

# Middleware – Cloud Computing

## Latenzminimierung in Datenzentren

---

Wintersemester 2024/25

Tobias Distler

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



**Lehrstuhl für Informatik 4**  
Systemsoftware



**FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG**

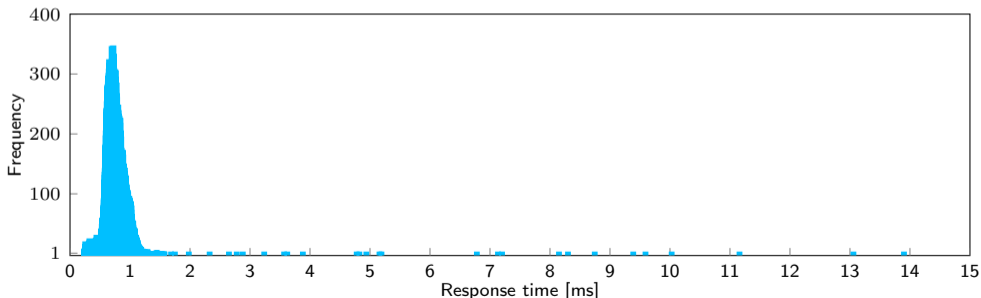
TECHNISCHE FAKULTÄT

## Latenzminimierung in Datenzentren


Motivation

Ansätze und Techniken

- Typische Antwortzeitverteilung in der Praxis
  - Mehrheit der Anfragen wird in relativ kurzer Zeit beantwortet
  - **Ausreißer mit höherer Antwortzeit**
    - Zahlenmäßig geringer Anteil an der Gesamtmenge aller Anfragen
    - Antwortzeiten übersteigen Mittelwert um ein Vielfaches
  - „Tail Latency“
- Beispiel: ZooKeeper-Implementierung aus der Übungsaufgabe



# Gründe für Antwortzeitschwankungen

- Warten beim **Zugriff auf geteilte Ressourcen**
  - Lokal: CPU, Caches, Speicher, Netzwerk etc.
  - Global: Koordinierungsdienste, verteilte Dateisysteme etc.
- Verzögerungen durch **Warteschlangen auf verschiedenen Ebenen**
- Zusatzbelastung durch die Aktivierung periodischer Tasks (Beispiele)
  - Garbage-Collection
  - Datenrekonstruktion in Dateisystemen
  - Komprimierung von Log-Dateien
- Variabilität durch **Einflüsse unterschiedlicher Energiesparmodi**
- ...
- Literatur
  -  Jeffrey Dean and Luiz André Barroso  
**The tail at scale**  
*Communications of the ACM*, 56(2):74–80, 2013.

# Verteilte Bearbeitung von Anfragen

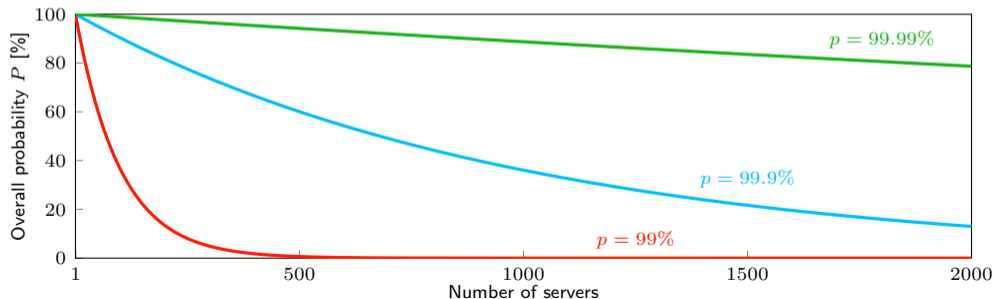
- Bearbeitung von Anfragen durch mehrere Server
  - **Aufspaltung in kleine Teilaufgaben**
  - Eventuell weitere Aufgliederung der Teilaufgaben
  - Bestimmung der **Gesamtantwort auf Basis der Teilergebnisse**
- Beispiel: Teilaufgaben bei Ausführung einer Suchmaschinenanfrage
  - Bereitstellung von statischen Elementen der Web-Seite
  - Ermittlung des Suchergebnisses
  - Zusammenstellung von Vorschlägen für verwandte Suchanfragen
  - Aufbereitung von Kontextinformationen
  - Auswahl der eingeblendeten Werbung
  - ...
- Problem
  - Gesamtantwortzeit einer Anfrage hängt von vielen Faktoren ab
  - **Schwankungen in einzelnen Servern** wirken sich unmittelbar aus

# Auswirkungen langsamer Server

## ■ Vereinfachtes Systemmodell

- An der Bearbeitung einer Anfrage sind **parallel  $s$  Server beteiligt**
- Bei der Erfüllung seiner Teilaufgabe ist jeder Server
  - mit Wahrscheinlichkeit  $p$ : *schnell*
  - mit Wahrscheinlichkeit  $1 - p$ : *langsam*
- Antwort erfolgt, sobald alle Server mit ihren Teilaufgaben fertig sind

## ■ Wahrscheinlichkeit für die schnelle Ausführung einer Anfrage: $P = p^s$



## ■ Beobachtung

„Systems that respond to user actions quickly (**within 100ms**) **feel more fluid and natural to users** than those that take longer.“  
[Dean et al.]

