Vorlesung
Verteilte Systeme

Prof. Dr. Ronald Moore

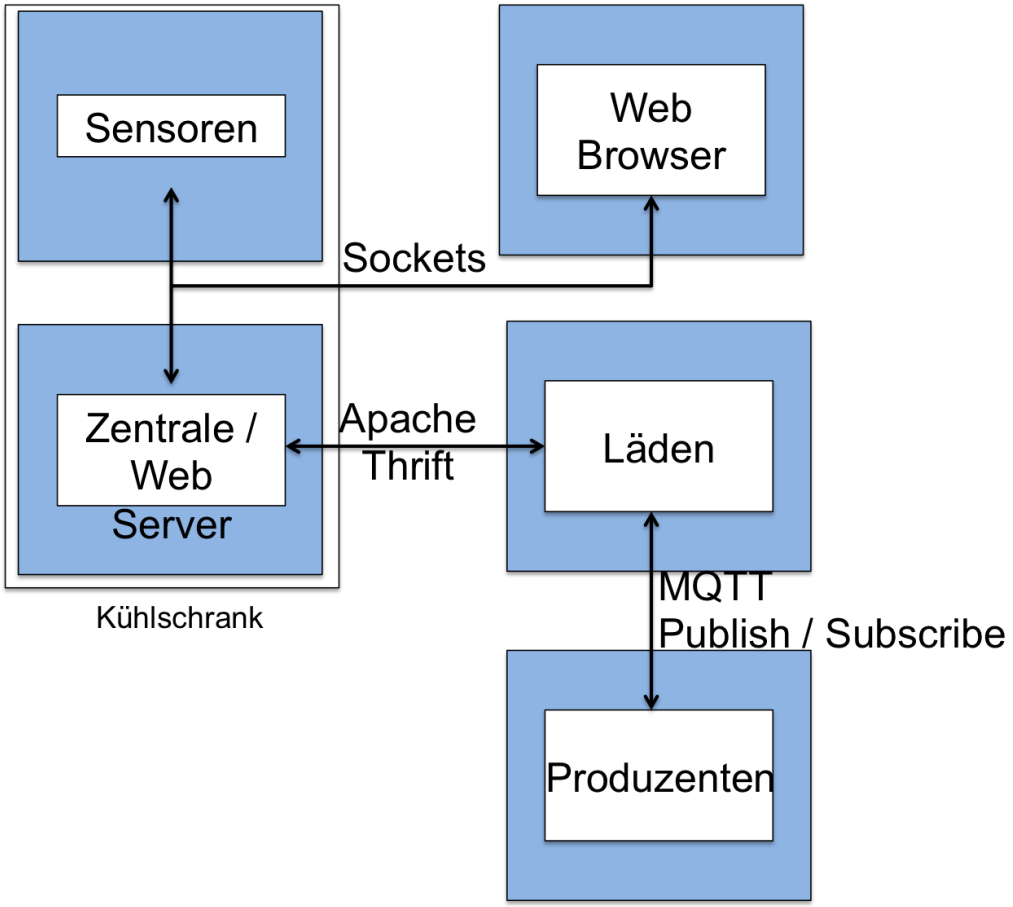
h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

basierend auf Vorlagen von
Lars-Olaf Burchard & Alois Schütte & Peter Wollenweber

Organisatorisches

- Ich betreue keine Praktikums-Gruppe.
  - Alle VS-Praktika sind im Sommersemester 2017 gleich, ob von Kollegen Abel, Burchard oder Priesnitz betreut!
  - Vielleicht dieselbe Klausur (das ist eine Absichtserklärung; noch nicht versprochen).
  - Alle Praktika finden in Labor D14/3.10 statt, und verwenden die dort installierten VMs (virtuelle Maschinen).
  - Sehe Moodle!!!



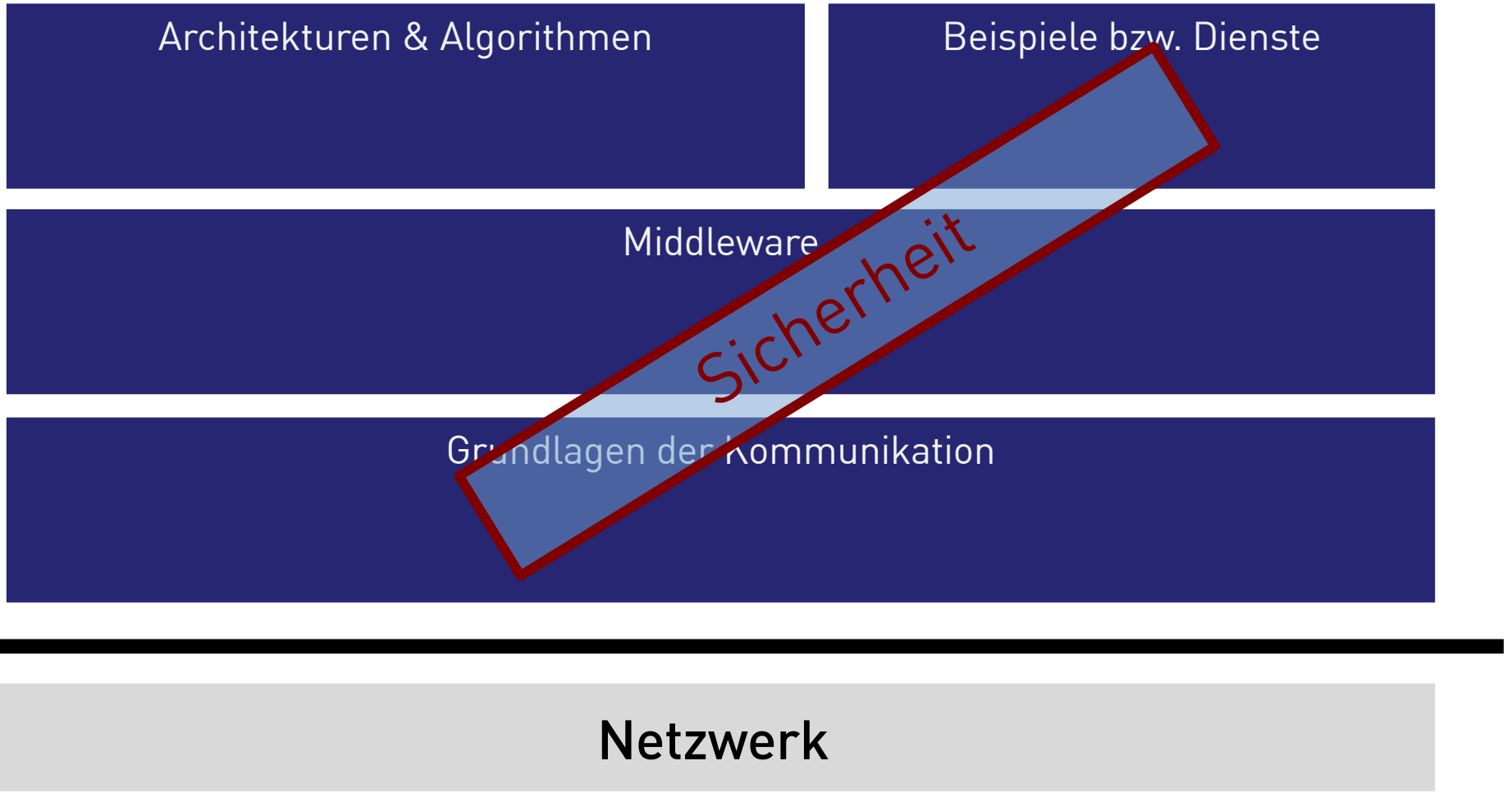
Struktur der Vorlesung

Änderungen vorbehalten!

Stand: 03.04.17

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Aufbau der Vorlesung



h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

VII Zusammenfassung

- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Architekturen
- E Typische Fehlannahmen
- F Architekturübergreifende Themen

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme? (0)

Welche finden Sie interessant?

Welche würden Sie gern besser verstehen?

Nennen Sie ein Paar bekannter Beispiele!

Problem: Fast alle Systeme sind heutzutage verteilt.
Wir müssen die Vorlesung irgendwie *einschränken*.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme?

Laut Tanenbaum & van Steen:

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

* Außer den Skripts von Prof. Burchard, Schütte & Wollenweber, ist unserer Haupttext **Verteilte Systeme, Prinzipien und Paradigmen,** von A. S. Tanenbaum & Maarten van Steen (TvS), 2. Auflage, Pearson Studium, 2008.

Fast so viel verwendet: **Distributed Systems, Concepts and Design, 5th Edition,** von G. Coulouris, J. Dollimore, T. Kindberg and G. Blair (CDKB), 5. Edition, Pearson Education, 2011 (bzw. deutsche Übersetzung der 4. Auflage von Pearson Studium).

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme? (1)

Laut Tanenbaum & van Steen:

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

Unabhängig

- also keine parallele Hochleistungsrechner – mindestens keine mit gemeinsamen Speicher,
- also keine Multi-Core Mikroprozessor,
- also keine homogene Rechner-Cluster, wie z.B. Googleplex?!? *Doch…*

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme? (2)

Laut Tanenbaum & van Steen:

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

Erscheinung: ein einzelnes System

- D.h. die Verteilung wird *verborgen* – ist *transparent.*
- Wenn die Systemkomponenten verborgen sind, können sie *skaliert* werden.

Warum kann das nicht immer erreicht werden?

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme? (3)

Laut Alois Schütte:

Definition: Verteilte Anwendung
Eine verteilte Anwendung ist eine Anwendung, die auf mehreren Rechnern bzw. Prozessoren abläuft und unter diesen Informationen austauscht.

Daraus folgt:

Definition: Verteiltes System
Ein verteiltes System ist ein System mit mehreren Rechnern bzw. Prozessoren, auf denen mindestens eine verteilte Anwendung lauffähig installiert ist.

Implizit:

- Die Rechner sind *vernetzt.*
- Also gibt es irgendwelche Art *Netzwerk.*
- Das Netzwerk kann u.U. *drahtlos* sein.
- Eine verteilte Anwendung muss nur *vorhanden* sein...

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Was sind verteilte Systeme? (4)

Laut Ronald Moore:

Axiom: So gut wie alle „Information-Systems“ sind inzwischen irgendwie verteilt. Die Frage ist, ob die Verteilung *interessant* ist.
Behaptung: Es gibt allerdings genau zwei *interessante* Gründe, ein System zu verteilen:

- Weil manche Komponenten **geographisch entfernt** sind (bzw. sein sollten);
- Weil manche Komponenten **vervielfacht** werden sollten - **Replikation**.

Frage: Wofür ist Replikation gut?

Implizit:

- Verteilte *Anwendungen* werden verwendet.
- Ein Netzwerk ist dabei – die Komponenten kommunizieren
- Beide Gründe können gleichzeitig vorliegen.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

VII Zusammenfassung

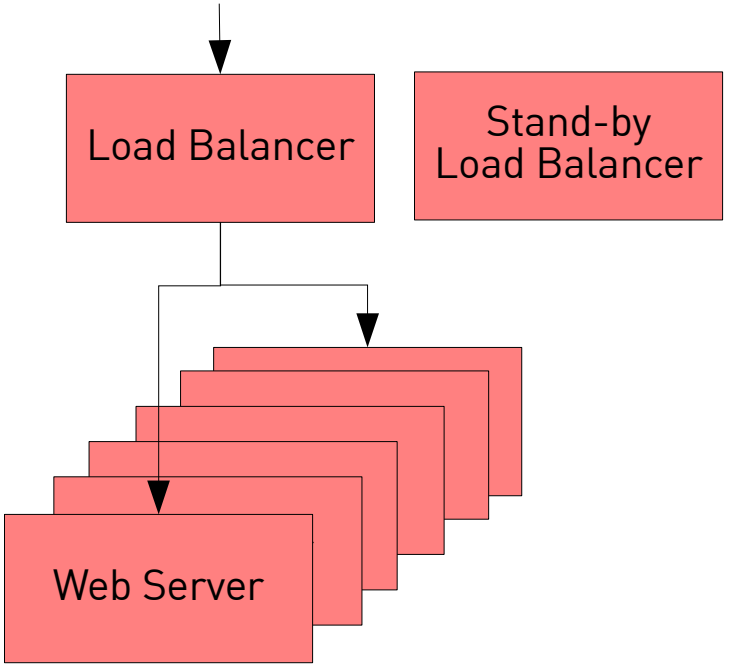
- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Architekturen
- E Typische Fehlannahmen
- F Architekturübergreifende Themen

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Beispiele – Internet, Web Sites

- **Das Internet?** Erscheint nicht wirklich wie *ein* System...
  ...und ist eine Nummer zu groß für diese Vorlesung...
- **Ein Produktion Web-Site** wird oft von mehreren Web-Server produziert:

  Verschiedene Dienste:
  - 2 (oder mehr) Load Balancers
  - Viele Web Servers
  - Datenbank Servers
  - Andere Datenquellen



Nun stellen Sie sich vor, wie Google diese Architektur skalierte, sodass sie www.google.com betreiben können!?!

