Verteilte Systeme

Oktober – November 2023

1. Vorlesung – 11.10.2023

Kurs: TINF21AI1

Dozent: Tobias Schmitt, M.Eng.

Kontakt: d228143
@student.dhbw-mannheim.de

Quelle: https://www.lrz.de/presse/ereignisse/2020-10-10_ExtremeScalingWorkshop_DE/

Verteilte Systeme

- Supercomputer SuperMUC -NG

Insgesamt stehen 311.040 Rechenkerne mit einem Hauptspeicher von 719 TB und einer Spitzenleistung von 26,9 PetaFlop/s zur Verfügung. Alle Rechenknoten sind mit Intel Xeon Skylak-Prozessoren ausgestattet. Die interne Verbindung ist ein schnelles OmniPath-Netzwerk mit 100 Gbit/s.

•Quelle: https://doku.lrz.de/display/PUBLIC/SuperMUC-NG

Verteilte Systeme

- Supercomputer SuperMUC -NG

SuperMUC-NG consists of

6,336 Thin compute nodes each with 48 cores and 96 GB memory

144 Fat compute nodes each 48 cores and 768 GB memory per node

Quelle: https://doku.lrz.de/display/PUBLIC/SuperMUC-NG

Organisatorisches – Durchführung

- Folien zur Verfügung gestellt
  - von Prof. Dr. Holger Gerhards
  - angepasst von Tobias Schmitt

Organisatorisches – Durchführung

- viel Informationsvermittlung
  - Ideenvermittlung – Aspekte, Probleme
  - Kennenlernen weitestgehend allgemeiner Konzepte
- kaum bis keine Mathematik
- Übungen / Aufgaben während der Vorlesung

- Fragen an Dozenten sind erlaubt **und erwünscht**

Organisatorisches – Durchführung

Mi	11.10.2023 – 08:00 bis 11:15 – Raum 037 B
Do	12.10.2023 – 13:45 bis 17:00 – online
Mi	25.10.2023 – 13:00 bis 16:15 – online
Do	26.10.2023 – 13:45 bis 17:00 – online
Di	31.10.2023 – 13:00 bis 16:15 – online
Mi	08.11.2023 – 13:00 bis 16:15 – online
Do	09.11.2023 – 13:45 bis 17:00 – online
Do	16.11.2023 – 13:45 bis 17:00 – online
Mi	15.11.2023 – 08:30 bis 11:00 – Raum 037 B

Organisatorisches – Literatur

- Tanenbaum und Steen, Verteilte Systeme, Prinzipien und Paradigmen (Pearson, 2008, 2te aktualiserte Auflage)
- Bengel, Grundkurs Verteilte Systeme (Springer Vieweg, 2014, 4te Auflage)
- Bengel, Masterkurs Parallele und Verteilte Systeme (Springer Vieweg, 2015, 2te Auflage)

Organisatorisches – Durchführung

ACHTUNG – nicht davon ausgehen, dass man alles nur anhand der Folien versteht

Empfehlung: Mitschrift anfertigen

- Prüfungsleistungen:
  - Klausur!
- Hinweise zur Klausur
  - Überblick über die Themengebiete haben
  - Konkrete Wissensabfrage
  - Konzepte kennen und explizieren können
  - Beispiele nennen können
  - Anwendung auf spezielle Probleme

Themenüberblick heute

- Organisatorisches
- **Wiederholung**
  - Betriebssysteme
  - Kommunikationssysteme
- Einführung in die Verteilten Systeme
- Architektur

Wiederholung Betriebssysteme

- Was ist der Unterschied zwischen einem Programm und einem Prozess?
- Was ist der Unterschied zwischen einem Prozess und einem Thread?
- Welche Scheduling-Strategien kennen Sie?
- Welche Probleme bei der Steuerung mehrere Prozesse kann es geben?
- Wie verhindert man Dateninkonsistenzen bei eingebauten Festplatten, welche durch Systemabstürze entstehen könnten?
- ...

Wiederholung Betriebssysteme

- Prozess → Programm in Ausführung
  - z.B.: Kuchenrezept = Programm, Backen = Prozess
- Thread → Ausführungsfaden innerhalb eines Prozesses
  - z.B.: Rollenspiel
    - Thread 1: Charaktersteuerung
    - Thread 2: Darstellung der Umgebung
    - Thread 3: Monster etc.
- Scheduling-Strategien
  - First-Come-First-Serve
  - Shortest-Job-First
  - Round-Robin

Wiederholung Betriebssysteme

- Probleme bei der Prozesssteuerung
  - Race-Condition
    - Mehrere Prozesse greifen auf eine Ressource zu
    - Ergebnis hängt von der Abarbeitungsreihenfolge ab
    - Auftreten nicht deterministisch
  - Deadlock
    - Eine Menge von Prozessen, die alle auf Ressourcen warten, welche aber von anderen Prozesse aus der Menge gehalten werden.
    - Auftreten nicht deterministich

Wiederholung Betriebssysteme

- Verhinderung von Dateninkonsistenzen bei Speichermedien
  - Journaling
    - Ausführungsschritte werden in Journal geloggt
    - Journal wird abgearbeitet. → Bei Erfolg wird Journal gelöscht.
    - Bei Absturz → Journal wird von Anfang an noch einmal ausgeführt.
    - Bedingung: Ausführungsschritte sind idempotent (mehrmaliges Ausführen führt zum gleichen Ergebnis)

Wiederholung Kommunikationssysteme

- Wie heißen die Schichten des OSI-Referenzmodells?
- Welche Möglichkeiten gibt es physikalisch Daten zu übertragen?
- Welche Möglichkeiten gibt es, über eine Leitung mehrere Kanäle laufen zu lassen? (Multiplexing-Verfahren)
- Was verstehen Sie unter verbindungslosen und verbindungsorientierten Diensten?
- Was wissen Sie über IPv4 und IPv6?
- Was verstehen Sie unter dem Begriff Fluten?

