



Willkommen bei Verteilte Systeme

Willkommen bei Verteilte Systeme!

Von Datenbanken
über Webdienste
bis zu p2p und Sensornetzen.

...

Heute: Einführung und Überblick

A distributed system is a system that prevents you from doing any work, when a computer you have never heard about fails.



Dozent

Arne Babenhausrede

- Physik (Dipl., Dr., CO₂)
- Seit 2004 p2p Netze
- Seit 2017 Softwareentwickler als Beruf
- Python, Scheme, HTML/CSS, Java, JS/TS, Fortran, Bash, Emacs, Ruby, ...
- arne_bab@web.de

Vorlesung bis 2020 gemeinsam mit Carlo Götz.



Sie

- Motivation für das Studium?
- Vorwissen zu verteilten Systemen?
- Erinnerung an Netztechnik?



Ihre Wünsche

- Projekt Erfolgreich abschließen
-
-
-

Im Cryptpad / auf Tafel sammeln.



Präsentationen

- Wählen Sie ein Thema
- Themenvorschlag per E-Mail
 - gerne auch eigene
 - sehr gerne mit Vorwissen / Hobby
 - bis zur dritten VL



Projekte 2022

- Berechnung auf mehreren Servern
- Botnetz mit CnC
- Chat mit 2 Servern
- Chat mit Python
- ESP32-Captive Portal
- Game Of Life verteilt
- Gruppenchat in Java
- Gruppenchat mit Javascript
- IoT im Heimnetz
- Kommunikation zwischen Spieler servern
- MQTT Broker



Materialien

Distributed Systems

- Martin van Steen und Tanenbaum (2017)
- kostenloses ebook: <https://www.distributed-systems.net>
- ISBN-13: 978-1543057386

Distributed Systems - An Algorithmic Approach

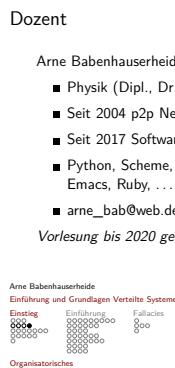
- Sukumar Ghosh (2015).



Arbeitsstelle

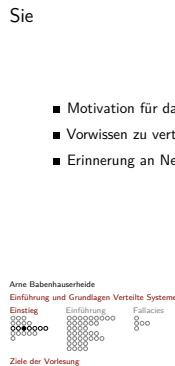
- Disy Informationssysteme GmbH in KA
- Datenanalyse, Reporting, Geoinformation und Geo-Analytics
- ~ 200 MA, Gründung 1997 (ich bin seit 2017 dabei)

<https://www.disy.net>



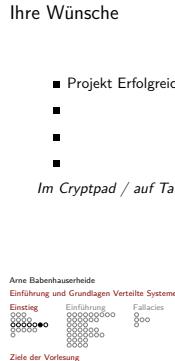
Ziele der Vorlesung

- Sie verstehen, wie sie Verteilung vermeiden sollten.
- Sie verstehen, was sie beachten müssen, wenn Sie verteilen.
- Sie kennen Werkzeuge und Techniken, die Ihnen helfen.
 - Sie können einschätzen, welche Garantien sie wirklich brauchen.
- Sie können einschätzen, welche **Kompromisse** sie eingehen sollten und welche nicht.
 - Latenz < 300ms für eine Webseite
 - Latenz < 30ms für Interaktive Systeme!
- Das passiert Ihnen nicht:
 - <https://www.ccc.de/de/uploads/2022/web-patrouille-ccc>



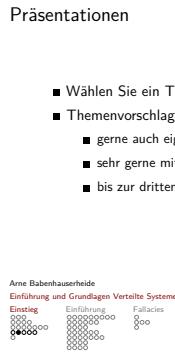
Fragen

- Verteilung der Slides?
- Mail der Kurssprecher(in)?



