

Verteilte Systeme

Weitverteilte Systeme

Sommersemester 2025

Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Systemsoftware)



Lehrstuhl für Informatik 4
Systemsoftware



**FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG**

TECHNISCHE FAKULTÄT

Weitverteilte Systeme

Motivation

Akamai

- Ziel: Schneller Zugriff auf **Dienste in weitverteilten Systemen**
- Probleme
 - „Mittlere Meile“ des Internets als **Flaschenhals**
 - Teil der Übertragungsstrecke, der weder nutzer- noch anbieternah ist
 - Heterogenes Geflecht von Netzwerken verschiedener Internet-Provider
 - **Nachteile von TCP** in weitverteilten Netzwerken
 - Mehrfacher Nachrichtenaustausch beim Verbindungsaufbau
 - Fensterbasierter Ansatz → Drosslung des Durchsatzes bei hoher Latenz
- Herausforderungen
 - Wie kann das „Mittlere Meile“-Problem abgemildert werden?
 - Wie lassen sich die von Clients beobachtbaren Latenzen minimieren?

■ Literatur



Ankit Singla, Balakrishnan Chandrasekaran, P. Brighten Godfrey, and Bruce Maggs
The Internet at the speed of light
Proceedings of the 13th Workshop on Hot Topics in Networks, S. 1–7, 2014.

Weitverteilte Systeme

Motivation

Akamai

■ Grundsätzliche Architekturansätze

■ **Zentralisierte Datenspeichersysteme**

- Bereitstellung sämtlicher Daten eines Diensts von einem Ort aus
- Eventuell: Replikation der Daten auf (wenige) weitere Orte

■ **Zentralisierte Content-Delivery-Netzwerke**

- Auslagerung zwischenspeicherbarer Inhalte in große Datenzentren
- Auslieferung der restlichen Daten durch die Anwendungsserver

■ **Weitverteilte Content-Delivery-Netzwerke**

- Auslagerung zwischenspeicherbarer Inhalte in viele Cache-Server
- Platzierung von Zwischenspeichern in der Nähe von Clients

■ **Peer-to-Peer-Netzwerke**

- Clients fungieren als Zwischenspeicher
- Auslieferung heruntergeladener Daten an andere Clients

■ Literatur



Tom Leighton

Improving performance on the Internet

Communications of the ACM, 52(2):44–51, 2009.

■ Überblick

■ **Weitverteiltes Content-Delivery-Netzwerk**

- ~350.000 Server in 134 Ländern
- Auslieferung von statischen und dynamischen Daten

■ Anmerkungen zur Wahl des Architekturansatzes

■ Vorteile

- **Gute Skalierbarkeit** durch Verteilung auf viele Server
- Geringe Entfernung zwischen Clients und Zwischenspeichern
- Leistungsfähigkeit ist unabhängig von der aktuellen Client-Anzahl

■ Nachteile

- Signifikanter Zeit- und Kostenaufwand beim Aufbau des Systems
- **Hohe Komplexität** durch Einbindung einer großen Anzahl von Providern

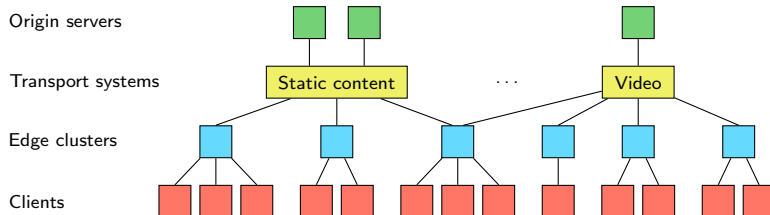
■ Literatur



Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun

The Akamai network: A platform for high-performance Internet applications
SIGOPS Operating Systems Review, 44(3):2–19, 2010.

- Origin-Server
 - **Erzeugung und Bereitstellung der Nutzdaten**
 - Anwendungsserver der Dienstanbieter
- Transportsysteme
 - **Verteilung der Nutzdaten** an Edge-Server
 - Implementierung abhängig von den auszuliefernden Inhalten
- Edge-Cluster
 - **Auslieferung der Nutzdaten** an Clients
 - Platzierung nach Möglichkeit in der Nähe von Clients



Zuordnung von Clients zu Edge-Servern

- Kontinuierliche Erfassung der Konnektivität
 - Einteilung von Servern in **Äquivalenzklassen** auf Basis ihrer IP-Adressen
 - Abschätzung der Verbindungsqualität zwischen Äquivalenzklassen
 - Kombinierung früherer und aktueller Messdaten
 - Metriken (Beispiele): Umlaufzeiten, Verlustraten, Routeninformationen
- Ermittlung des Edge-Servers für einen Client
 - Selektion des Edge-Clusters mittels **Konnektivitätsinformationen**
 - Edge-Server-Auswahl unter Berücksichtigung der angeforderten Daten
- Einsatz des *Domain Name System (DNS)*
 - **Mehrstufige Hierarchie** von DNS-Servern
 - Auflösung von DNS-Namen bei Anfragen von Clients
 1. Top Level Domain Server → Akamai Top Level Name Server
 2. Akamai Top Level Name Server → Akamai Low Level Name Server
 3. Akamai Low Level Name Server → Edge-Server
 - Reduzierte Gültigkeitsdauer von Einträgen auf unteren Ebenen

■ Transportsystem-Cluster

- Geringe Anzahl von **persistenten Verbindungen** zu den Origin-Servern
- Aufbau in mehreren Schichten
- Hoher Verbindungsgrad zu den Edge-Servern

→ Entlastung der Origin-Server

■ Optimierungen

- Suche nach schnelleren Netzwerkpfaden
- **Eigenes Transportprotokoll** zwischen weit entfernten Akamai-Servern
 - Wiederverwendung von Netzwerkverbindungen
 - Anpassung von Timeouts und Fenstergrößen basierend auf Latenzmessungen
- **Applikationsspezifische Ansätze** (Beispiele)
 - Prefetching von in Web-Seiten eingebetten Inhalten (z. B. Bildern)
 - Verlagerung von Anwendungslogik auf die Edge-Server
- Komprimierung von Daten