Wiederholung Kommunikationssysteme

- OSI-Referenzmodell
- Möglichkeiten der Übertragung von Daten

– Transport von Speichermedien

– Twisted-Pair-Kabel, Koax-Kabel

– Trägerfrequenzanlagen

– Glasfaserleiter

– Drahtlose Übertragung

Wiederholung Kommunikationssysteme

- Kanalnutzungsmöglichkeiten
  - Zeitmultiplexing
  - Frequenzmultiplexing
  - Codexmultiplexing
- Dienste
  - Verbindungsorientierter Dienst
    - Leitung wird aufgebaut und alle Pakete gehen über diese Leitung

Wiederholung Kommunikationssysteme

- Verbindungsloser Dienst
  - Alle Pakete können unterschiedliche Route im Netz gehen

Wiederholung Kommunikationssysteme

- IPv4
  - 32-Bit-Adressen – Präfix + Host-Anteil
  - CIDR – Classless InterDomain Routing

- IPv6
  - 16-Byte / 128-Bit-Adressen
  - Erweiterung aufgrund Verknappung der IPv4-Adressen + Vereinfachung des Protokolls

Wiederholung Kommunikationssysteme

- Begrifflichkeit – Fluten
  - „Jedes ankommende Paket wird an jede Ausgangsleitung verschickt, außer an die Herkunftsleitung“
  - Zweck: Broadcast oder für Routingalgorithmus
  - Problem der ewig laufenden Pakete muss verhindert werden.
  - (Nutzung eines Teilstreckenzähler)

Themenüberblick heute

- Organisatorisches
- Wiederholung
  - Betriebssysteme
  - Kommunikationssysteme
- **Einführung in die verteilten Systeme**
- Architekturen

Einführung in die Verteilten Systeme

- Was stellen Sie sich unter verteilte Systeme vor?
- Welche Gründe kann es für den Einsatz von verteilten Systemen geben?
- Welche Ziele gibt es für verteilte Systeme?
- Welche Probleme könnten bei verteilten Systemen existieren?
- Kennen Sie Beispiele für verteilte Systeme? Welche wären das?

Gründe für den Einsatz

- Skalierbarkeit
- Bereitstellung von teuren oder entfernten Ressourcen
- Erhöhung der Ausfallsicherheit (Erzeugung von Redundanzen)
- Wirtschaftliche Gründe (Vernetzung preisgünstigerer Rechner)

Gründe für den Einsatz

- Kooperation
- Lastverteilung
- Nebenläufigkeit
- ...

Definition nach Tanenbaum

Ein verteiltes System ist eine Ansammlung *unabhängiger Computer*, die den Benutzern wie *ein einzelnes kohärentes System* erscheinen.

(Tanenbaum u. Steen, Verteilte Systeme, S. 19)

- Unabhängige Computer = autonome Komponenten
- Benutzer = Menschen oder Programme
- Verhalten wie ein einziges System

Definition nach Lamport*

Sie wissen, dass Sie eines haben, wenn der Absturz eines Computers, von dem Sie nie zuvor gehört haben, verhindert, dass Sie Ihre Arbeit erledigen können.

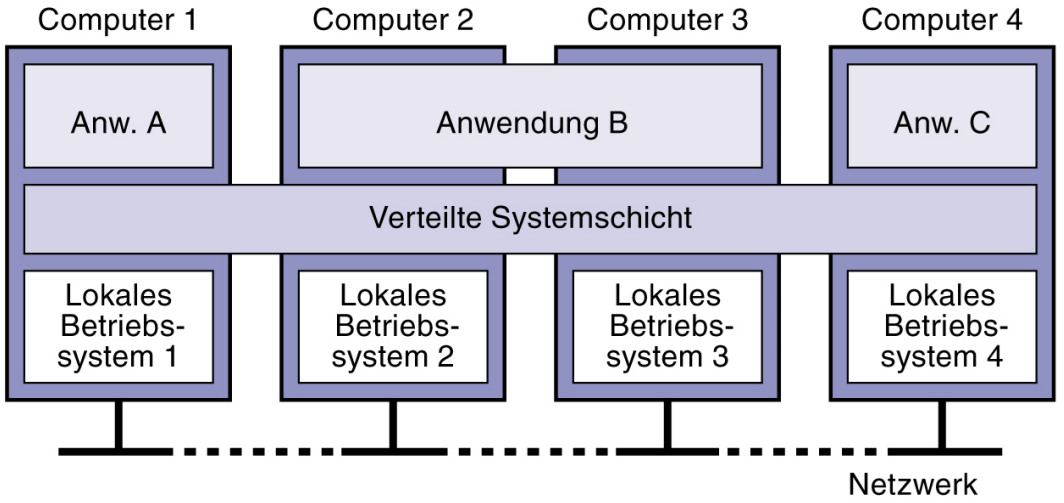
von Leslie Lamport (nach Tanenbaum u. Steen, Verteilte Systeme, S. 23)