Themenideen

- FOSDEM Vorträge
- Verbindung in GNU Taler
- Wifi und LoRa mit Reticulum
- libresilient (p2p web fallback)
- Distributed Game Architecture
- Virtualization & Containerization
- Load Balancing + Autoscaling
- Verschlüsselung in Signal
- Botnet(-s)
- Message Queues: Rabbit, Kafka, ZeroMQ, ...
- Seti@home / Folding@Home
- telnet
- IRC
- ssh
- OAuth
- Ein eigenes Projekt
- Papers we love
- OWASP Top 10



Projekte 2023

- Multithreading in Spice
- Nerdlegame als 1vs1
- Online TicTacToe
- Online-Spiel in GoDot
- P2P-Chat mit Directory-Server
- RPi Captive Portal
- Schiffeversenken 1vs1
- P2P Voice Chat über UDP
- Teilen von Trainingsplänen
- VS Chat in Java
- Viewer-Room
- Kaesekestchen
- BlackJack
- ChannelMaster
- DistributedKeyValueStore
- dipool
- Password Encryption App
- Sonoff Basic R2 Webservice
- ChatABC
- MapReduce on Steroids
- Der Pizzabesteller
- dezentraler Chat
- SmartDown
- eins-chat
- peer gewinnt
- MileStarter
- DontGetAngry



Arne Babenhausrede

Einführung und Grundlagen Verteilte Systeme

Folien u.ä.

■ Weitere Quellen werden bei Verwendung aufgeführt.

■ Folien: <https://www.draketo.de/software/vorlesung-verteilte-systeme>

■ Glossar zu Netztechnik:

<https://www.draketo.de/software/vorlesung-netztechnik#glossar-netztechnik>

■ Gemeinsamer Glossar und Notizen in einem cryptpad.

Zur Vorbereitung: Drucken Sie bitte die Version mit vielen Folien pro Seite für schnelle Notizen.



Arbeitshintergrund

Tech hinter Cadenza

- Java 24 (mit Spring und Ignite)
- Web-Tech (Web-Components, JS/TS)
- Verschiedene Datenbanken
- 3 mio LOC

Organisation

- ~80% in Karlsruhe, weitere international verteilt.
- N-Wochen Sprints, viel Home-Office ⇒ Zoom, Rocket Chat
- Tools: Linux, Mac, Windows, IntelliJ, VS Code, Emacs, ...
- Infrastruktur: CI mit Jenkins und Gitlab, Kubernetes-Cluster
- Trunk-based development mit slibs gegen Merge-Konflikte



Erwartungen

Meine Wünsche

Ich will, dass Sie gerne kommen.
Es ist Arbeit, und Arbeit sollte Spaß machen.
Ich will, dass Sie Verständnis von Verteilten Systemen mitnehmen.



Ablauf Semester

- Grundlagen
- p2p (peer-to-peer)
- Clocks/Zeit, usw.
- Algorithmen, Shared State
- Datenbanken konkret: CAP, DBs
- Sicherheit in der Praxis, Sensornetze
- Präsentationen der Studierenden, 15min pro 3er-Gruppe
- tbd, Wiederholung



Projekte 2021

- All Chats Are Beautiful
- Distributed Key/Value Store
- FreeChat
- HTTP-Tunneling
- IPC in Interpreter
- Kooperative WLED-Steuerung
- Load Balancer
- p2p Chat in Minecraft
- p2p iOS Kommunikation (Local Pal)
- Risiko Multiplayer
- Schiffe versenken
- Snake Multiplayer (Butchered Snake)
- Statuspage (openmonitor)
- Vier gewinnt
- Vollständig dezentralisierte MessageQueue
- WebChat



Projekte 2024

- ChatApp
- Schachspiel
- Schiffe versenken
- JWT Chat
- Chat in Python
- Konsolenbasierter Gruppenchat in Java
- Verteilte 3DS-ButtonBox
- Dokument- Kollaborationsanwendung
- Taskify





Projekte 2025

- Task Manager
- Online 4-gewinnt
- SSL Chat Projekt
- Musik Synchronisation
- Kollaboratives Whiteboard
- Ich packe meinen Koffer
- lightweight Load Balancer
- Multiplayer Dartscounter
- Tic Tac Toe
- Reversi



Was sind verteilte Systeme?

