den Systemzustand
 - Unschärfepinzip: Knotensicht auf den Systemzustand ist
 - entweder aktuell, aber unvollständig
 - oder vollständig, aber veraltet
 - Herangehensweise
 - Kein Verlass auf globale Sichten (z. B. eine gemeinsame Zeitbasis)
 - **Kombination der lokalen Sichten** unterschiedlicher Knoten



- Nebenläufigkeit
 - Knoten stellen Betriebsmittel zur gemeinsamen Nutzung zur Verfügung
 - „Gleichzeitige“, sich **überlappende Zugriffe sind wahrscheinlich**
 - Koordinierung der Zugriffe erfordert mitunter verteilte Algorithmen
- Fehlerbehandlung
 - Umgang mit **verschiedenen Fehlerklassen**
 - Ausfälle
 - Beliebiges Fehlerverhalten (*Byzantinische Fehler*)
 - Wünschenswerte Maßnahmen sind nicht in jedem Fall praktikabel
 - Fehlererkennung
 - Fehlermaskierung
 - Fehlertolerierung
 - Wiederherstellung nach unterschiedlichen Fehlerarten
 - Transiente Fehler: **Automatisierte Mechanismen** zur Konsistenzwahrung
 - Permanente Fehler: Manuelle Reparatur fehlerhafter Komponenten

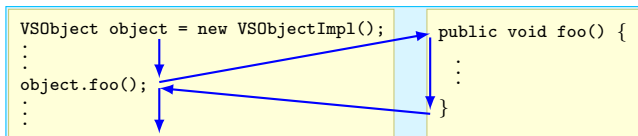


- Heterogenität
 - Knoten eines verteilten Systems sind **nicht notwendigerweise gleichartig**
 - Beispiele
 - Hardware: Smartwatch vs. Server
 - Programmiersprachen: Java vs. C/C++
- Indirekte Bindung
 - Konfigurierung als dynamischer Vorgang
 - **Bindungsunterstützung zur Laufzeit** erforderlich
- Kein einheitlicher Namensraum
 - Zusammenschluss **verschiedener Namensräume**
 - Namensauflösung muss gegebenenfalls über Verwaltungsgrenzen erfolgen
- Kein gemeinsamer Speicher
 - Gültigkeit von Speicherreferenzen ist auf jeweiligen Knoten beschränkt
 - Maßnahmen zum **Umgang mit Referenzen** erforderlich



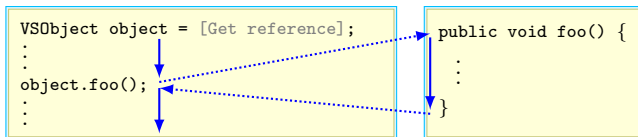
■ Methodenaufwurf im **lokalen System**

- Ausführung der Methode im Thread des Aufrufers
- Kein Fortschritt beim Aufrufer während der Methodenausführung



■ Methoden**fer**naufwurf im **verteilten System**

- Rollenverteilung: Aufrufer (*Client*) und Aufgerufener (*Server*)
- Client-Prozess unabhängig vom Server-Prozess
- Anwendungsbeispiel: Auslagerung aufwändiger Operationen



- **Zentrale Frage: Soll die Verteiltheit dem Client verborgen werden?**
 - Ja → Ähnliche Sicht wie im lokalen Fall
 - Nein → Verteiltheit als Basis für Entwicklungsentscheidungen
- **Netzwerktransparenz**
 - **Zugriffstransparenz**
 - Lokale und entfernte Zugriffe erfolgen über **identische Aufrufe**
 - API: Keine Unterscheidung zwischen lokalen und entfernten Operationen
 - **Ortstransparenz**
 - Zugriff auf Ressourcen **ohne Wissen über ihren Aufenthaltsort**
 - System muss sich um das Auffinden von Ressourcen kümmern
- **Migrations-/Mobilitätstransparenz**
 - Ortswechsel von Clients oder Ressourcen zur Laufzeit möglich
 - Keine Beeinträchtigung durch **Ortswechsel von Komponenten**



- **Redundante Auslegung** der Server-Seite
 - Fehlertoleranz
 - Schutz vor Server-Ausfällen
 - Steigerung der Verfügbarkeit eines Diensts
 - Skalierbarkeit
 - Erhöhung der Leistungsfähigkeit durch zusätzliche Rechner / Ressourcen
 - Eventuell in Kombination mit einer Partitionierung des Anwendungszustands
 - Mehraufwand für Konsistenzwahrung
- Georeplikation
 - Verbesserte Fehlertoleranz durch **geografische Verteilung der Replikate**
 - Leistungseinbußen aufgrund hoher Latenzen
- **Replikationstransparenz**
 - Server-Seite erscheint dem Client als eine Einheit
 - Vorteil: Fehlertransparenz durch dynamischen Replikatwechsel realisierbar
 - Nachteil: Automatische Wahl des Replikats in manchen Fällen ineffizient