Aspekte verteilter Systeme

- Zusammenarbeit autonomer Komponenten
  - Sensorknoten bis Mainframe-Computer

- Eigenschaften
  - Verbergen der
    - der Unterschiede zwischen verschiedenen Computern
    - der Art der Kommunikation miteinander
    - des Internen Aufbaus
  - Konsistente und einheitliche Nutzung des Systems durch Benutzer und Anwendungen
  - Erweiterbarkeit und Skalierbarkeit
  - …

- Verteiltes System manchmal als
- Middleware bezeichnet



Ziele von verteilten Systemen

- Leichter Zugriff auf Ressourcen
- Verbergen, dass Ressourcen über ein Netzwerk verteilt sind
- Es sollte offen sein.
- Es sollte skalierbar sein.

Ziel: Ressourcenzugriff

- Beispiele für Ressourcen
  - Drucker, Computer, Speichergeräte, Daten, Dateien, Webseiten, Software, …
- Gründe
  - Einsparung
  - Informationsaustausch

Ziel: Verteilungstransparenz

- Transparenz = Verbergen diverser Sachverhalte
- Arten von Transparenz

Transparenz	Beschreibung
Zugriff	Verbirgt Unterschiede in der Datendarstellung und die Art und Weise, wie auf eine Ressource zugegriffen wird
Ort[1]	Verbirgt, wo sich eine Ressource befindet
Migration	Verbirgt, dass eine Ressource an einen anderen Ort verschoben werden kann
Relokation	Verbirgt, dass eine Ressource an einen anderen Ort verschoben werden kann, während sie genutzt wird
Replikation	Verbirgt, dass eine Ressource repliziert ist
Nebenläufigkeit	Verbirgt, dass eine Ressource von mehreren konkurrierenden Benutzern gleichzeitig genutzt werden kann
Fehler	Verbirgt den Ausfall und die Wiederherstellung einer Ressource

Hinweis: Man spricht beispielsweise von Zugriffstransparenz oder Fehlertransparenz.

Ziel: Offenheit

- Schnittstellendefinitionen, bestenfalls
  - richtig spezifiziert
  - vollständig
  - neutral
- Vollständigkeit und Neutralität wichtig für Interoperabilität und Portabilität
  - Interoperabilität = Nebeneinander verschiedenen Implementierungen (ggf. versch. Hersteller)
  - Portabilität = Nutzung einer Implementierung für ein anderes verteiltes System mit denselben Schnittstellen

Ziel: Skalierbarkeit

- Skalierbarkeit bzgl.
  - Größe
  - Geografie
  - Administration
- Skalierungstechniken
  - Verbergen von Latenzzeiten der Kommunikation
    - Asynchronität und Optimierung der Kommunikation
  - Verteilung
    - z.B. WWW … wirkt wie ein riesiges dokumentenbasiertes Informationssystem
  - Replikation (ggf. auch Caching)

Probleme von verteilten Systemen

- Sicherheit
  - Abhören und Aufzeichnen von Kommunikationen
  - Erstellung eines Vorliebenprofils
- Transparenz um jeden Preis nicht vorteilhaft
- Probleme aus der Skalierung
  - Zentralisierte Dienste, Daten, Algorithmen
  - Dauer und Zuverlässigkeit der Kommunikation bei Weitverkehrsnetzen
- ...

Probleme von verteilten Systemen

- Entwicklung von verteilten Systemen schwierig
- Typische Fehlerannahmen bei der Entwicklung sind:
  - Das Netzwerk ist zuverlässig.
  - Das Netzwerk ist sicher.
  - Das Netzwerk ist homogen.
  - Die Topologie ändert sich nicht.
  - Die Latenzzeit beträgt null.
  - Die Bandbreite ist unbegrenzt.
  - Die Übertragungskosten betragen null.
  - Es gibt genau einen Administrator.

Beispiele für verteilte Systeme

- WWW
- Wikipedia
- SETI@home
- Facebook, Google, …
- Napster, BitTorrent, ...
- ...

Themenübersicht der Vorlesung

- Architekturen
- Prozesse
- Kommunikation
- Synchronisierung
- Konsistenz und Replikation
- Fehlertoleranz
- Sicherheit