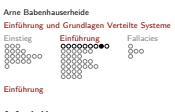
- Was sind für Sie verteilte Systeme?
- Beispiele?

Im Cryptpad / auf Tafel sammeln.



Sammlung von Knoten: Gruppenzugehörigkeit

- Einstieg ins System
- Darf jeder Knoten beitreten?
- Wie finden sie sich?
- Wie wird sichergestellt, dass nur mit Knoten innerhalb des Systems kommuniziert wird?
- Oft als Overlay Network realisiert.



Middleware

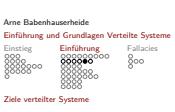
- Separate Schicht über dem Betriebssystem.
- Von Applikationen verwendete Funktionalität:
 - verteilte Transaktionen
 - Fehler Recovery
 - Authentication & Authorization
 - Kommunikation mit anderen Knoten
 - ...
- Risiko: Effizienzverlust durch zu starke Garantien.



Teilen von Ressourcen

Auf geteilte Ressourcen zugreifen

- Beispiele:
 - Dropbox, GDrive etc.
 - Google Docs
 - p2p Filesharing (Bittorrent, Blizzard Launcher)
 - p2p Streaming (Spotify anfangs)
 - p2p Accountsuche (Skype anfangs)



Offenheit

- Verteilte Systeme bieten und verwenden Komponenten, die einfach integriert oder wiederverwendet werden können
- Anforderungen:
 - definierte Schnittstellen (IDL (Syntax) + docs (Semantik))
 - Anwendungen portabel
 - Systeme erweiterbar

IDL: Interface Definition Language.³

Wer kontrolliert die API?

³Im OS: Hurd translator. Beispiele: <https://git.savannah.gnu.org/git/hurd/hurd.git/tree/hurd/msg.defs#n28>



Ablauf heute

Ablauf heute

- Asynchrone Kommunikation in Microservice Architekturen
- Verteilte Sensorsauswertung
- Telefunkens
- FileShare
- Verteilte Datenbank
- pushNotification
- QuizTogether
- Werwolf
- YT-Watch-Together
- Hot Potato



Was sind verteilte Systeme?

Was sind verteilte Systeme?

Ein verteiltes System ist eine Sammlung autonomer Rechen-elemente, die den Nutzenden wie ein einzelnes kohärentes System erscheint.

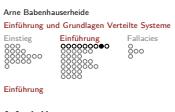
- autonome Rechenlemente** Arbeiten voneinander unabhängig, egal ob Soft- oder Hardware, auch **Knoten** genannt
einzelnes kohärentes System Nutzende haben den Eindruck ein einzelnes System zu bedienen (Erfordert Zusammenarbeit der Knoten).



Was sind verteilte Systeme?

Was sind verteilte Systeme?

- Meist dicht verbunden**
 ■ Für jedes Knotenpaar existiert ein Pfad zwischen den Knoten
- Zwei Varianten:**
strukturierte Overlays Jeder Knoten hat eine definierte Auswahl an Nachbarn mit denen er kommuniziert.
unstrukturierte Overlays Jeder Knoten hat Referenzen zu zufällig ausgewählten anderen Knoten.



Overlay Networks

Overlay Networks

- Meist dicht verbunden**
 ■ Für jedes Knotenpaar existiert ein Pfad zwischen den Knoten
- Zwei Varianten:**
strukturierte Overlays Jeder Knoten hat eine definierte Auswahl an Nachbarn mit denen er kommuniziert.
unstrukturierte Overlays Jeder Knoten hat Referenzen zu zufällig ausgewählten anderen Knoten.