Beispiele - Google

- **Google:** Betreibt mehrere Rechnerzentren mit insgesamt weit mehr als 200.000 PC's

  Verschiedene Dienste:
  - Load Balancers
  - Proxy Servers (caches)
  - Web Servers
  - Index servers
  - Document servers
  - Data-gathering servers
  - Ad servers
  - Spelling servers

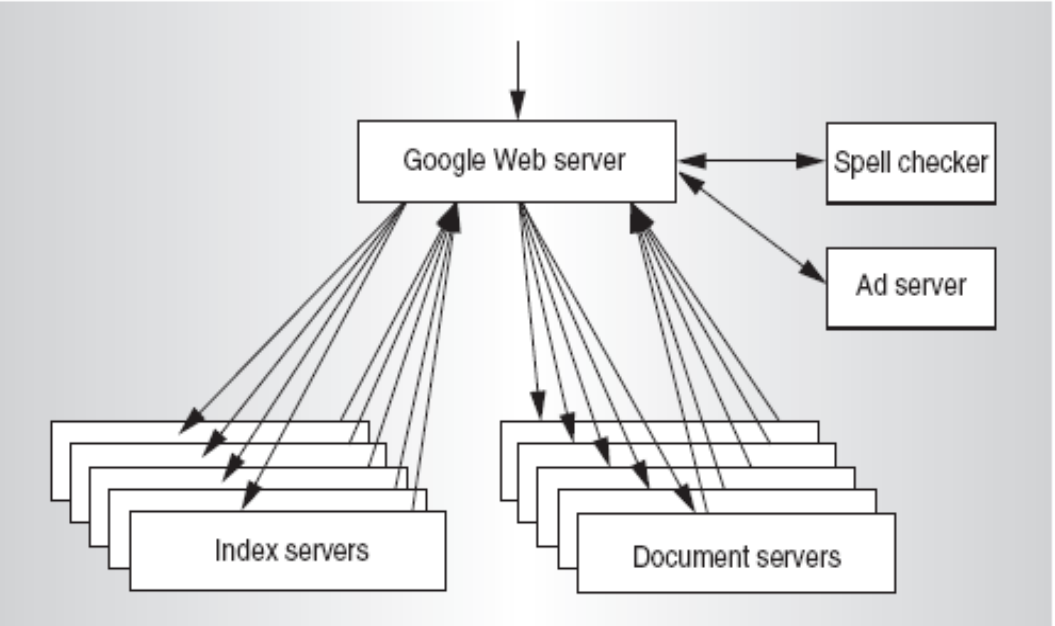
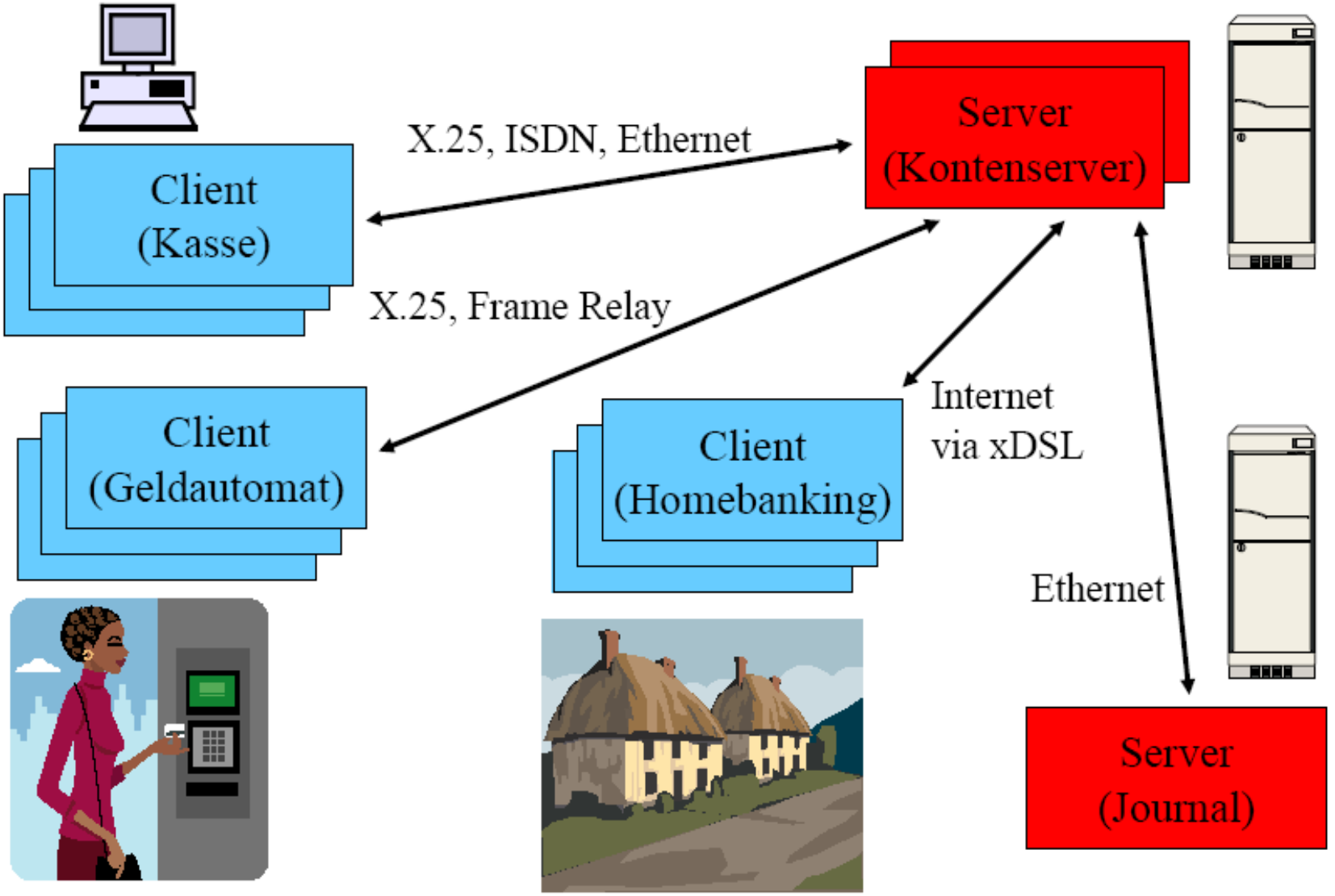


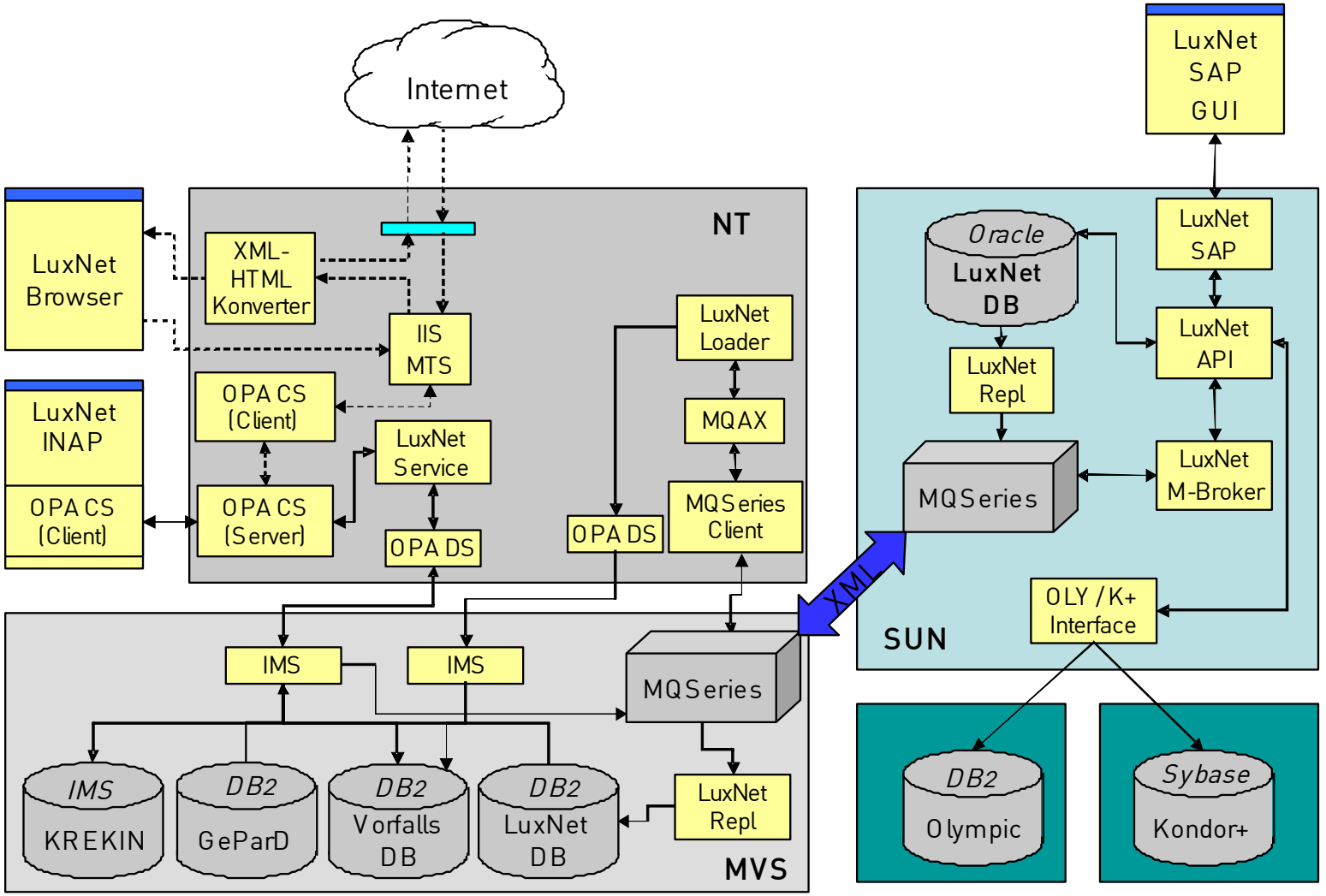
Figure 1. Google query-serving architecture.