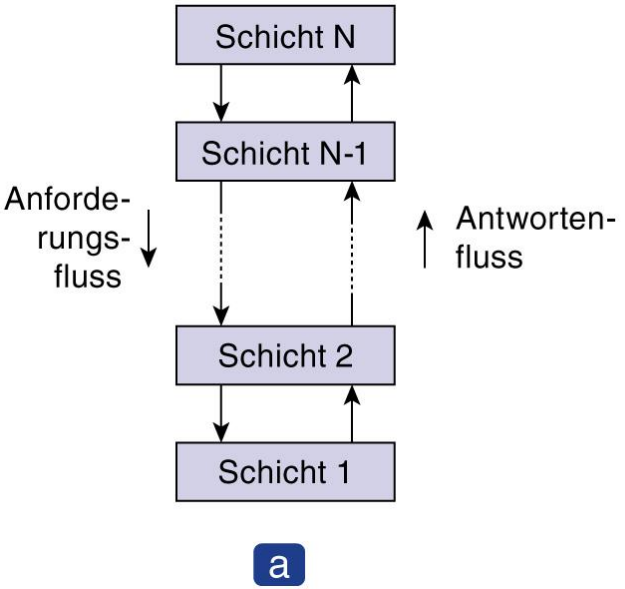
Themenüberblick heute

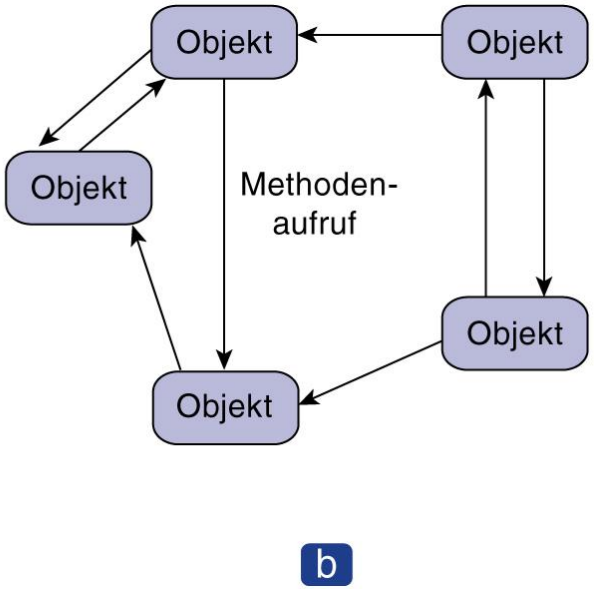
- Organisatorisches
- Wiederholung
  - Betriebssysteme
  - Kommunikationssysteme
- Einführung in die verteilten Systeme
- **Architekturen**
  - Architekturstile
  - Systemarchitekturen

Architekturstile

- Geschichtet Architekturen
  - Schicht L_i ruft Schicht L_{i-1} auf (z.B. bei Netzwerken)
- Objektbasierte Architekturen
  - Jede Komponente ein Objekt → losere Anordnung
  - Übereinstimmung mit Client-Server-Architektur

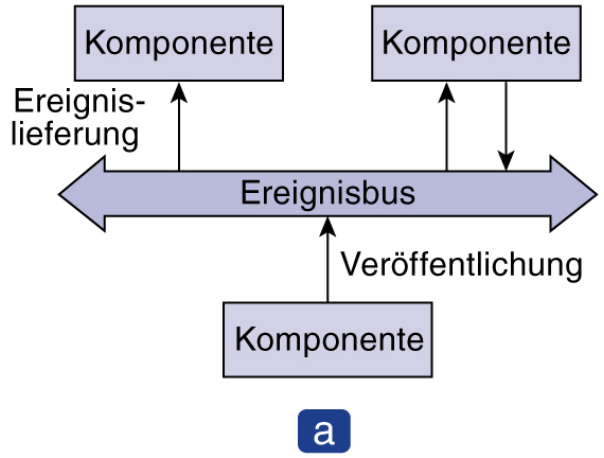
Architekturstile

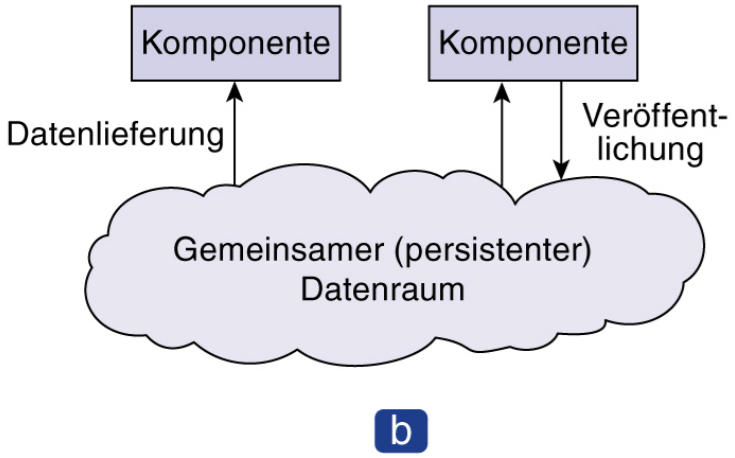




Architekturstile

- Ereignisbasierte Architekturen
  - Prozesse kommunizieren über Weitergabe von Ereignissen
- Datenzentrierte Architekturen
  - Idee der Kommunikation über gemeinsames Repository



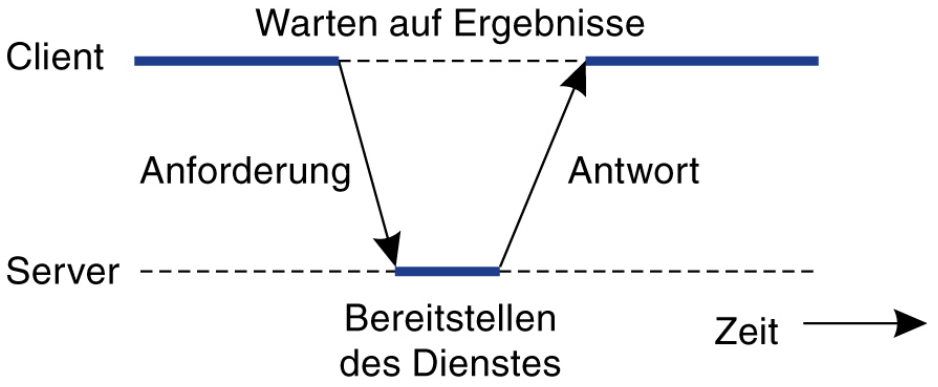


Systemarchitekturen

- Zentralisierte Architekturen
  - Multitier-Architekturen
- Dezentralisierte Strukturen
  - Peer-to-Peer-Systeme
- (Hybridarchitekturen)

Zentralisierte Architekturen

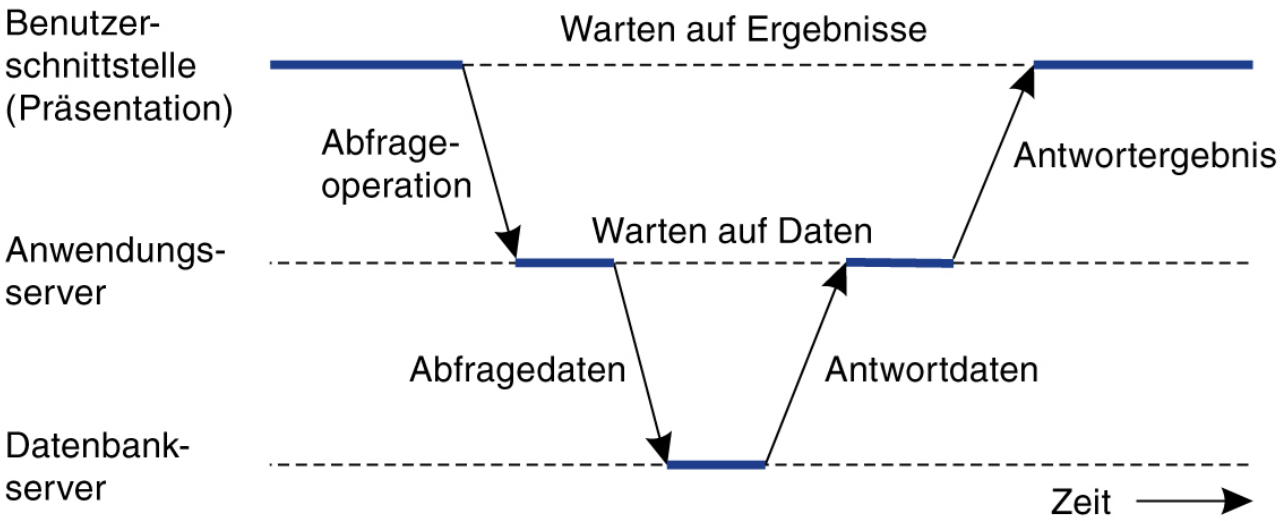
- Entsprechung: Client-Server-Modell
- Server
  - Prozess, der einen bestimmten Dienst implementiert
- Client
  - Prozess, der einen Dienst von einem Server anfordert
- Client-Server-Zusammenarbeit
  - Entspricht *Anforderungs-/Antwortverhalten*
  - Bzw. *Request/Reply*



Zentralisierte Architekturen Anwendungsschichten

- Client-Server-Modell problematisch
  - Klare Unterscheidung nicht immer möglich.
- Aufteilung in 3 Ebenen
  - Ebene der Benutzerschnittstelle (presentation layer)
    - Zusammenarbeit mit dem Benutzer, Darstellungsebene
  - Verarbeitungsebene (application layer)
    - Enthält normalerweise die Anwendung, Kernfunktionalität
  - Datenebene (data access layer)
    - Verwaltung der Daten