Overlay Networks

Overlay Networks

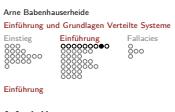
- Meist dicht verbunden**
 ■ Für jedes Knotenpaar existiert ein Pfad zwischen den Knoten
- Zwei Varianten:**
strukturierte Overlays Jeder Knoten hat eine definierte Auswahl an Nachbarn mit denen er kommuniziert.
unstrukturierte Overlays Jeder Knoten hat Referenzen zu zufällig ausgewählten anderen Knoten.



Kohärentes, einzelnes System

Kohärentes, einzelnes System

- Nutzer kann nicht sagen, ob:**
 ■ Berechnungen verteilt stattfinden
 ■ Daten verteilt gespeichert werden
 ■ Daten repliziert werden
 ■ „Verteilungstransparenz“
 ■ Problem: Knoten und Verbindungen zwischen Knoten können (und werden) jederzeit ausfallen.
 ■ Ausfall-Transparenz schwierig bis unmöglich.

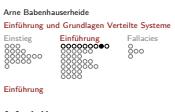


Kohärentes, einzelnes System

Kohärentes, einzelnes System

Just because it is possible to build a distributed system does not necessarily mean that it is a good idea.

- Teilen von Ressourcen
- Verteilungstransparenz
- Offenheit
- Skalierbarkeit
- Macht minimieren



Warum? Ziele verteilter Systeme

Warum? Ziele verteilter Systeme



Warum? Ziele verteilter Systeme

Warum? Ziele verteilter Systeme

Just because it is possible to build a distributed system does not necessarily mean that it is a good idea.

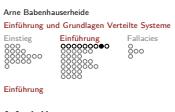
- Teilen von Ressourcen
- Verteilungstransparenz
- Offenheit
- Skalierbarkeit
- Macht minimieren



Verteilungstransparenz

Verteilungstransparenz

- Nutzenden soll nicht auffallen, dass Berechnungen und Daten über mehrere Computer verteilt sind.**



Verteilungstransparenz

Verteilungstransparenz

- Nutzenden soll nicht auffallen, dass Berechnungen und Daten über mehrere Computer verteilt sind.**



Verteilungstransparenz

Verteilungstransparenz

- Nutzenden soll nicht auffallen, dass Berechnungen und Daten über mehrere Computer verteilt sind.**



Verteilungstransparenz

Verteilungstransparenz

- Nutzenden soll nicht auffallen, dass Berechnungen und Daten über mehrere Computer verteilt sind.**



Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

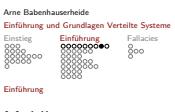
- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)



Verteilungstransparenz: Probleme

Verteilungstransparenz: Probleme

- Latenz²
- Tradeoff: Verteilungstransparenz vs. Performance
- Konsistenz bei Replikation
- komplett Verteilungstransparenz ist unmöglich
- Verteilung für Entwickelnde explizit? (Abstraktionsbruch; aber Effizienz!)





Probleme bei Skalierung der Größe

- Speicherkapazität inkl. I/O Transferrate
- Rechenkapazität, begrenzt durch CPUs
- Netzwerk zwischen Nutzer und System



Skalierungstechniken für Anwendungen

vertikale Skalierung mehr CPU, RAM etc. für die Computer (limitiert)

horizontale Skalierung mehr Kapazität durch Hinzufügen neuer Computer

- viele bestehende Systeme erwarten schnelle LANs
 - Oft synchrone Kommunikation
 - werden langsam durch erhöhte Latenz⁵

⁵ Manchmal sogar mit InfiniBand als Anforderung.

Verstecken von Latenz

- Asynchrone Kommunikation
 - manchmal nicht möglich (Bsp: interaktive Anwendungen)
 - oft komplexere Algorithmen
- Berechnung im Client
 - Bsp: Form-Validierung in JS
 - Konsistenz?
 - Gleicher Code?
 - Gleiche Daten?

PAUSE

Um welche Dimension der Skalierung handelt es sich? Welche Skalierungstechnik wird eingesetzt?