Quellen: http://en.wikipedia.org/w/index.php?title=Google_platform&oldid=202504102,, Barroso et. al. *Web Search for a Planet: The Google Cluster Architecture,* **IEEE Micro,** March-April 2003 (http://labs.google.com/papers/googlecluster-ieee.pdf)

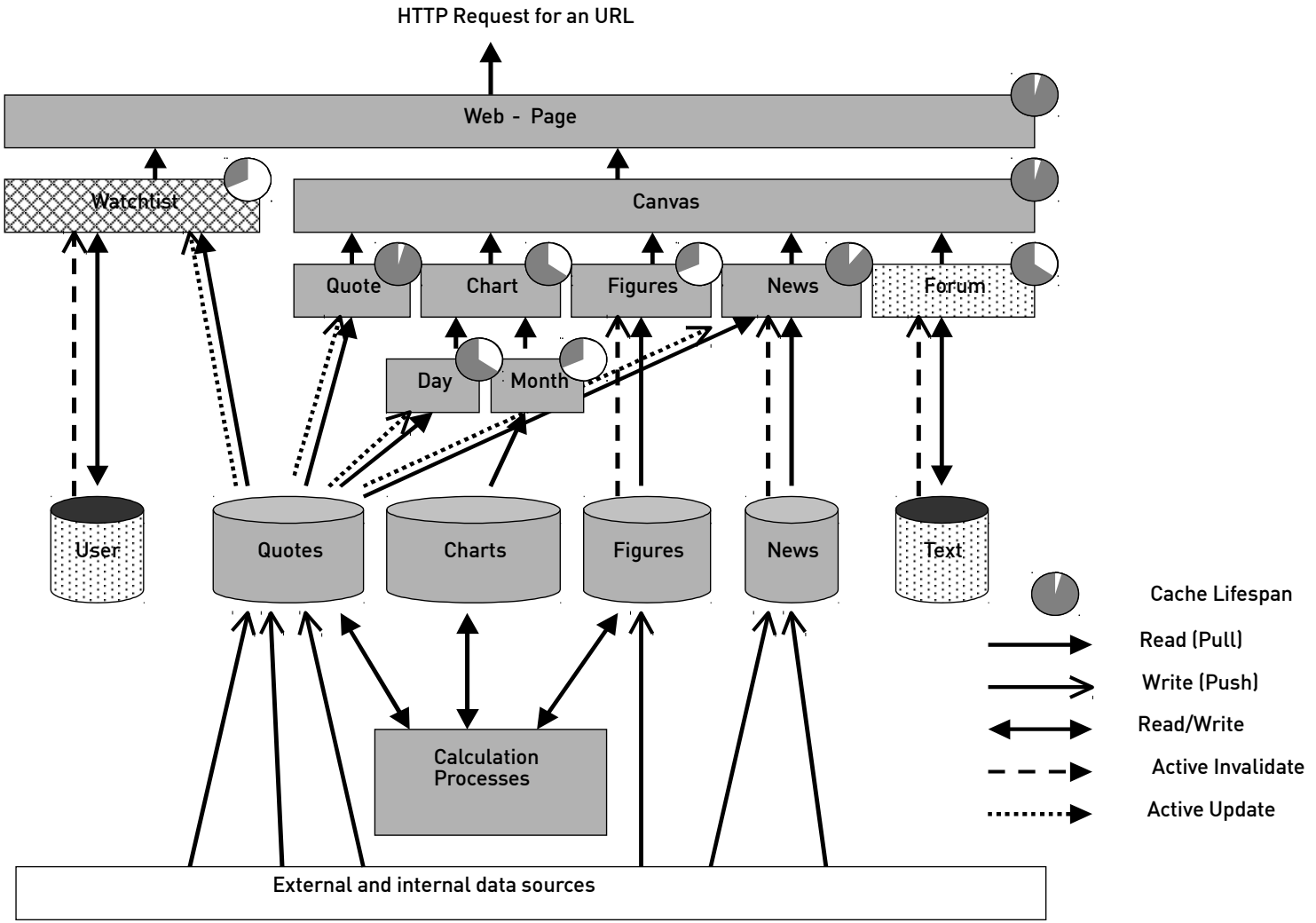
Beispiele – Banking (1)



Beispiele – Banking (3)



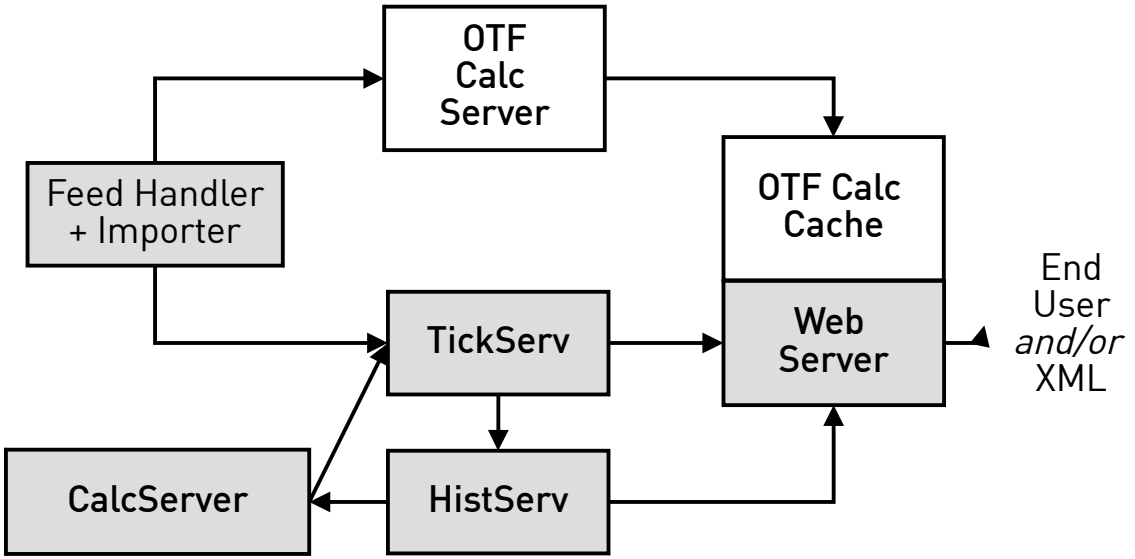
Beispiele – Finanzdaten → Web (1)

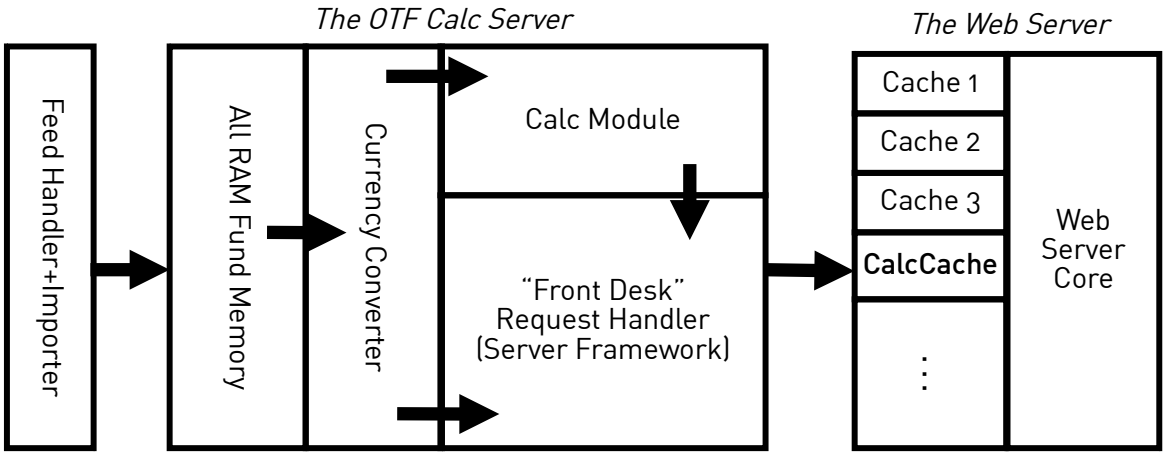


Quelle: Cotoaga, K.; Müller, A.; Müller, R.: *Effiziente Distribution dynamischer Inhalte im Web.* In **Wirtschaftsinformatik** 44 (2002) 3, p. 249-259.

Beispiele – Finanzdaten → Web (2)

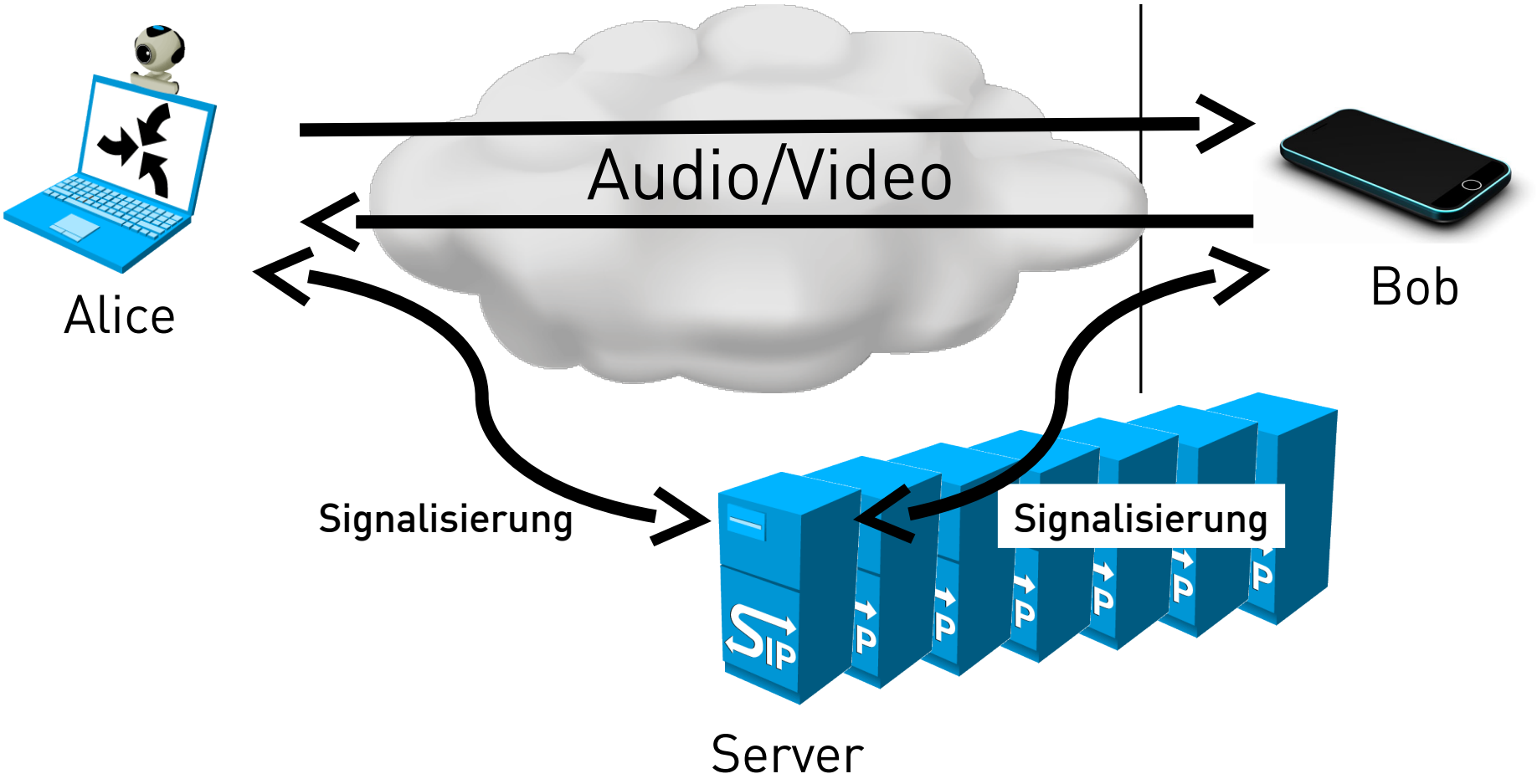
Eine Erweiterung:
On The Fly
Berechnungen:



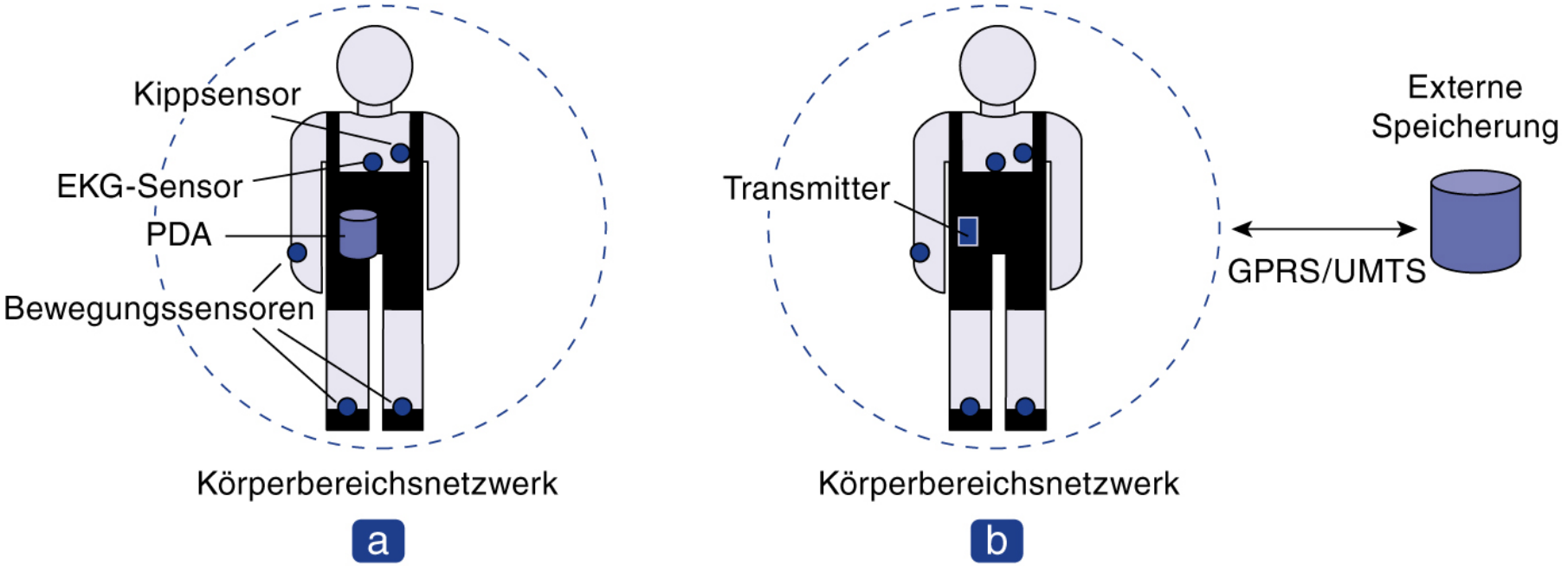


Quelle: Moore, Müller, Müller, Temmen, *Adapting eFinace Web Server Farms to Changing Market Demands.* In **Informatik 2003, Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesselschaft für Informatik e. V. (GI)**, 2003.

Beispiele – Voice-Over-IP (VoIP)

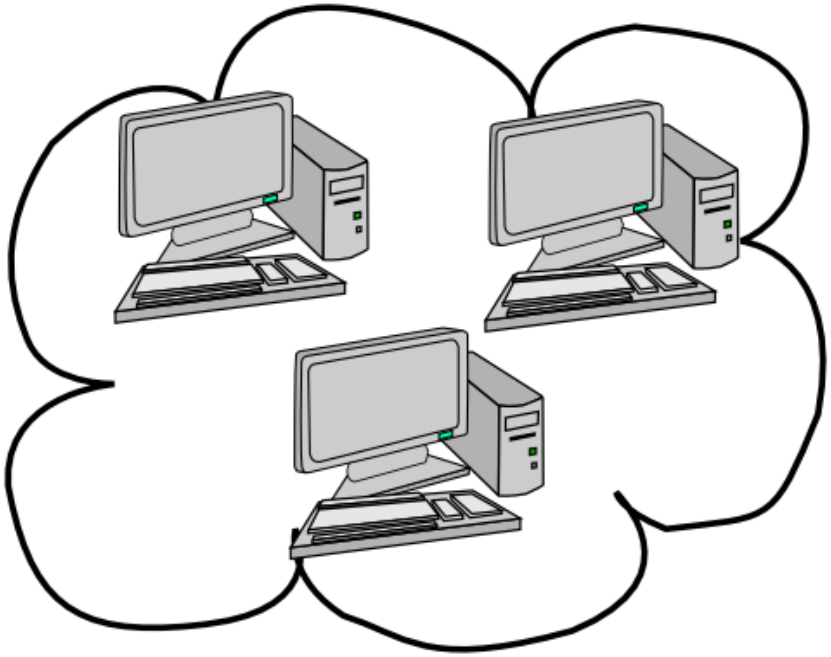


Beispiele – Body Area Networks



Überwachung einer Person in einem pervasiven elektronischen System zur Gesundheitsvorsorge mithilfe (a) eines lokalen Hubs und (b) einer kontinuierlichen kabellosen Verbindung. (TvS, S. 44).

Neue Beispiele – BOINC, Grids & Clouds



Quelle: www.gridcafe.org

BOINC = Berkeley Open Infrastructure for Network Computing = Offene Infrastruktur, womit verteilte Anwendungen – z.B. SETI@home – sonst unbenutzte Rechnerzeit verwenden können.

Grids = Infrastruktur, um Zusammenarbeit zwischen verteilten Rechner zu ermöglichen = ein Form vom *Utility Computing.* Schwerpunkt: Forschung.

Cloud-Computing = Infrastruktur, um verteilte *virtuelle* Rechner zu *vermieten* = ein andere Form vom *Utility Computing* . Schwerpunkt: Kommerz.

Aktuelle Schlagwörter: Google Map/Reduce, Apache Hadoop, Spark, Beam & Co...

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

VII Zusammenfassung

- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Typische Fehlannahmen
- E Architekturen
- F Architekturübergreifende Themen

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Lehrbuch-Ziele

Ein *gutes* verteiltes System hat (liefert, unterstützt):

- **Zugriff auf Ressourcen:**
  Ein verteiltes System sollte Ressourcen leicht zugreifbar machen.
- **Verteilungstransparenz:**
  Es sollte die Tatsache vernünftig verbergen, dass Ressourcen über ein Netzwerk verteilt sind.
- **Offenheit:**
  Es sollte offen sein – es sollte *Heterogenität* erlauben.
- **Skalierbarkeit:**
  Es sollte skalierbar sein – es sollte *wachsen* können.
- **Zuverlässigkeit:**
  Das System sollte (fast) immer verfügbar sein

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Ziele – Zugriff auf Ressourcen

- **Zugriff auf Ressourcen**
- Verteilungstransparenz
- Offenheit
- Skalierbarkeit
- Zuverlässigkeit

Verteilte Systeme sollen es ermöglichen, dass Ressourcen gemeinsam genutzt werden können:

- Hardwarekomponenten wie Drucker, Platten und Prozessoren
- gemeinsames Nutzen von **Daten** und **Diensten**

Folgende Problematik tritt dabei auf:

- Regelung nebenläufiger Zugriffe,
- Fragen der **Konsistenz** und der **Fehlertoleranz**.

„Zugriff auf Ressourcen“ ist als Ziel (Eigenschaft) trivial bis nichts aussagend. Jedes System hat es.

Die Frage ist viel mehr: Auf *welche Ressourcen* wollen wir zugreifen?

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Ziele – Transparenz

- Zugriff auf Ressourcen
- **Verteilungstransparenz**
- Offenheit
- Ska
- Zuv

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

Von ISO/IEC IS10746, 1995 (vgl. TvS, S. 22).

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Ziele – Transparenz?!?

- Zugriff auf Ressourcen
- **Verteilungstransparenz**
- Offenheit
- Ska
- Zuv

Transparenz	Beschreibung
Zugriff	Verbirgt Unterschiede in der Datendarstellung und die Art und Weise, wie auf eine Ressource zugegriffen wird
Ort[1]	Verbirgt,
Migration	Verbirgt,
Relokation	Verbirgt, während
Replikation	Verbirgt,
Nebenläufigkeit	Verbirgt, gleichzeit
Fehler	Verbirgt

Offene Fragen:

- Ist Verteilunstransparenz immer eine gute Idee?
- **Hinweis:** In wie weit ist sie überhaupt möglich?
- Welcher **Grad der Transparenz** ist angemessen?

Von ISO/IEC IS10746, 1995 (vgl. Tanenbaum & van Steen, S. 22).

Ziele – Offenheit

- Zugriff auf Ressourcen
- Verteilungstransparenz
- **Offenheit**
- Skalierbarkeit
- Zuverlässigkeit

Es geht hier eigentlich um Menschen!
- verschiedene MitarbeiterInnen
- zur verschiedenen Zeiten (Zukunftsfähigkeit!)
- von verschiedenen Firmen!

Heißt „Offenheit“, dass es „Open Source“ sein muss?

- *Portabilität* – Komponenten können ausgetauscht werden
- *Interoperabilität* – Heterogene Komponenten können gemischt werden.
- *Erweiterbarkeit* – durch Hinzufügen bzw. Ersetzen von Komponenten.
- Die *Schnittstellen* der Hardware- und Software-komponenten müssen offen gelegt werden.
- Die Interprozesskommunikation auf Anwendungsebene muss einheitlich und offen gelegt sein.
- Dazu werden *Schnittstellendefinitions-sprachen (Interface Definition Languages, IDLs)* verwendet (u.a.).

Ziele – Skalierbarkeit (1)

- Zugriff auf Ressourcen
- Verteilungstransparenz
- Offenheit
- **Skalierbarkeit**
- Zuverlässigkeit

Warnung! Das sind nicht Beispiele von Skalierbarkeit, sondern **Gegenbeispiele**!

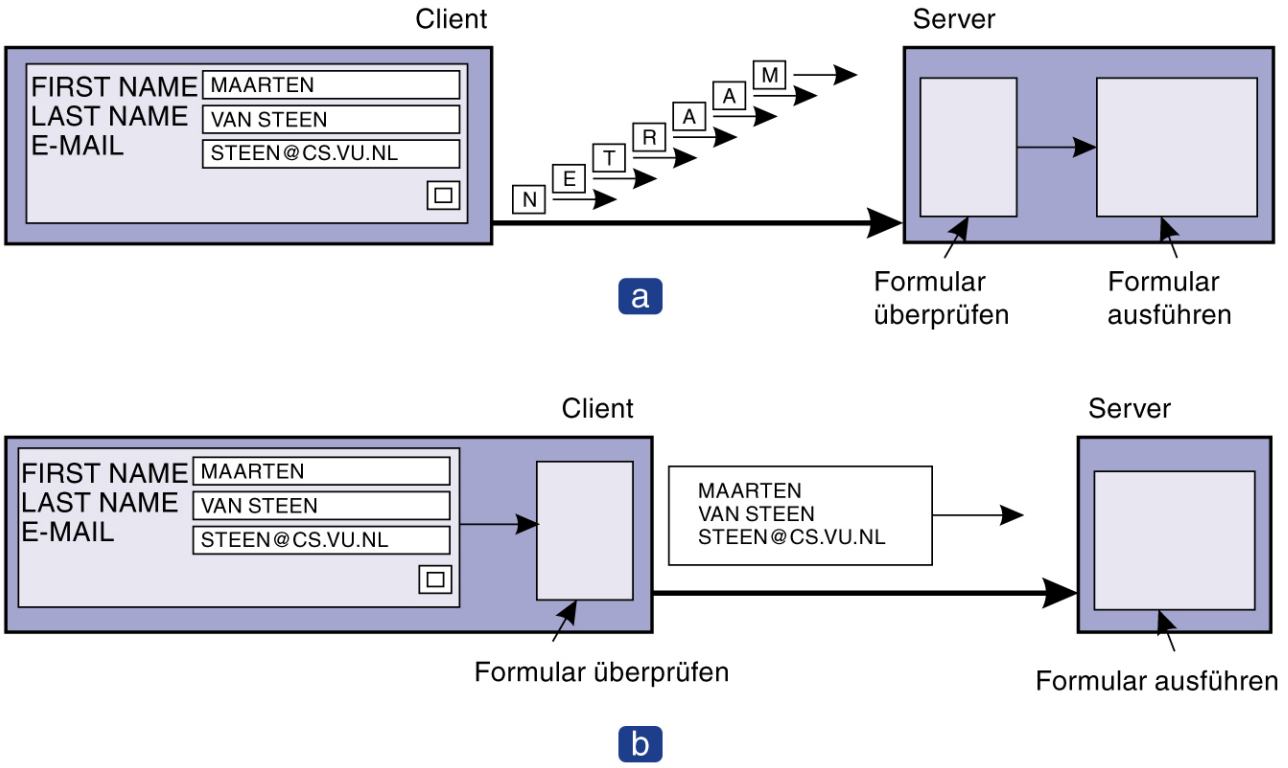
Konzept	Beispiel
Zentralisierte Dienste	Ein einziger Server für alle Benutzer
Zentralisierte Daten	Ein einziges Online-Telefonbuch
Zentralisierte Algorithmen	Routing (Weiterleitung) aufgrund vollständiger Informationen

Beispiele für **Beschränkungen** an Skalierbarkeit (TvS, S. 27).