## ■ Wahl der Metrik

- Arithmetisches Mittel: Ungeeignet, da kaum Aussage über Ausreißer
- **$n$ %-Quantil** ( $n^{\text{th}}$ -percentile)
  - Bedeutung: Antwortzeit für  $n$ % aller Anfragen ist  $\leq n$ %-Quantil
  - Typische Werte:  $n = 95$  oder  $n = 99$

## ■ **Kombinierter Ansatz** zur Minimierung von Gesamtantwortzeiten

- Techniken zur Vermeidung von Ausreißern
- Konzipierung von *tail-toleranten* Systemen
- Vergleiche: Vorgehensweise beim Umgang mit Fehlern

## ■ Herausforderungen

- Wie lässt sich die Anzahl bzw. das Ausmaß von Ausreißern reduzieren?
- Wie geht man mit Verzögerungen bei der Bearbeitung von Teilaufgaben um?

## Latenzminimierung in Datenzentren

Motivation

Ansätze und Techniken



# Vermeidung von Ausreißern mittels Lastbalancierung

## ■ Probleme

- Zugriffshäufigkeit variiert zwischen Objekten des Anwendungszustands
- Erhöhte Antwortzeiten als Konsequenz **überlasteter Partitionen**
- Zugriffsmuster können sich ändern → statische Partitionierung unwirksam

## ■ Einsatz von **Mikropartitionen**

- Anzahl der Partitionen  $\gg$  Anzahl der Server
- Vorteile
  - Feingranulare Lastverteilung möglich
  - Effizientes Verschieben von Partitionen zwischen Servern
- Beispiele
  - Verwaltung von Range-Partitions im Microsoft Azure Storage Partition Layer
  - Aufteilung eines MapReduce-Jobs in Tasks

## ■ **Selektive Replikation**

- Replikationsgrad eines Objekts ist abhängig von seiner „Popularität“
- Beispiel: Dateispezifischer Replikationsfaktor in HDFS

## ■ **Priorisierung zeitsensitiver Anfragen**

- Einteilung von Anfragen in Dienstklassen
- Beispiel: BEEMR
  - Interaktive Jobs
  - Batch-Jobs
- Vorteil: Verzögerungen betreffen vor allem nicht-zeitsensitive Operationen

## ■ **Aufspaltung nicht-zeitsensitiver Aufgaben** mit langer Bearbeitungszeit

- Ausführung als Sequenz mehrerer kurzer Operationen
- Bessere Verzahnung mit zeitsensitiven Anfragen möglich
- Vorteil: Steigerung der Effektivität des Priorisierungsansatzes

## ■ **Optimierte Einplanung von Hintergrundaktivitäten**

- Ausführung in Zeiten mit geringer Systemlast
- Abstimmung zwischen unterschiedlichen Servern
- Vorteil: Störender Einfluss ist auf wenige zeitsensitive Anfragen begrenzt

## ■ **Abgesicherte Anfragen** (*Hedged Requests*)

- Aktionen des Clients
  - Senden derselben Anfrage an mehrere Replikate
  - Verwendung der schnellsten Antwort
  - Abbruch der noch ausstehenden Ausführungen
- Effiziente Implementierung
  - Senden der Anfrage an das vermutlich schnellste Replikat
  - Kontaktierung weiterer Replikate erst nach Timeout

## ■ **Verknüpfte Anfragen** (*Tied Requests*)

- Anwendungsfall
  - Größere Schwankungen in den Wartezeiten vor der eigentlichen Ausführung
  - Relativ konstante Bearbeitungszeiten
- Vorgehensweise
  - Client verschickt dieselbe Anfrage an zwei Replikate
  - Anfrage enthält Replikatadressen
  - Bei Ausführungsbeginn: Replikat sendet Abbruchnachricht an das andere

## ■ Prüfung mittels Stichproben (*Canary Requests*)

- Problem
  - Ungewöhnliche Anfrage resultiert in Ausführung ungetesteter Code-Pfade
  - Parallele Verzögerungen oder Fehler auf mehreren Servern
- Lösung
  - Senden von ein oder zwei Teilaufgaben an die entsprechenden Server
  - Verteilung der restlichen Teilaufgaben an die anderen Server erst bei Erfolg

## ■ Einschränkung der Funktionalität (*Graceful Degradation*)

- Ausklammern von Teilaufgaben, deren Bearbeitung zu lange dauert
- Ausliefern der Gesamtantwort, sobald ihre Qualität „gut genug“ ist
- Beispiele
  - Weglassen der Werbung
  - Ausblenden der Kommentarfunktion
- Vorteile
  - Reduzierung des Einflusses einzelner Ausreißer auf die Gesamtantwortzeit
  - Im Allgemeinen bemerken die meisten Nutzer den Unterschied nicht