- Ziel: Windows Updates gleichzeitig laden
- Problem: Einbruch Netzeistung durch viele Downloads
- Skalieren Proxyserver, der die Updates einmal von MS lädt
 - Downloads aus dem Firmennetz vom Proxy

Dimensionen: Größe, Geographie, Administration

Techniken: Latenz verbergen, Partitionierung, Replikation

PAUSE

- Unterschiedliche Richtlinien für Verwaltung, Sicherheit etc.
- teilweise politische und soziologische Probleme
- DSGVO?

Fallbeispiel



Partitionierung, Replikation, Caching

Partitionierung verteile komponenten auf mehrere Maschinen

■ Bsp: DNS, DB-Sharding, WWW

Replikation und Caching Kopien auf mehreren Computern

■ Bsp: Replizierte DBs, Browsercache, Proxies

■ Führt zu Inkonsistenz

■ globale Synchronisation ist langsam

■ Abhängig von der Anwendung

PAUSE

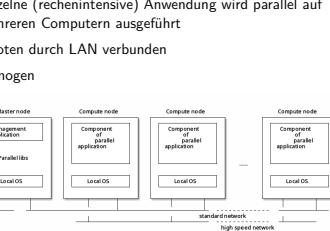
Fallacies of distributed Systems

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

PAUSE



HPC: Grid Computing



HPC: Cluster Computing

- einzelne (rechenintensive) Anwendung wird parallel auf mehreren Computern ausgeführt
- Knoten durch LAN verbunden
- homogen

- keine Annahmen bzgl. Ähnlichkeit von:
 - Hardware
 - Betriebssystem
 - Netzwerk
 - Sicherheit
 - Administrative Domänen
- Bsp: mehrere Hochschulen schließen ihre Cluster zu einem Grid zusammen.
- Grid kann von Allen verwendet werden.

Forschung: Probleme wie beim Cluster, aber Grid gibt Förderung.



Cloud Computing - Schichten

Cloud Computing - Gefahren

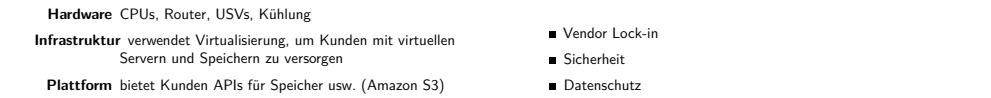
Hardware CPUs, Router, USVs, Kühlung

Infrastruktur verwendet Virtualisierung, um Kunden mit virtuellen Servern und Speichern zu versorgen

Plattform bietet Kunden APIs für Speicher usw. (Amazon S3)

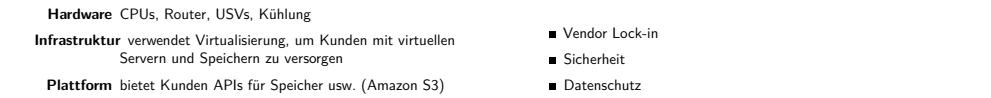
Anwendung Programme für Endanwender (Google Docs)

- Vendor Lock-in
- Sicherheit
- Datenschutz



Cloud Computing

- Geräte sind vernetzt
- Interaktion mit Benutzer ist kaum merkbar
- System erkennt Nutzerkontext und optimiert Interaktion
- Geräte laufen weitestgehend autonom
- System beherrscht viele Interaktionen
- Wer weiß, was ich mache?



Pervasive Systems

Das Netz ist immer dabei.

- Treten durch Mobile und IOT verstärkt auf.
- Wird unterteilt in:
 - Ubiquitous Computing
 - Mobile Computing
 - Sensornetze



Ubiquitous Computing



Distributed Information Systems (DIS)

- Einzelne Anwendungen zu einem verteilten System zusammenfassen
- Oft Legacy-Anwendungen.
- Methoden: verteilte Transaktionen, Enterprise Application Integration





Resource-based - an REST orientiert

- Ressourcen über Namen identifiziert (URLs bei REST).
- Alle bieten die gleiche Schnittstelle (HTTP Verben bei REST).
- Alle benötigten Informationen sind im **Serviceaufruf enthalten** (z.B. als URL-Parameter).
- Nach einem Aufruf vergisst der Service alles über den Aufrufer.