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Ziele – Skalierbarkeit (2)

- Zugriff auf Ressourcen
- Verteilungstransparenz
- Offenheit
- **Skalierbarkeit**
- Zuverlässigkeit

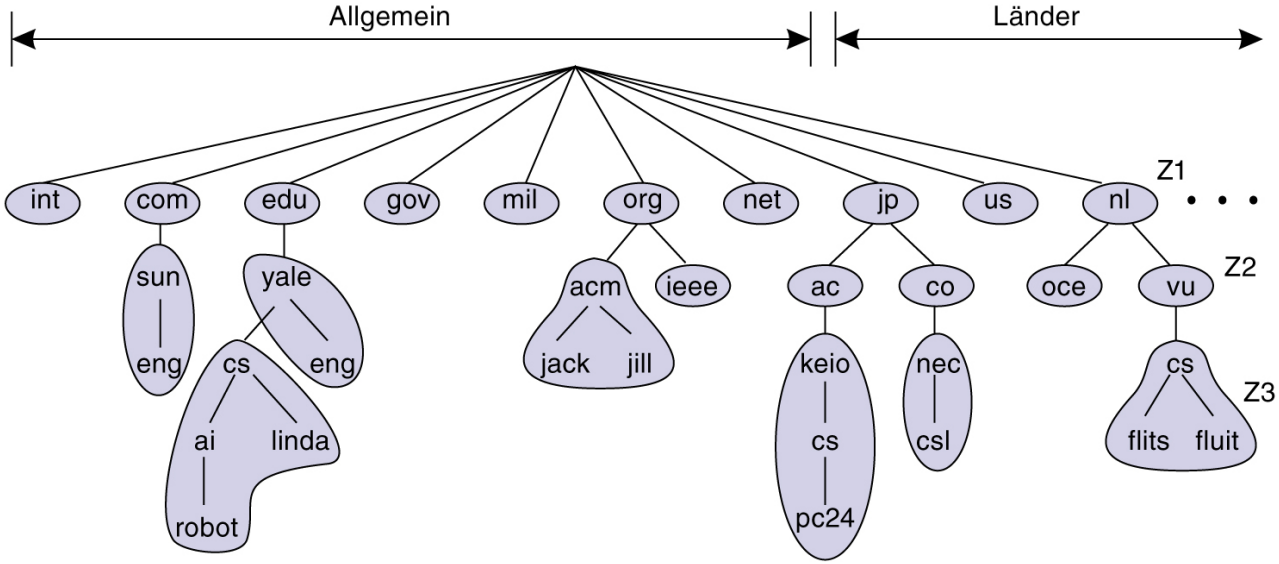


Eine Verbesserung? Sind Sie damit einverstanden?

Beispiel für die Verbesserung der Skalierbarkeit: Überprüfung beim Ausfüllen von Formularen (a) auf dem Server oder (b) auf dem Client (TvS, S. 30).

Ziele – Skalierbarkeit (3)

- Zugriff auf Ressourcen
- Verteilungstransparenz
- Offenheit
- **Skalierbarkeit**
- Zuverlässigkeit



Beispiel für Skalierbarkeit: Die Unterteilung des DNS-Namensraumes in Zonen (Tanenbaum & van Steen, S. 31).

Ziele – Zuverlässigkeit

- Zugriff auf Ressourcen
- Verteilungstransparenz
- Offenheit
- Skalierbarkeit
- **Zuverlässigkeit**

- **Verfügbarkeit** wird zunehmend in SLA's – Service Level Agreements – verträglich geregelt!
- **Fehlertoleranz** = in Hinsicht auf Hardware, Software & Benutzung!
  - Skalierbarkeit ist nicht nur für *Performance*, sondern auch für *Redundanz*
  - *Keine Single Points of Failure*
  - *Fail Safe* wo möglich
  - *IT-Sicherheit* spielt hier auch eine Rolle!

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Ziele – Zuverlässigkeit

- Zugriff auf Ressourcen
- Verteilungstransparenz
- Offenheit
- Skalierbarkeit
- **Zuverlässigkeit**

Definition: **Verteiltes System**
(nach Leslie Lamport)
"Ein verteiltes System ist ein System, mit dem ich nicht arbeiten kann, weil irgendein Rechner abgestürzt ist, *von dem ich nicht einmal weiß, dass es ihn überhaupt gibt.*"

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Drei widerspruchliche Ziele

Diese drei Ziele werden durch alle Kapiten uns begleiten!

1. Ziel: Baue Systeme im Raum und Zeit

Raum – über grosse Entfernungen.

Zeit – Maximiere *Performance* (und Verfügbarkeit!)

2. Ziel: Niedrige Entwicklungs-/Betriebskosten

IT-Systems sind schwer zu bauen, schwer zu verstehen und sehr schwer zu debuggen! Wir müssen Time-to-market und Betriebskosten im Griff bekommen.

3. Ziel: Heterogenität überleben bzw. anbieten!

Verschiedene Systeme, die vielleicht zu verschiedenen Firmen gehören (!), sollen zusammen arbeiten.

Wir wollen *Hardware, Software und Daten über Grenzen hinweg* zusammenarbeiten lassen.

Offenne Frage: Kann man alle drei haben?

Hörsaalübung – Klausur Vorschau

Die letzte Frage auf die Klausur vom Wintersemester 2015/16 lautete:

Gehen Sie von folgender Aufgabenstellung aus: Sie sind Berater und sollen Ihrem Auftraggeber helfen, ein verteiltes System zu planen bzw. die Machbarkeit zu analysieren.

Der Auftraggeber möchte eine *Industrie 4.0,* automatisierte Fabrik bauen. Wir gehen davon aus, dass die Fabrik mehrere Produkte produzieren kann. Die Idee ist, dass Läden in Echtzeit melden sollen, welche Produkte gerade verkauft werden. Produkte, die verkauft werden, werden weiter von der Fabrik produziert, während die Produktion von Produkten, die nicht gekauft werden, eingestellt wird.

Mindestens folgende Rollen bzw. Komponenten sind vorgesehen:

- **„Fabrik-Sensoren“:** Sie melden, wie viele Produkte produziert werden, wie viele schon produziert aber noch nicht ausgeliefert wurden, und welche Teilen der Fabrik Probleme haben und daher still stehen.
- **„Läden“:** Sie verkaufen Produkte. Sie sind Kunden Ihres Auftraggebers und melden jeden Verkauf.
- **„Logistik“:** Sie holt Produkte von der Fabrik ab und liefert sie an die Läden aus. Die Logistik-Firmen sind Partner des Fabrik-Besitzers.
- **„Prognose“:** Die Data-Miner analysieren die Verkaufsdaten der letzen Tage und Wochen, um Trends zu errechnen und Prognosen zu erstellen.

Sie dürfen weitere Komponenten vorschlagen.

...

Hörsaalübung – Klausur Vorschau fortgesetzt

Die letzte Frage auf die Klausur vom Wintersemester 2015/16, weiter:

...

1) Architektur: Skizzieren Sie eine Architektur für das System. (5 Punkte)

Nehmen Sie so viel Platz, wie Sie brauchen

2) **Anforderungsanalyse:** Welche Ziele sollten bei der Planung des verteilten Systems berücksichtigt werden? Welche Schwierigkeiten sind zu erwarten? Erklären Sie Ihre Auswahl (7 Punkte).

Nehmen Sie auch hier so viel Platz, wie Sie brauchen

Wir kömmen wieder hier zurück nach Kapitel 2 bzw. 3.

3) Implementierung: Welche Kommunikationsdienste oder Middleware sollten verwendet werden? Sollte dieselbe Middleware überall im System verwendet werden? Welche Alternativen gibt es? Welche sind am besten geeignet? (8 Punkte)

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

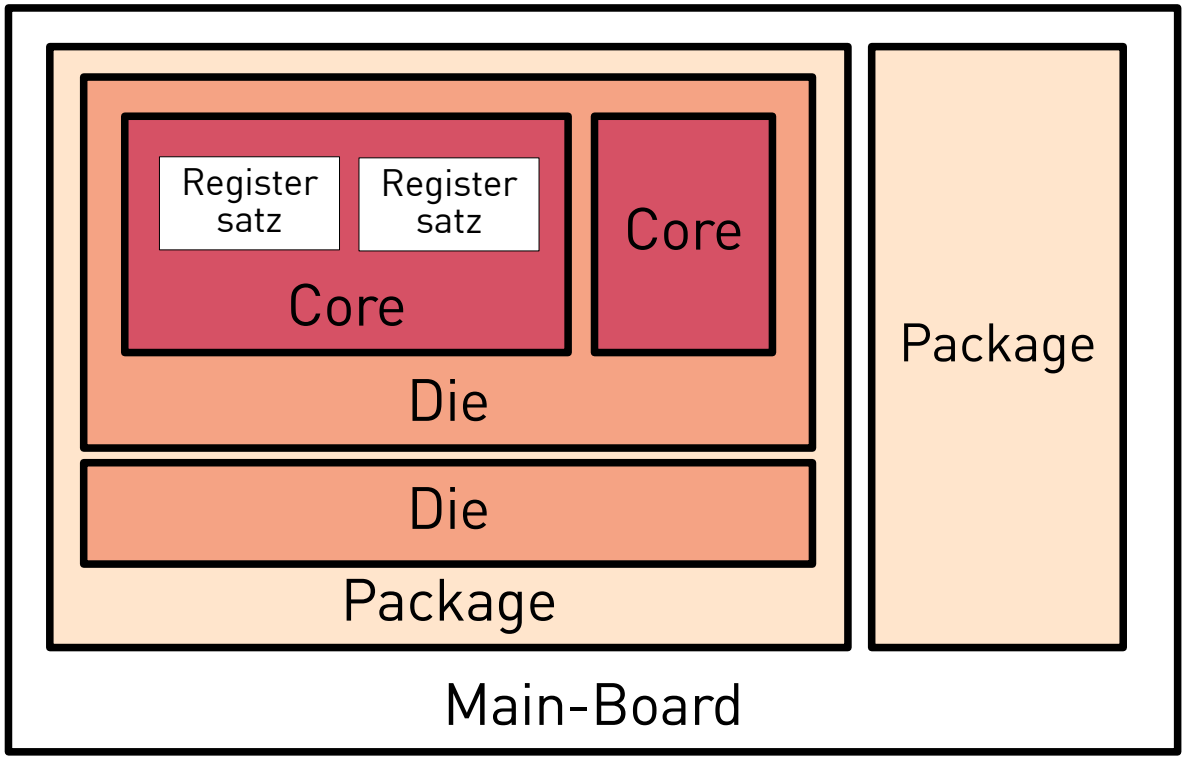
VII Zusammenfassung

- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Architekturen
- E Typische Fehlannahmen
- F Architekturübergreifende Themen

Multiprozessor-Architektur (Server)