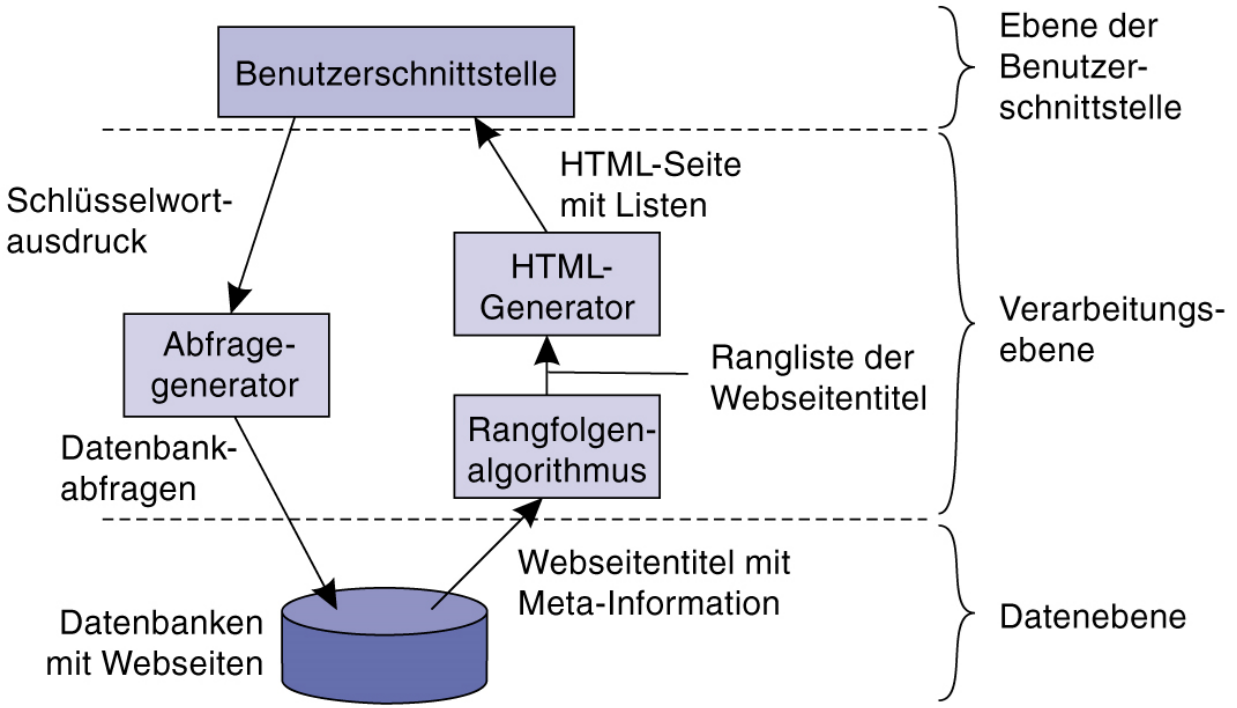
Zentralisierte Architekturen
Anwendungsschichten



Zentralisierte Architekturen

- Beispiele:
  - Suchmaschine
  - Entscheidungssystem für den Aktienhandel
  - Desktop-Paket
  - ...

Zentralisierte Architekturen



Bsp.: Suchmaschine

Zentralisierte Architekturen

- Rahmenbedingung:
  - 1 Computer = Client
  - 1 Computer = Server
- Frage:

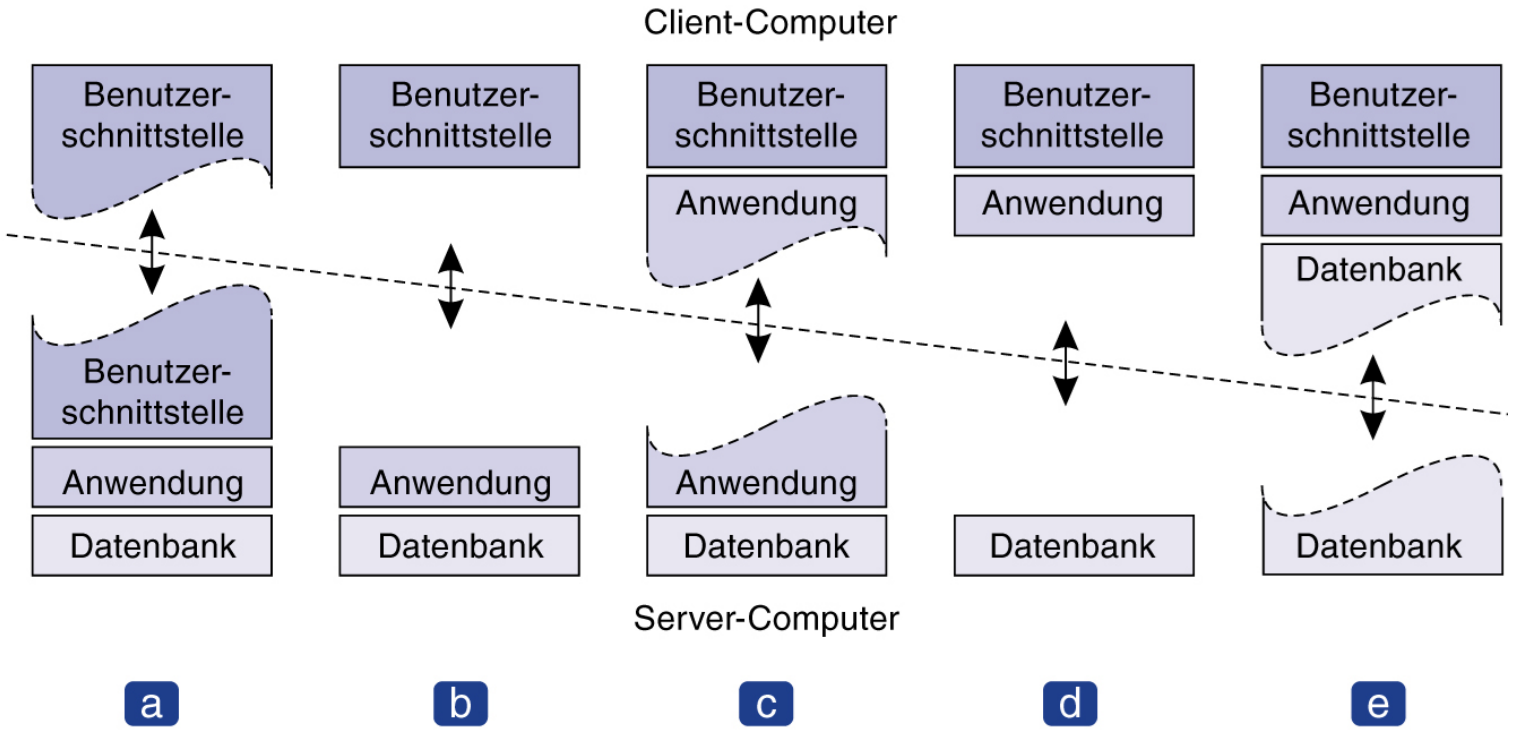
  Wie lassen die 3 Ebenen auf Client und Server verteilen?

- Erinnerung:
  - Benutzerschnittstelle
  - Verarbeitungsebene
  - Datenebene

Zentralisierte Architekturen
Multitier-Architekturen

- Zwei-Tier-Architektur / Zwei-Schicht-Architektur
  - Aufteilung der Schichten auf 2 Computer
  - Grundlage: 1 Client und 1 Server
  - Stichwort: Thin Clients und Fat Clients (Thick Clients)

Zentralisierte Architekturen
Multitier-Architekturen



Zentralisierte Architekturen
Multitier-Architekturen

- Drei-Tier-Architektur
  - Typisches Beispiel: Webseiten
  - Einfacher Aufbau
    - Benutzerschnittstelle (Präsentation)
    - Anwendungsserver
    - Datenbankserver

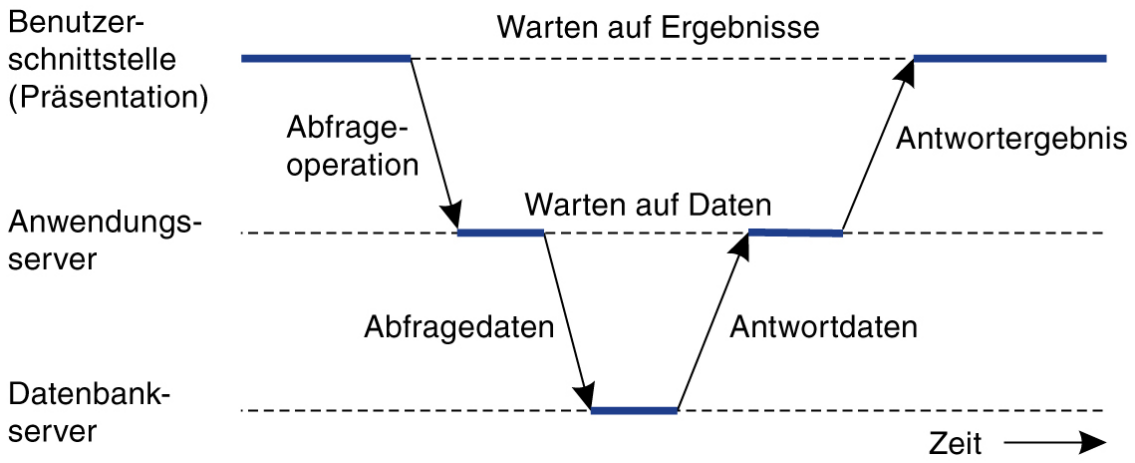
Beispiel:

Web-Frontend

Apache-Server

SQL-Server
(MySQL, MariaDB)

Zentralisierte Architekturen
Multitier-Architekturen



Zentralisierte Architekturen Erweiterungen

- Fragen:

  Welche Vorteile könnte haben eine Client-Server-Server-Struktur zu haben?

  Welche Rollen / Aufgaben könnte der mittlere Server haben?

Zentralisierte Architekturen Erweiterungen

- Aufbau: **Client-Server-Server**
- Vorhandensein
  - mehrerer Clients
  - mehrerer Endserver

Zentralisierte Architekturen Erweiterungen

- Aufbau: **Client-Server-Server**
- Aufgaben des Zwischenservers
  - Proxy
  - Broker
  - Trader
  - Filter
  - Balancer
  - Koordinator
  - Agent

Zentralisierte Architekturen Erweiterungen

- Proxy
  - Vertretung mehrerer Server
  - Ggf. Caching von aufgerufenen Dokumenten
  - Ggf. Entlastung des Web-Servers oder sogar der allgemeinen Netzlast
- Broker
  - Vermittler zwischen Clients und Serven
  - Besitzt Informationen über benannte Services und bietet Zugriffsmöglichkeit darauf für Clients
  - Zweck: Ortstransparenz

Zentralisierte Architekturen Erweiterungen

- Broker (noch mehr Wissenswertes)
  - Unterteilung
    - Intermediate Broker / forwarding Broker
      - Weiterleitungseigenschaften
    - Separater Broker / handle-driven Broker
      - Rückgabe aller Infos an Client, damit dieser mit entsprechenden Service eines Servers zu interagieren
    - Hybrider Broker
      - Funktionalität je nach Anfrage
  - Problem
    - Zentraler Broker → Ausfallproblematik und Flaschenhals
  - Lösung
    - Replikation des Brokers

Zentralisierte Architekturen Erweiterungen

- Trader
  - Ausgangspunkt: mehrere Server für einen Service, aber ggf. in unterschiedlicher Qualität
  - Trader übernimmt Auswahl des geeignetesten Servers für Client
- Filter
  - Aufgabe: Anfragen an oder Antworten vom Server analysieren und modifizieren
  - Möglichkeit zur Auslagerung wiederkehrender Tätigkeiten des Servers zur Vor- bzw. Nachbearbeitung
  - z.B. für Verschlüsselung, Datenkomprimierung, Logging

Zentralisierte Architekturen Erweiterungen

- Balancer
  - Annahme: mehrere identische bzw. replizierte Server stehen zur Verfügung
  - Sorgt für Gleichauslastung (kein unter- oder überbelasteter Server)
- Koordinator
  - Annahme: Service besteht aus mehreren Einzelservices (Durchlauf nach Reihenfolge, alternative Reihenfolgen mgl. oder parallele Ausführung)
  - Verbirgt replizierte Server, sich ergänzende Server, mehrere Teilservices
  - Stichwort: Service Composition

Zentralisierte Architekturen Erweiterungen

- Agent
  - Zweck: mehrere und komplexere Serviceleistungen in Anspruch nehmen
  - Serviceanfrage wird zerlegt und koordiniert
  - Rückantworten werden gesammelt und zu einziger Rückantwort zusammengestellt
  - Abgrenzung zum Koordinator: Selbstbestimmung des Agenten über aufzurufende Services
  - Arbeitsweise zwischen Agenten und Servern
    - iterativ oder parallel

Zentralisierte Architekturen Erweiterungen

- Client-Server-Ketten
  - z.B. Client – Agent – Broker – Server ($C^+S_AS_{Br}S^+$)
  - Rekursiv – Anfrage an nächsten Prozess der Rekursionsstufe und Rückanwort wieder an die vorhergehende Rekursionsstufe
  - Transitiv – Anfrage an nächsten Prozess der Rekursionsstufe, aber Rückantwort an den Client
- Client-Server-Bäume

Dezentralisierte Architekturen

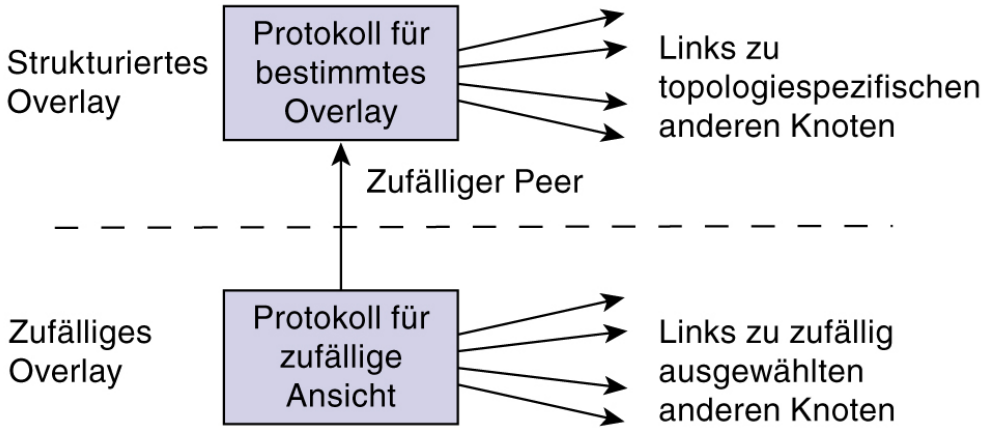
- Peer-to-Peer-Systeme
  - Client und Server horizontal verteilt
  - Prozesse sind symmetrisch und agieren gleichzeitig als Client und als Server (Servent = Server+Client)
  - Anordnung des Overlay-Netzwerks
    - Strukturierte Peer-to-Peer-Systeme
    - Unstrukturierte Peer-to-Peer-Systeme

Dezentralisierte Architekturen

- Strukturierte Peer-to-Peer-Architekturen
  - Konstruierung des Overlay-Netzwerks durch deterministische Verfahren
  - Häufig eingesetzt: verteilte Hash-Tabellen (distributed Hash-Table, DHT)
  - Struktur vereinfacht Zugriff auf angefragte Datenelemente
- Unstrukturierte Peer-to-Peer-Architekturen
  - Zufallsalgorithmen zur Konstruktion des Overlay-Netzwerks
  - Idee: jeder Knoten hat Liste von Nachbarn
  - Datenelemente sind zufällig auf Knoten verteilt
  - Suchanfrage auf bestimmte Datenelemente führen zum überfluten des Netzwerkes

Dezentralisierte Architekturen

- Topologieverwaltung in Overlay-Netzwerken
  - Idee: unstrukturierte Peer-to-Peer-Systeme tauschen Einträge der Listen zu Nachbarn aus
  - Sorgfätiges Austauschen und Auswählen der Einträge kann zu strukturiertem Overlay-Netzwerk führen
  - Realisierung durch 2-Schicht-Ansatz



Dezentralisierte Architekturen

- Superpeer-Konzepte
  - Ansatz bei anwachsenden unstrukturierten Peer-to-Peer-Systemen
    - Problem: Anordnung relevanter Datenelemente
    - Last auf das Netzwerk wächst beim Durchsuchen
  - Auswahl eines Knotens als Superpeer
    - Agiert als Makler (Broker)
    - Oder verwaltet Index (Caching)
  - Client-Superpeer-Beziehung meist fest
  - Problem bei Ausfall eines Superpeers
    - Führerauswahlproblem