Bsp: REST interface für S3 PUT

<http://bucket.s3.amazonaws.com/Key>

Problem: Großer Zustand (z.B. Error 414: Request-URI Too Long — ab 2048 Zeichen in IE11/Edge16, 8k in CDNs)



Zentralisierte System-Architekturen

- Prozesse in 2 (evtl. überlappende) Gruppen unterteilt:
 - Clients
 - Server
- Server bieten Dienste an.
- Clients nutzen diese Dienste.
- Kommunikation meist Request-Reply

Macht beim Server ⇒ Hierarchie.



Extrembeispiel

<https://dryads-wake.1w6.org/>

Wieviel Logik ist im Server? Schauen Sie nach
⇒ **F12 Hacking Key!**

Bitte schießen Sie es nicht ab. Läuft auf kleinstem mietbaren VPS
:-)



Dezentralisierte System Architekturen (p2p)

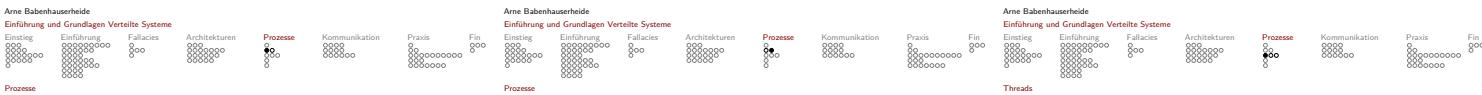
- Prozesse werden nicht nach Client und Server unterteilt; alle sind gleichgestellt.
- Overlay Network:
 - Knoten im Netz sind Prozesse.
 - Kanten im Netz sind Kommunikationswege.
- 2 Arten von Overlay Networks: Strukturiert und Unstrukturiert

Übersicht und Konzepte. Konkrete Netze im nächsten Block.



Zusammenfassung

- Architekturstile: Layered, Object-based, Resource-based, Event-based.
- Zentralisierte Architekturen: n-Schichten.
- Dezentralisierte Architekturen: (un)strukturierte Overlay Networks
- In Realität meist Mischformen.
 - Bsp: BitTorrent verwendet zentralisierte Server (Tracker) zum Sammeln aktiver Knoten.

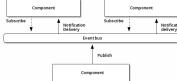


Prozesse: Zustand

- Ein Prozess ist ein Programm in Ausführung.
- Der Zustand eines Prozesses wird im Prozesskontext gespeichert:
 - Registerwerte
 - Stackpointer
 - Programmzähler
 - Memory Maps
 - ...



Event-based | Eventbasiert



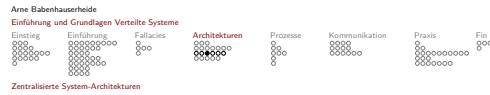
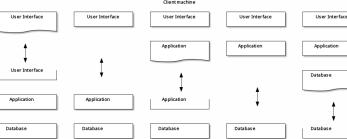
- Komponenten kommunizieren durch Events
- Events durch **Event Bus** propагiert
- Komponenten referentiell entkoppelt (⇒ kein shared memory)
- persistente Speicherung der Events führt zu temporaler Entkopplung
- Verwandte Events als topic abonnieren

Verschiedene Arten von Koordination

Temporally coupled	Temporally decoupled
Referentially coupled	Direct
Referentially decoupled	Event-based
	Mailbox
	Shared data space
	■ Shared data space: fancy für „wie eine Datenbank“
	■ Wie unterscheiden sich die Kosten?



Zweischichtige Konfiguration Diagramm



Zweischichtige Konfiguration

- Die Anwendungsebenen UI, Application und DB werden zwischen Client und Server aufgeteilt.
- Auf 3 Schichten erweiterbar indem die DB auf eigene Maschine ausgelagert wird.



Übung 1/2: Aufgabe



- Ordne die Komponenten 1-4 ihren Anwendungsebenen zu.
- Anwendungsebenen: Welche Komponenten sollten **auf dem Server** und welche **auf dem Client** laufen? Begründen Sie Ihre Antwort. Gruppenarbeit Randbedingungen (nächste Folie)

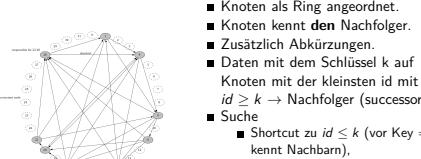
Übung 2/2: Randbedingungen: Latenz Client-Server

- A: 1ms (Lokaler Rechner, effizient⁹)
- B: 10ms (Lokaler Rechner, effizient aber mit Compositor, Regionale Spiele-Ping, Typische USB-Tastatur)
- C: 100ms (Lokaler Rechner¹⁰, Spiele-Ping: 20.000km)
- D: 1s (Neue Webseite öffnen)
- E: 2.5 Minuten bis 5 Minuten (Mars) 50–100 mio km
- F: 3 Wochen (Delay Tolerant Networking — Nomaden)

⁹GVm ohne Compositor → <https://pavelfatin.com/typing-with-pleasure/>
¹⁰vscode: <https://github.com/Microsoft/vscode/issues/27378>



Strukturierte Overlay Networks - Beispiel Chord



- Knoten als Ring angeordnet.
- Knoten kennt den Nachfolger.
- Zusätzlich Abkürzungen.
- Daten mit dem Schlüssel k auf Knoten mit der kleinsten id mit $id \geq k \rightarrow$ Nachfolger (successor).
- Suche
 - Shortcut zu $id \leq k$ (vor Key ⇒ kennt Nachbarn),
 - Nachfolger mit $id \geq k$
- Konstruktion: kürzester Weg zwischen 2 Knoten hat Länge $O(\log N)$.

Flooding

- Algorithmus:
 - Ein Knoten erhält eine Anfrage für einen Wert.
 - Der Knoten sucht lokal nach dem Wert.
 - Findet er ihn nicht, übergibt er die Anfrage an alle Nachbarn.
 - Das Spiel wiederholt sich.
- Kann hohe Last erzeugen.
- Anfragen mit einer Time-To-Live (TTL).
 - z.B. maximale Anzahl von Sprüngen

Geht immer, aber selten gut.



PAUSE



PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



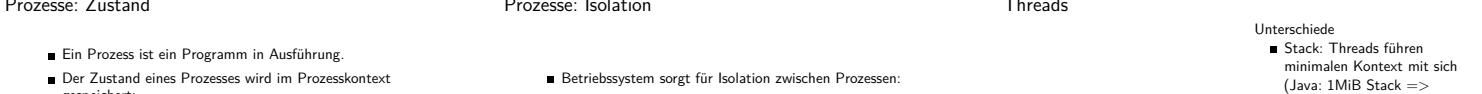
Prozesse: Isolation

- Betriebssystem sorgt für Isolation zwischen Prozessen:
 - Eigene Speicherbereiche.
 - Unerlaubt Zugriffe (durch einen anderen Prozess): segfault.
- Kommunikation über Message passing.
 - Einfachste: Unix pipe.



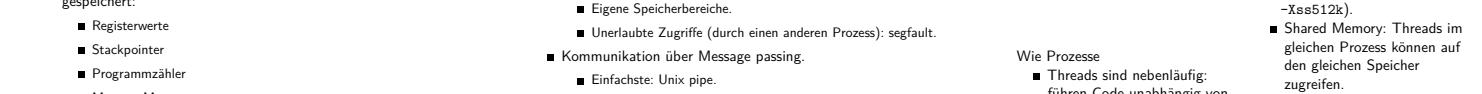
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



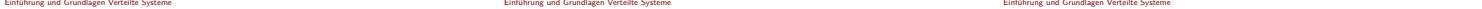
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



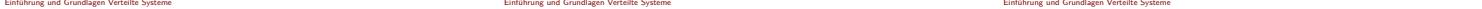
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