Computer = Multiprozessor **X** Multicore **X** SMT

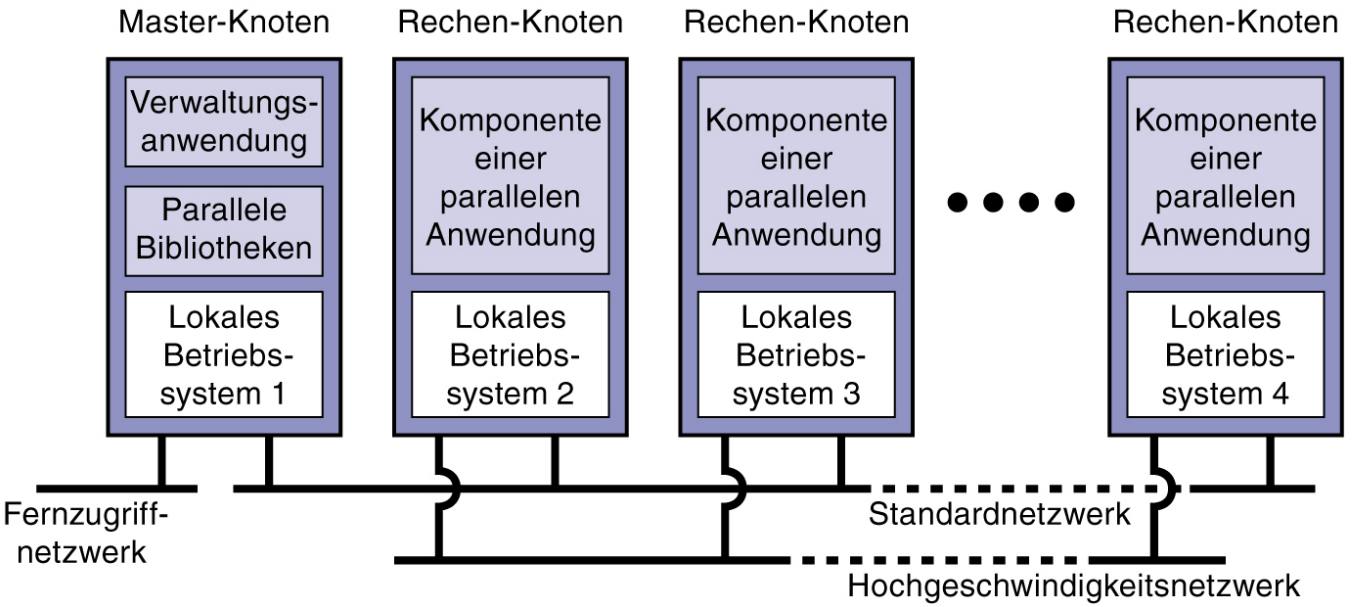
- Es kann mehrere CPU-*Gehäuse* (Sockets) an einem *Main-Board* geben.
- Es können mehrere *Dice* in einem *Gehäuse* geben.
- Es können mehrere *Cores* auf einer *Die* geben.
- Manche Cores können SMT = *Simultaneous Multithreading* – und damit mehrere Cores *simulieren*.
- **In jedem Fall:** Alle CPUs in einem Multiprozessor teilen:
  - gemeinsamen Speicher,
  - gemeinsamen Netzwerkverbindung(en).
  - *Prozesse bzw. Threads*



Multicomputer-Architektur

Ein Beispiel Clustzer-Computersystem

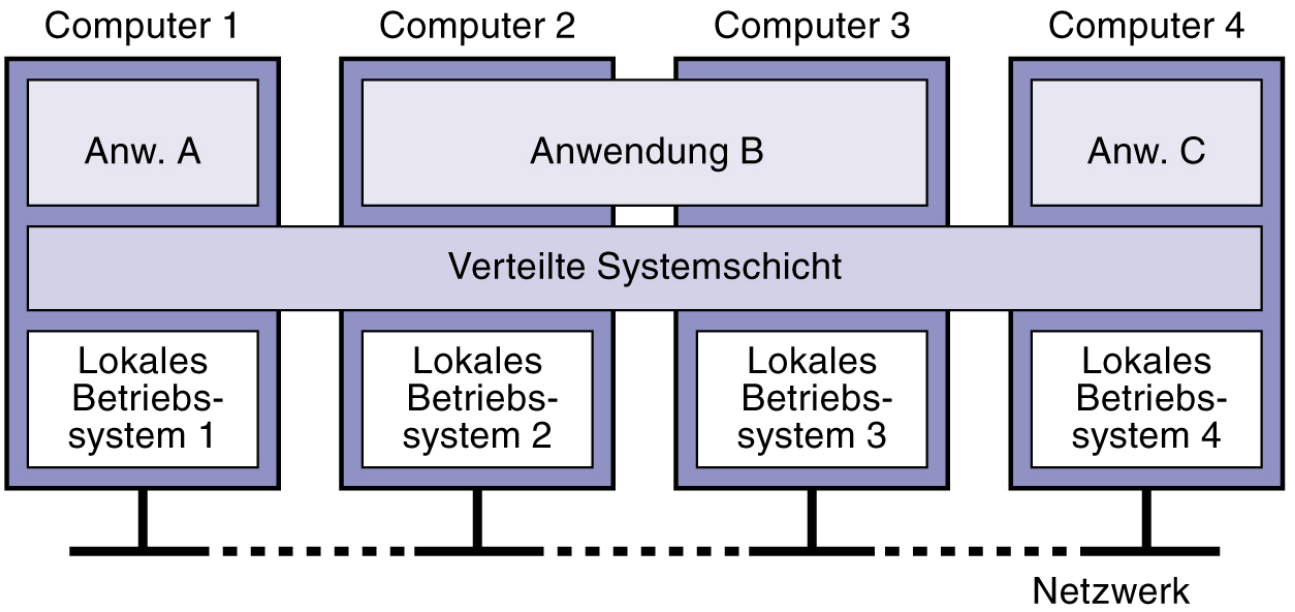
- Oft zwei (oder mehr) Netwerke:
  (1) intern
  (2) extern
- Kompliziertere interne Netzwerke sind auch möglich (z.B. Kreuzschienverteiler).
- Systeme fangen oft homogen an, werden (oft) mit der Zeit heterogen ("legacy" bzw. "installed base")...



Software-Architektur

Heterogenität → *Middleware*

- Wird als Zugriffs-schnittstelle genutzt.
- **Ziel: Verbergen** bzw. **Vereinheitlichung** der Verteilungsaspekte & Kommunikation vor der Anwendung.
- Setzt auf dem Transportprotokoll (z.B. TCP) und der Zugriffsschnittstelle (z.B. Sockets) des verteilten Systems auf.

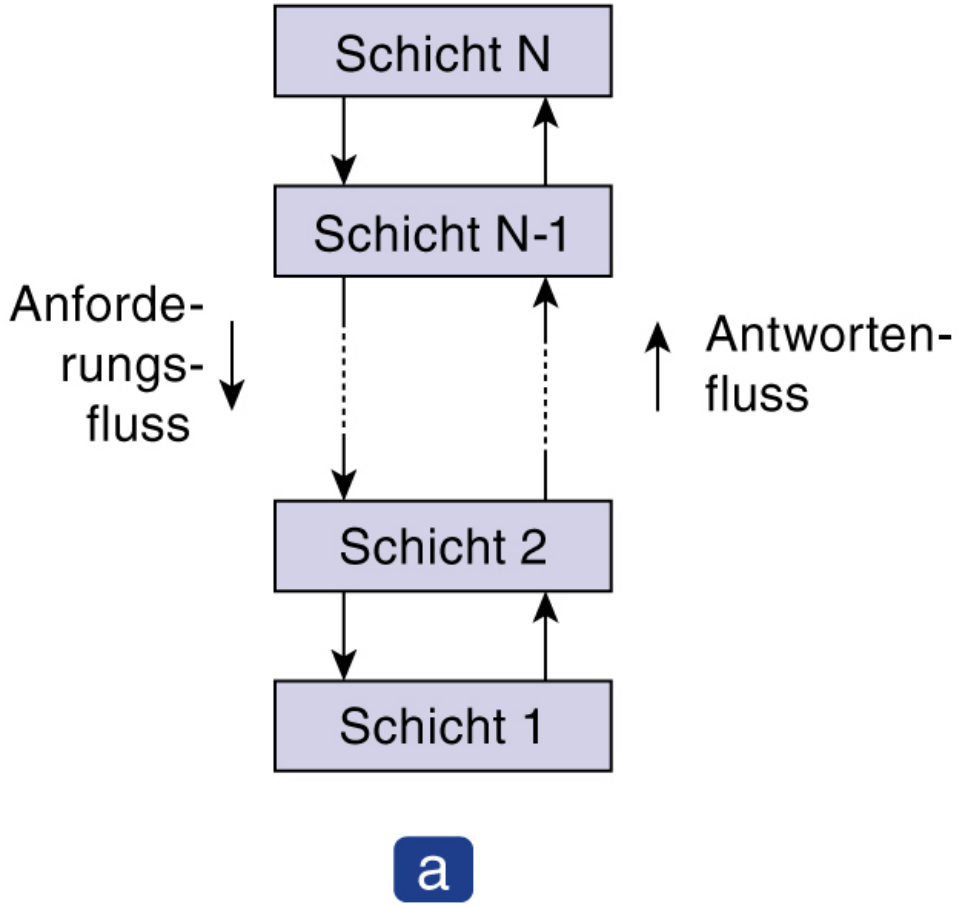


- **Beispiele:**
  - Kommunikations-orientiert: Sun RPC, Java RMI, IBM WebSphere MQ...
  - Anwendungs-orientiert: CORBA, J2EE, .Net

