Fachhochschule Nordwestschweiz
Hochschule für Technik

Application Performance Management

FS 2021

Performance Management Cloud

Michael Faes

Inhalt

Rückblick: Caching

Performance Management Cloud

Autoscaling

Performance-Indikatoren

Scaling-Policies

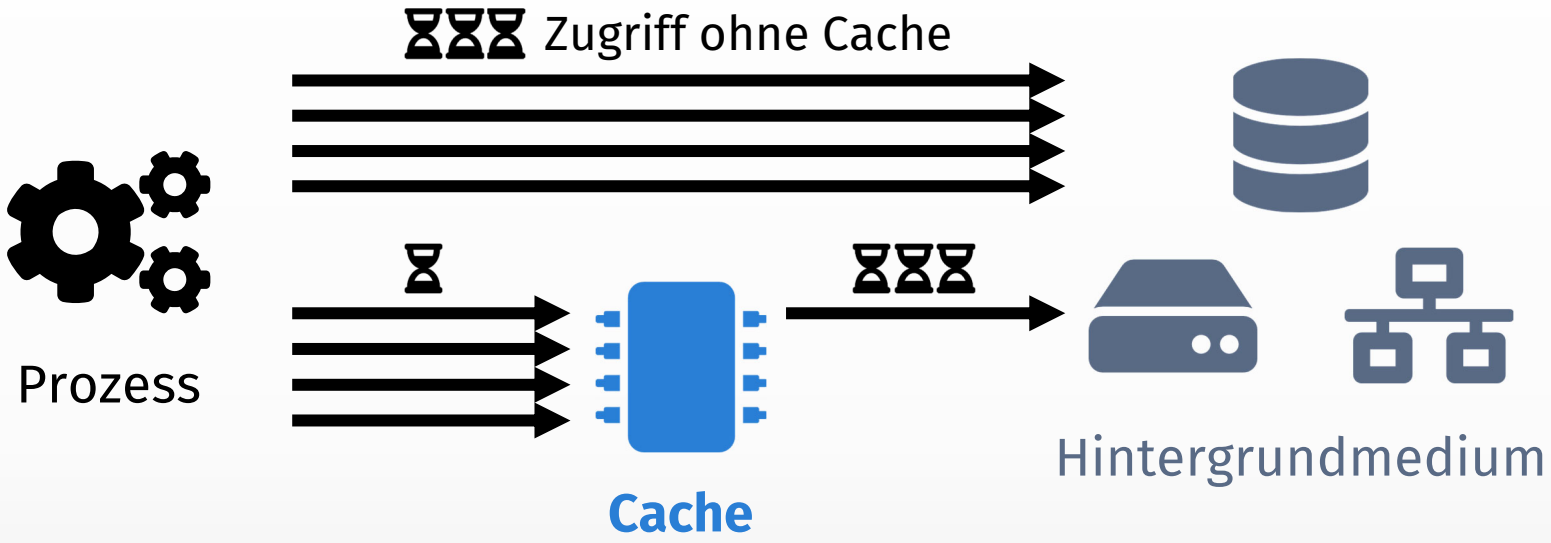
Automatisches Scale-in/out

Vorbesprechung Prüfung

Stoff-Repetition (Frage–Antwort)

Rückblick

Was ist ein Cache?



Beispiele

- CPU Cache (L1, L2, L3, ...)
- Filesystem Cache
- (Paging)
- Web (Browser oder Server)
- DNS
- Memoization
- Dynamic Programming
- Code Cache für JIT (Java)