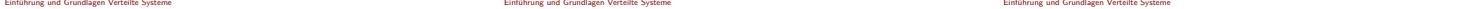
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



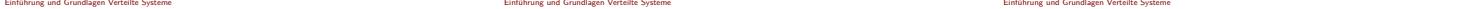
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



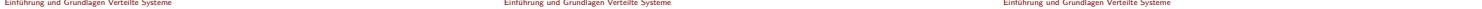
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



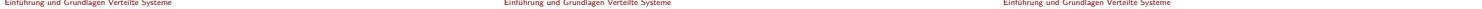
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



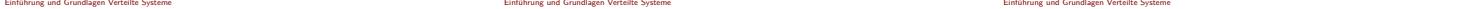
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



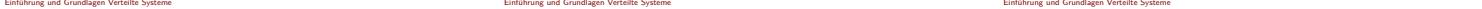
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



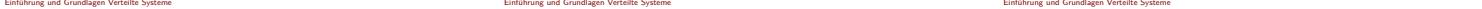
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



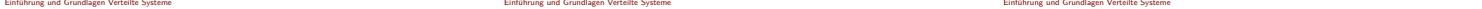
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



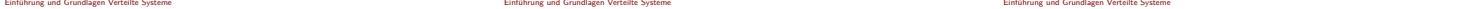
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



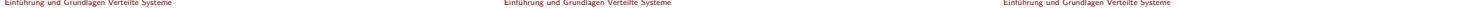
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



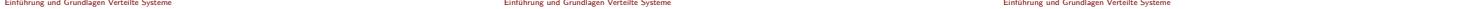
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



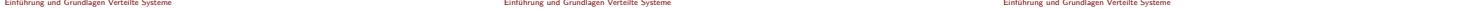
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



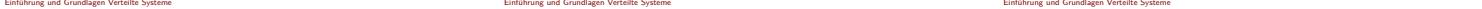
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



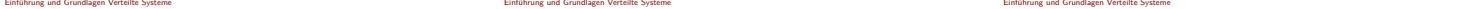
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



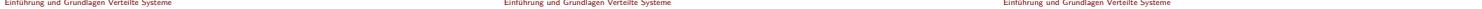
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



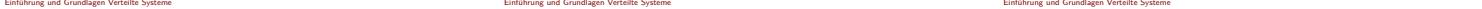
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



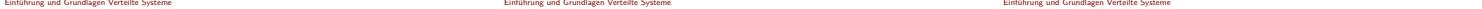
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



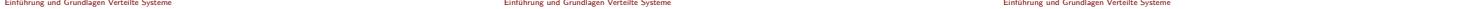
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



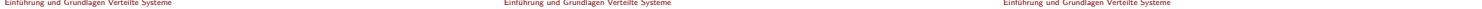
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



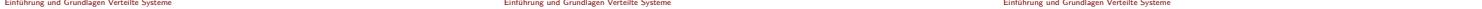
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



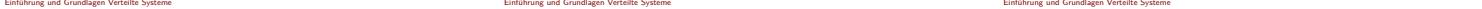
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



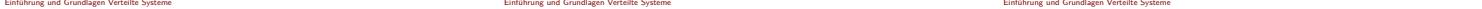
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



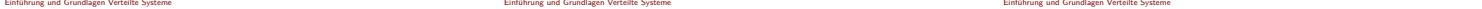
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



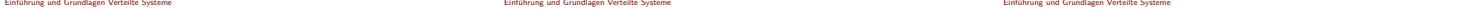
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



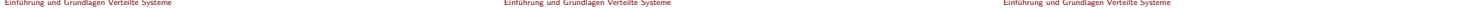
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



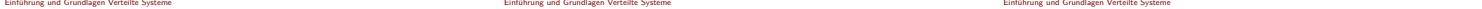
PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



PAUSE

- Sie verstehen den Unterschied zwischen Prozessen und Threads



PAUSE