System-Architektur: Geschichtet

Vier Architekturstile *(laut TvS)*

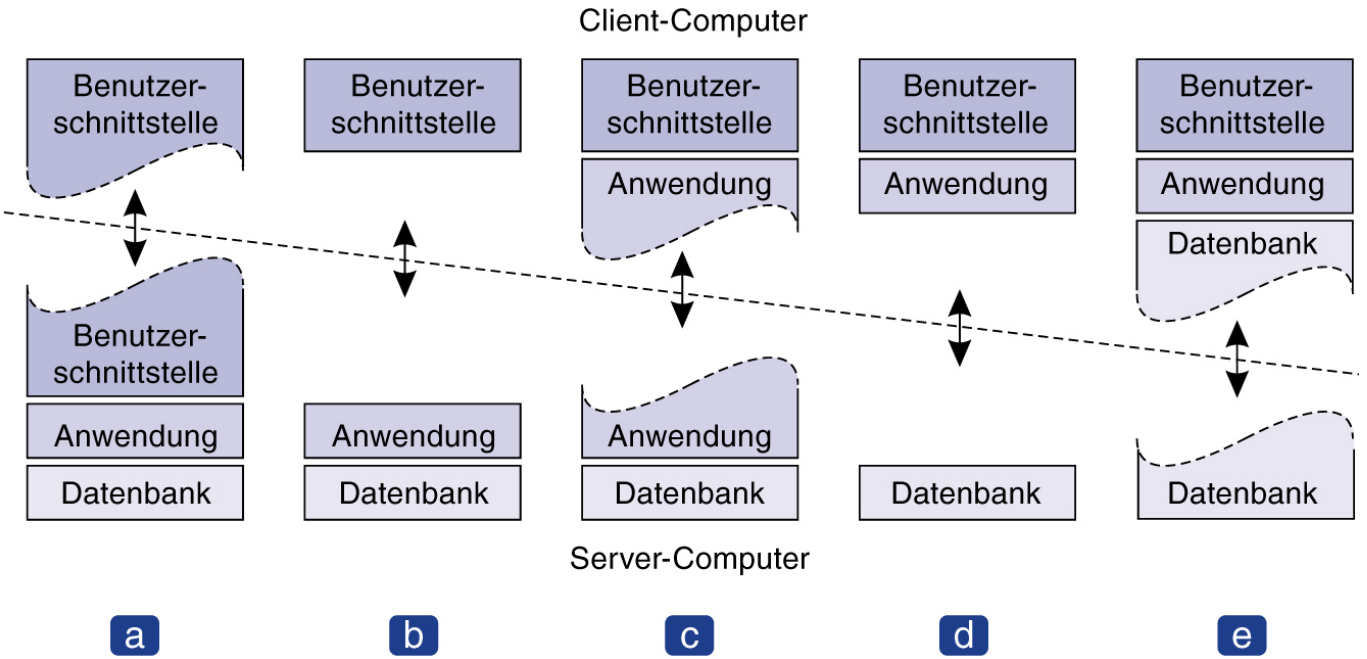
1. Geschichtet



System-Architektur: Client/Server

Vier Architekturstile

1. Geschichtet
   A. Client/Server

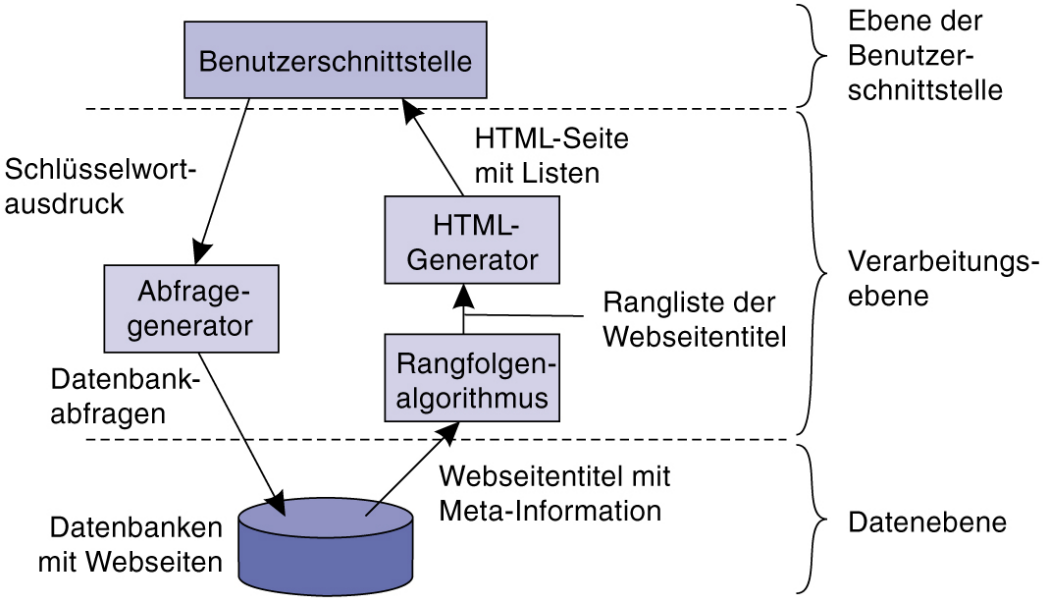


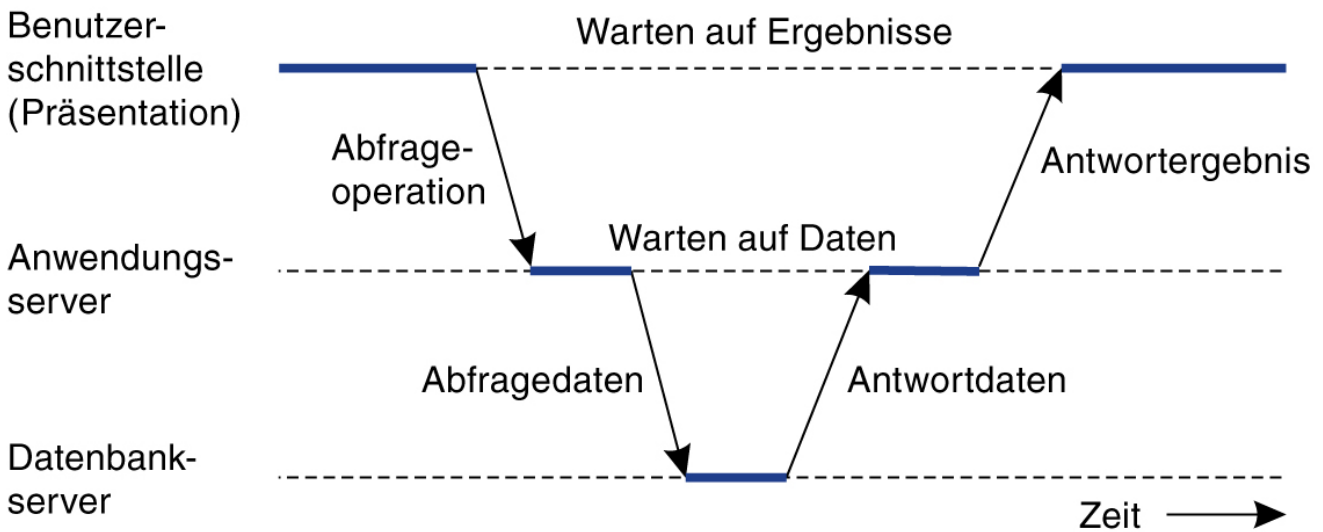
5 Alternative Client/Server Anordnungen

System-Architektur: Client/Server

Vier Architekturstile

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver

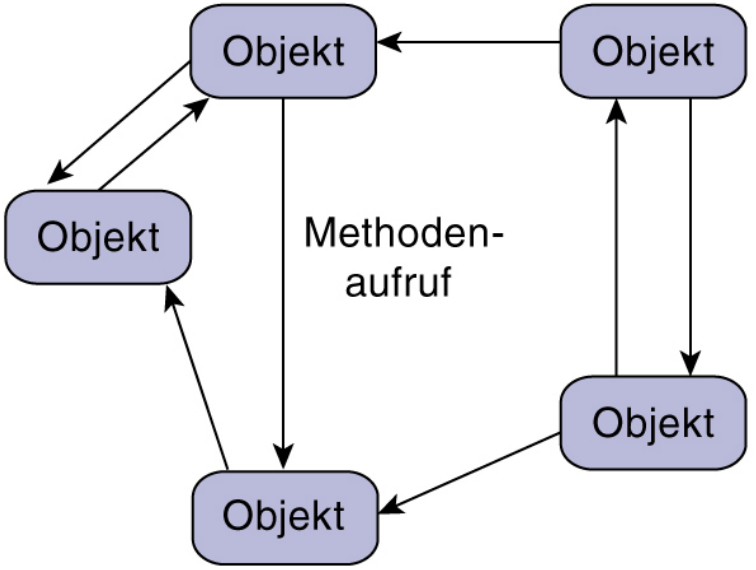




System-Architektur: Objektbasiert

Vier Architekturstile

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert – bzw *Service Oriented*



b

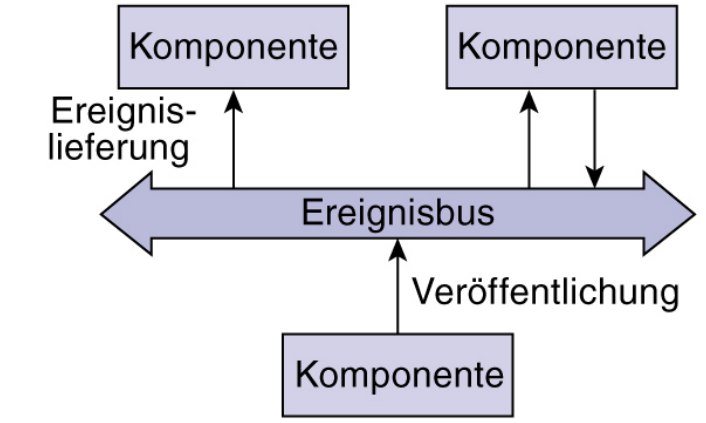
Hardware Architecture ≠ Software Architecture
≠ System Architecture

System-Architektur: Daten- bzw. Ereigniszentriet

Vier Architekturstile

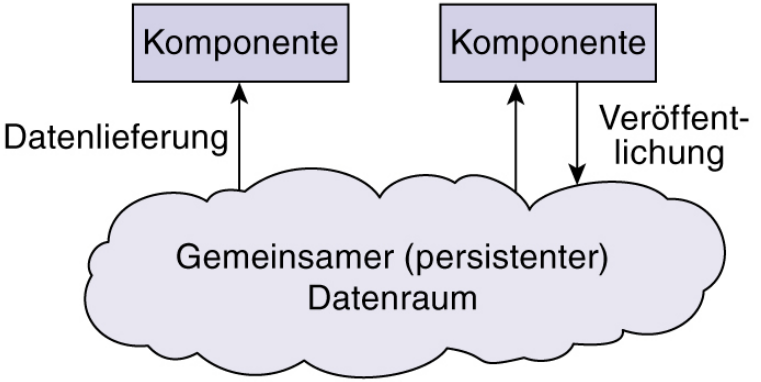
1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert

3 & 4 können in *Publish/Subscribe* Systeme kombiniert werden.



a

Ereignisbasierte Architektur



b

Datenzentrierte Architektur

System-Architektur: Überblick / Zusammenfassung

Vier Architekturstile

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert

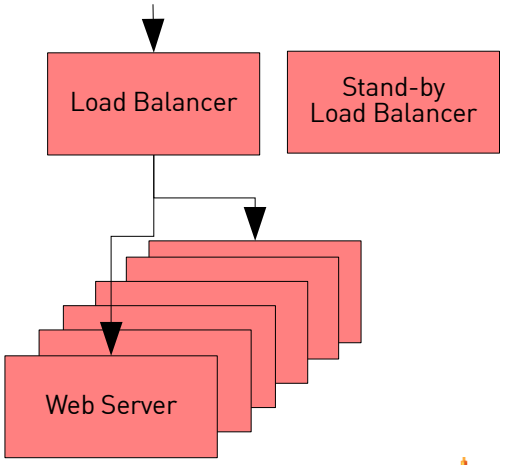
3 & 4 können in *Publish/Subscribe* Systeme kombiniert werden.

	Wenig Middleware	***Virtualisierung mittels Middleware***
Abfrage / Antwort Dienste	Geschichtete Architekturstil	Objektbasierte Architekturstil
Message / Ereignis-gesteuert	Ereignisbasierte Architekturstil	Datenzentrierte Architekturstil

System-Architektur: Übungen (1)

Welche Beispiel verwendet welche Architekturstil(e)?

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert



Web Site

Google's Web Site

- Load Balancers
- Proxy Servers (caches)
- Web Servers
- Index servers
- Document servers
- Data-gathering servers
- Ad servers
- Spelling servers

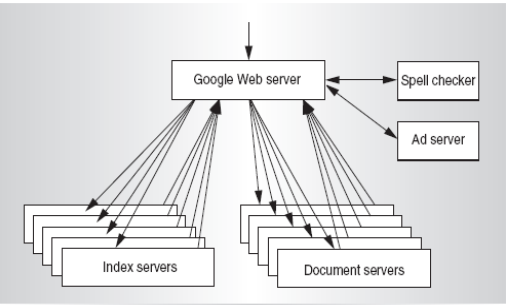
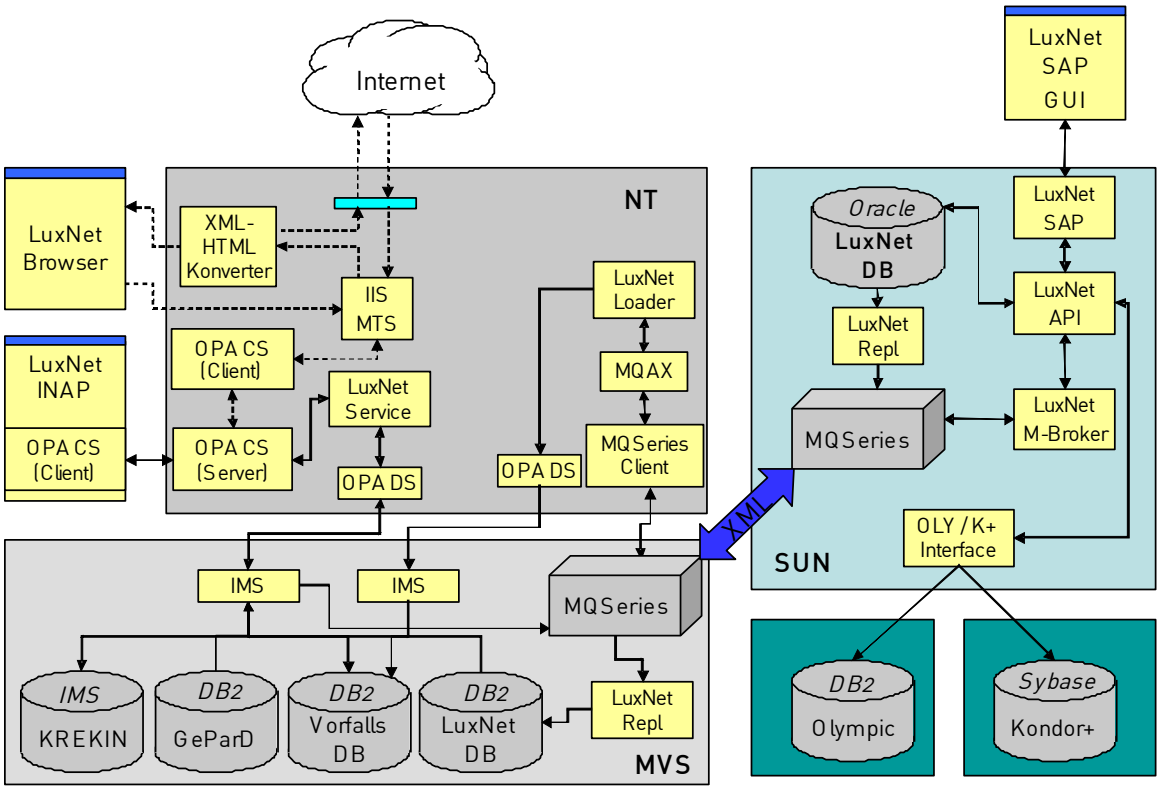


Figure 1. Google query-serving architecture.