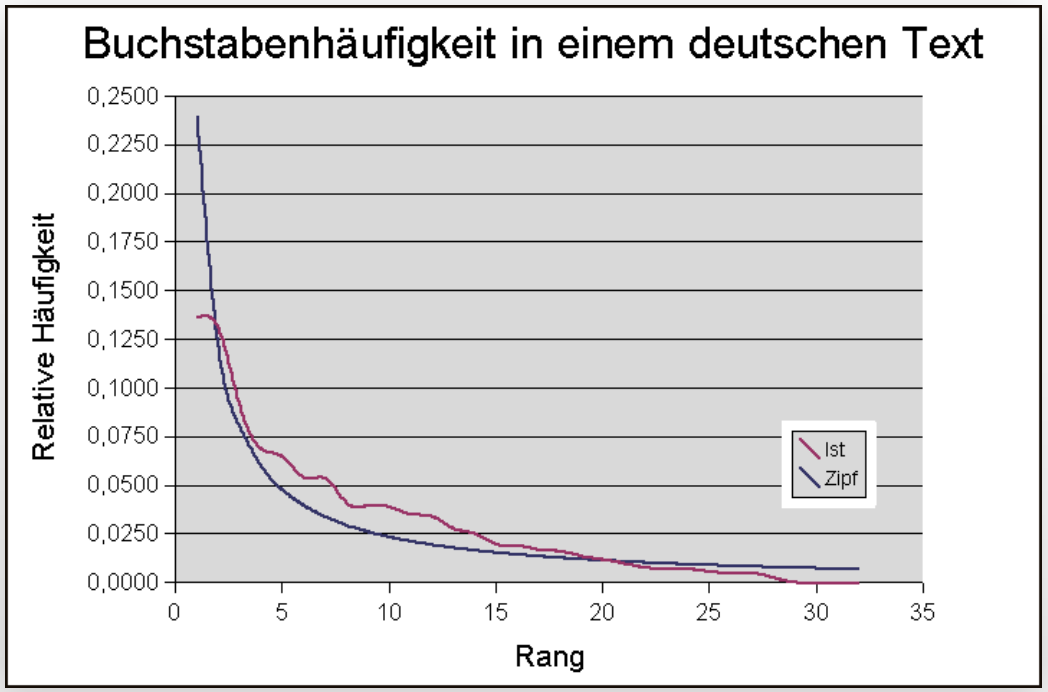
Warum funktionieren Caches?

Principle of Locality

- Locality by time
- Locality by space

Zipf's Law (80:20-Regel):

80% aller Zugriffe gehen auf die 20% selben Objekte



"Anton" (deutsche Wikipedia)

Cache-Algorithmen

(Bélády-Algorithmus)

First-in–first-out (FIFO)

Least recently used (LRU)

Most recently used (MRU)

Least frequently used (LFU)

Random replacement (RR)

Performance Management Cloud

Autoscaling

Scale-out/Scale-in

Monitoring von Performance-Metriken

Berechnen von Performance-Indikatoren (PI/KPI)

PI/KPI-Definitionen

Vergleich mit Policies (Schwellwerte, u.a.)

Policy-Definitionen

Massnahmen: (De-) Provisionierung

Performance-Indikatoren

“Relevante Eckdaten”

Können einfache Metriken sein

- Speicherverbrauch, CPU-Load, Antwortzeit (In-Server), …

Oder higher-level Indikatoren

- Kombinination mehrerer Metriken oder externe Metriken
- Aggregierung über definierten Zeitraum (Ø, Median, …)
- Beispiele: EU-Antwortzeit, aktive Users, Kosten, …

Kriterien für gute PI

Periodische, automatische Erfassung möglich

Kleine Verzögerungszeit

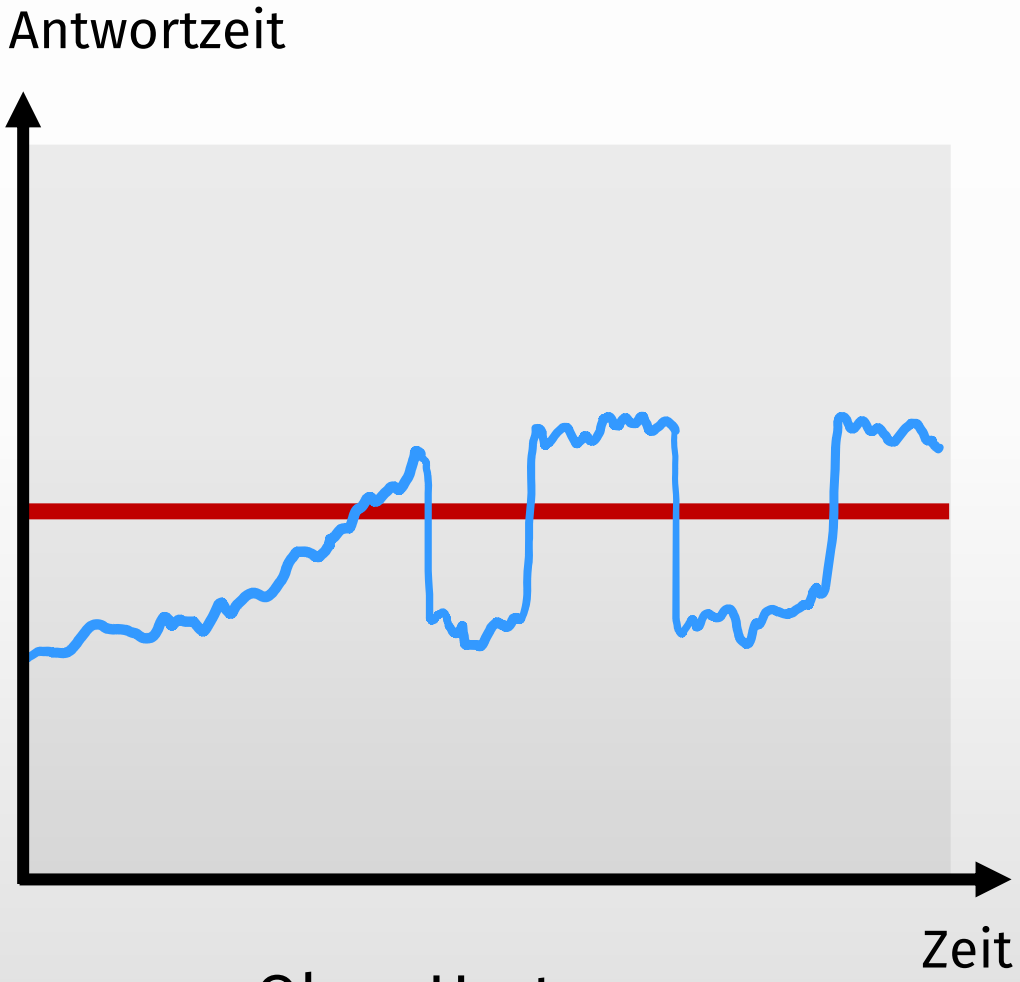
Kleine Volatilität

Trade-off!

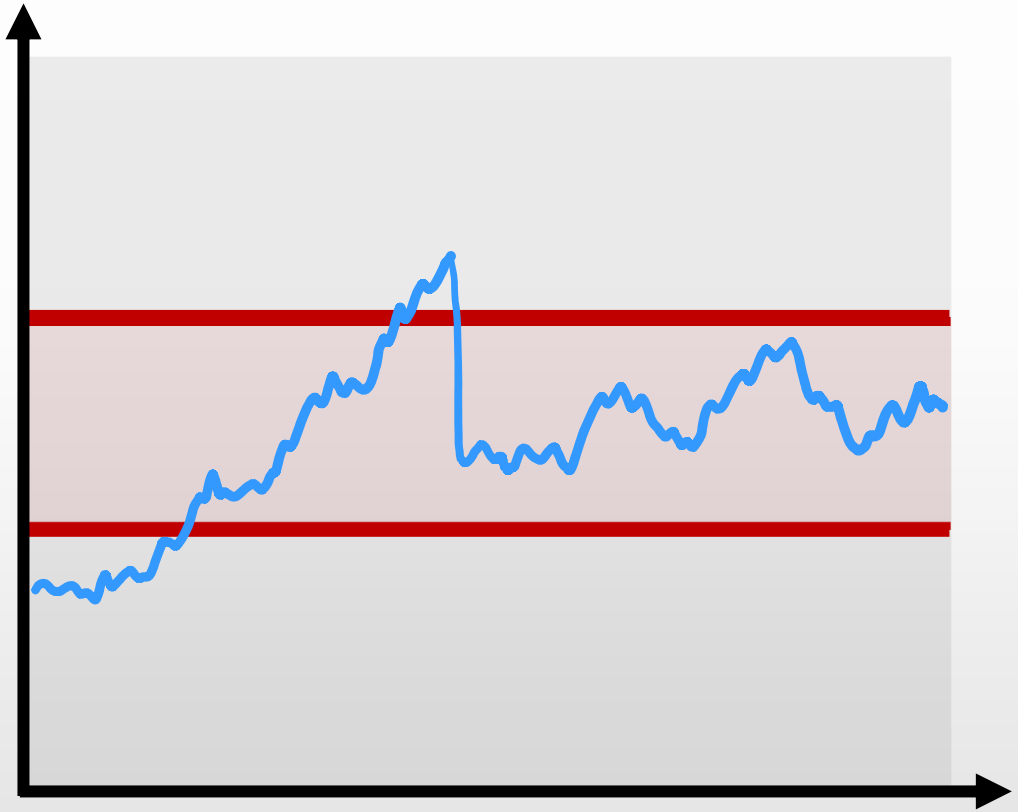
Aussagekraft für Massnahmen

Verständlichkeit (für Policy-Definitionen)

Policies: Schwellwerte



Ohne Hysterese



Mit Hysterese

Beispiel: App Auto-Scaler

```
"scaling_rules": [{
    "metric_type": "memoryutil",
    "breach_duration_secs": 600,
    "threshold": 90,
    "operator": ">=",
    "cool_down_secs": 300,
    "adjustment": "+1"
}, {
    "metric_type": "memoryutil",
    "breach_duration_secs": 600,
    "threshold": 30,
    "operator": "<",
    "cool_down_secs": 300,
    "adjustment": "-1"
}]
```

memoryused,
memoryutil,
cpu,
responsetime,
throughput

github.com/cloudfoundry/app-autoscaler/blob/master/docs/Readme.md

“Finde die Probleme”

```
"scaling_rules": [{
    "metric_type": "cpu",
    "breach_duration_secs": 180,
    "threshold": 100,
    "operator": ">",
    "cool_down_secs": 20,
    "adjustment": "+1"
}, {
    "metric_type": "cpu",
    "breach_duration_secs": 180,
    "threshold": 5,
    "operator": "<=",
    "cool_down_secs": 20,
    "adjustment": "-1"
}]
```

1

```
"scaling_rules": [{
    "metric_type": "memoryutil",
    "breach_duration_secs": 10,
    "threshold": 60,
    "operator": ">=",
    "cool_down_secs": 300,
    "adjustment": "+1"
}, {
    "metric_type": "memoryutil",
    "breach_duration_secs": 60,
    "threshold": 20,
    "operator": "<=",
    "cool_down_secs": 300,
    "adjustment": "-1"
}]
```

2

Automatisches Scale-out

1. Deployment der Basis-VM/des Containers
2. Evtl. Patchen von Software und OS
3. Node-spezifische Konfiguration
4. Software-Clustering (z.B. Replizieren von Daten)
5. Starten aller Services auf der Node
6. Aufnahme der Node in externe Services
7. Testen der Funktionsfähigkeit
8. Aufnahme in Loadbalancer-Liste

Automatisches Scale-in

1. Entfernen der Node aus LB-Liste
2. Entfernen aus externen Services
3. Shutdown der Services auf der Node
4. Entfernen des Software-Clusterings
5. (Deprovisionierung der Ressourcen)

Deprovisionierung

Warum? **Kosten sparen!**

- Energieverbrauch. Wartungsaufwand. Verschleiss?

Ansätze

- Stromsparmodus
- Sleep-Modus
- Herunterfahren
- (Verkaufen?)

Schwierigkeit: Was führt wirklich zur grössten Kostenersparnis?

Übung: Autoscaling mit Kubernetes

& alte Prüfungsfragen