System-Architektur: Übungen (2)

Welche Beispiel verwendet welche Architekturstil(e)?

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert

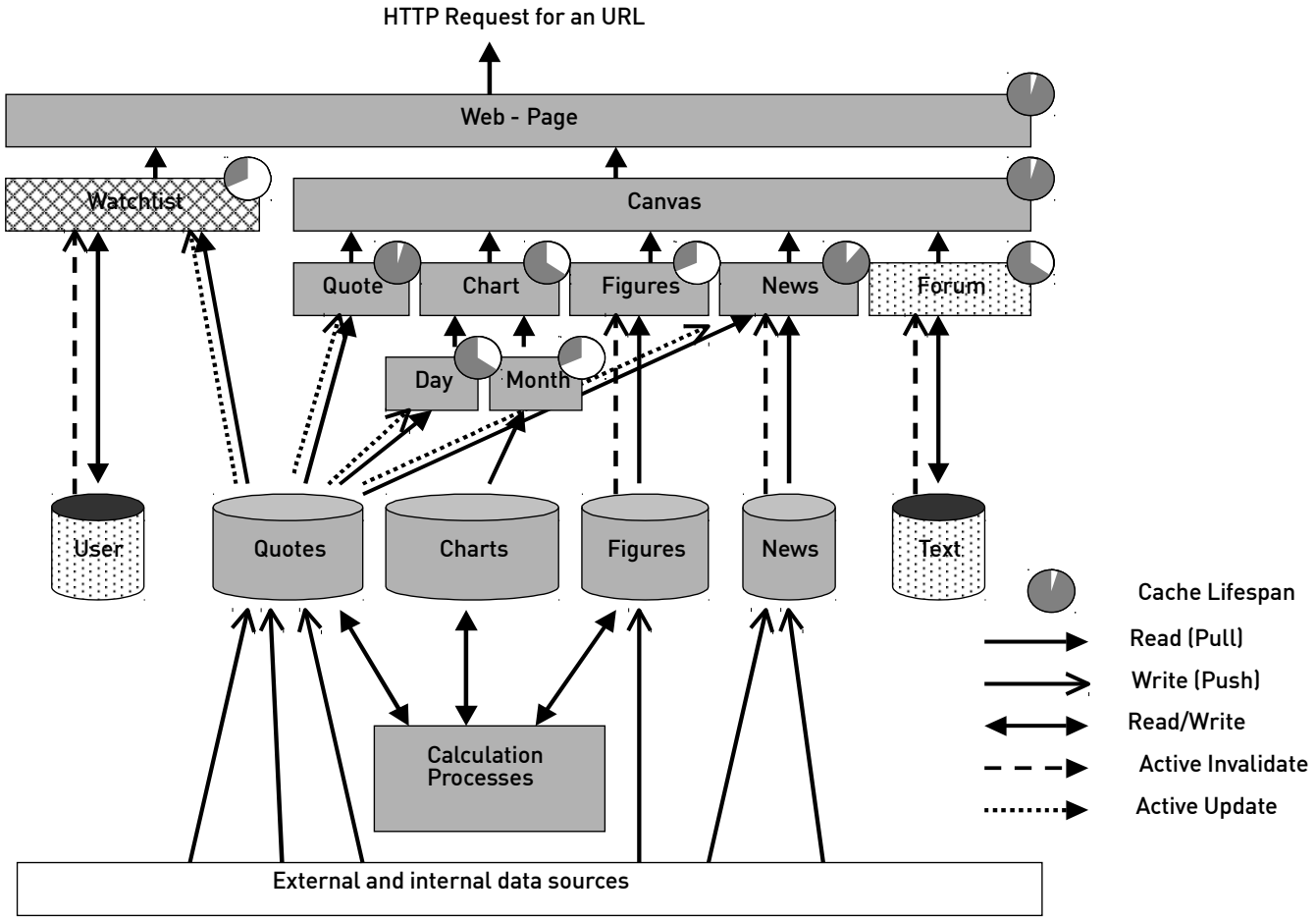


Rechnerzentrum

System-Architektur: Übungen (3)

Welche Beispiel verwendet welche Architekturstil(e)?

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert

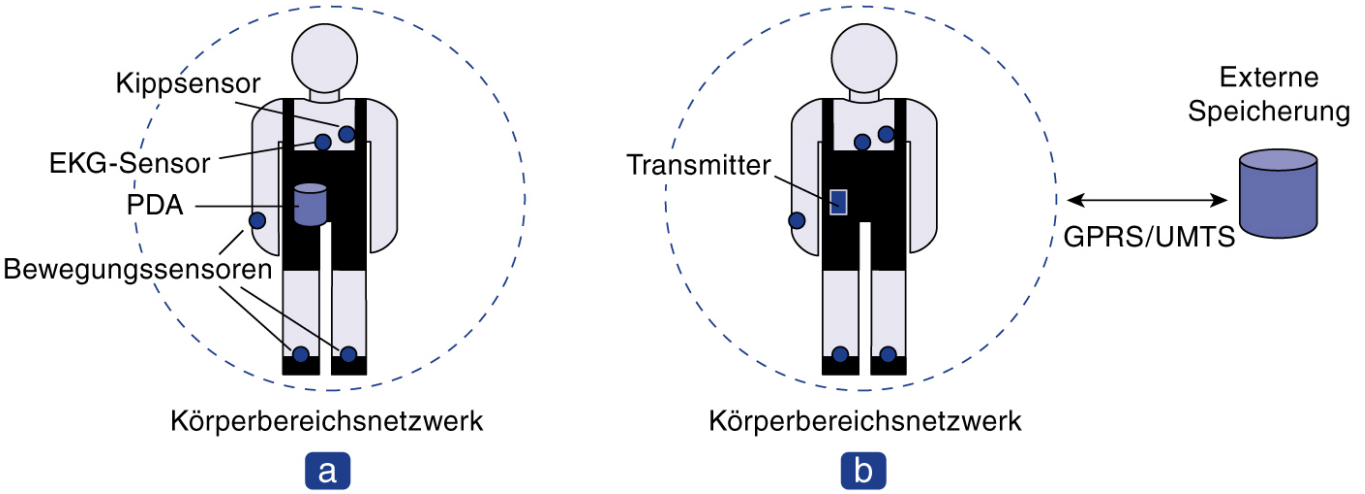


Finanzdatenanwendung

System-Architektur: Übungen (4)

Welche Beispiel verwendet welche Architekturstil(e)?

1. Geschichtet
   A. Client/Server
   B. Multitier bzw. Multiserver
2. Objektbasiert– bzw *Service Oriented*
3. Ereignisbasiert
4. Datenzentriert



Body Area network

System-Architektur: laut R. Moore

Drei Modelle von verteilte Systeme *(laut R. Moore)*

1. **Service Oriented Architecture (SOA):**
   - » Anfrage / Antwort pro Definition
   - » Modularität durch Dienst-Definitionen
2. **Ereignis Gesteuerte Systeme**
   - » „Fire and Forget“
   - » Weder Anfrage noch Antwort, nur Updates
3. **Big Data Systeme**
   - » Datenfluß-zentriert bzw.
   - » Gehen von Unmengen von all-gegenwärtige Daten aus

Besonders wichtig in dieser Vorlesung

Besonders wichtig in dieser Vorlesung

Zunehmend wichtig, jedoch nicht so sehr in dieser Vorlesung

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

VII Zusammenfassung

- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Architekturen
- E Typische Fehlannahmen
- F Architekturübergreifende Themen

Typische Fehlannahmen

- Das Netzwerk ist zuverlässig.
- Das Netzwerk ist sicher.
- Dast Netzwerk ist homogen.
- Die Topologie ändert sich nicht.
- Die Latenzzeit beträgt null.
- Die Bandbreite ist unbegrenzt.
- Die Übertragungskosten betragen null.
- Es gibt genau ein Administrator.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Typische Fehlannahmen vs. Realität

- Das Netzwerk ist zuverlässig.
- Das Netzwerk ist sicher.
- Dast Netzwerk ist homogen.
- Die Topologie ändert sich nicht.
- Die Latenzzeit beträgt null.
- Die Bandbreite ist unbegrenzt.
- Die Übertragungskosten betragen null.
- Es gibt genau ein Administrator.

Realität

- Die Übertragung über ein Netzwerk ist unsicher.
- Nachrichten können verloren gehen oder Bits umkippen.
- Es gibt unbestimmte Nachrichtenlaufzeiten durch unterschiedliches Routing oder veränderliche Übertragungsleistungen.
- Später abgeschickte Nachrichten können vor früher abgeschickten beim Empfänger ankommen.
- Es existiert keine gemeinsame Zeitbasis.
- Der Zustand des Systems ist verteilt.
- Teilkomponenten aus dem Gesamtverbund können ausfallen.

Struktur der Vorlesung

I Einleitung

II Grundlegende Kommunikationsdienste

III Middleware

IV Architekturen & Algorithmen

- A Synchronisierung
- B Konsistenz und Replication
- C Fehlertoleranz

V Beispiele bzw. Dienste

- A Verteilte Dateisysteme
- B Namensdienste

VI Sicherheit & Sicherheitsdienste

VII Zusammenfassung

- A Was ist ein verteiltes System?
- B Beispiele
- C Ziele
- D Architekturen
- E Typische Fehlannahmen
- F Architekturübergreifende Themen

Benennung & Namenssysteme

A Benennung und Namenssysteme
B Synchronisierung
C Konsistenz und Replication
D Fehlertoleranz

Herausforderung:
Wie spricht ein Rechner (ein Prozess, eine Ressource) einen anderen an, ohne Verteilungstransparenz bzw. Skalierbarkeit zu verlieren?

Beispiel: DNS = Domain Name System

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Synchronisierung

A Benennung und Namenssysteme
B Synchronisierung
C Konsistenz und Replication
D Fehlertoleranz

Herausforderung:
Es gibt keine gemeinsame Uhr. Wie *choreografiert* man die verschiedene Komponenten?

Beispiel: NTS = Network Time System

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Konsistenz & Replication

A Benennung und Namenssysteme
B Synchronisierung
C **Konsistenz und Replication**
D Fehlertoleranz

Herausforderung:
Replikation erhöht Redundanz und Performanz, kann aber zu Inkonsistenz führen. Wie sorgen wir für Konsistenz zwischen verschiedene Kopien einer Ressource?

Beispiel: Verteilte Datenbanken

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Fehlertoleranz

A Benennung und Namenssysteme
B Synchronisierung
C Konsistenz und Replication
D Fehlertoleranz

Herausforderung:
Komponenten können – über kurzere oder längere Zeit – ausfallen. Auch Nachrichten können verloren gehen. Wie können wir Systeme bzw. Anwendungen bauen, die damit klar kommen?

Beispiel: Fehlertoleranz durch Middleware

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK