Resilience Patterns Spickzettel: 12 Muster, wann sie passen und wann nicht

Vier Familien · je Karte: Mini-Diagramm, Einsetzen wenn, Vermeiden wenn · Nummern entsprechen den Abschnitten des Artikels · November 2026

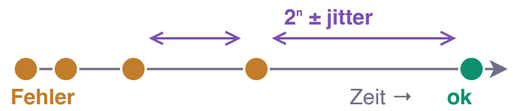
A · Warten begrenzen

1 Timeouts und Deadlines



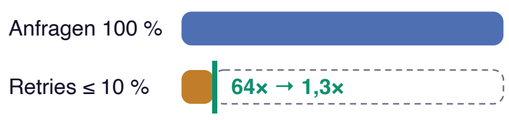
Einsetzen: bei jedem Netzwerkaufruf. Aus dem p99,9 des Aufgerufenen ableiten; Restzeit weiterreichen.
Vermeiden: Bibliotheks-Defaults bleiben, oder ein innerer Timeout länger als der äußere ist.

2 Retries mit Backoff und Jitter



Einsetzen: transiente Fehler bei idempotenten Aufrufen. Exponentielles Backoff mit Obergrenze und Jitter.
Vermeiden: nicht idempotente Schreibzugriffe, 4xx-Fehler, oder jede Schicht selbst wiederholt.

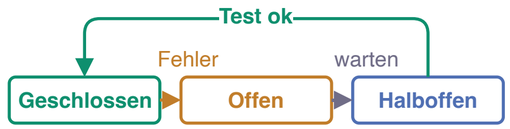
3 Retry-Budgets



Einsetzen: immer, zusätzlich zu Limits je Anfrage. Retries max. ~10 % der Anfragen; nur eine Schicht.
Vermeiden: nur Limits je Anfrage gelten: 3 Retries auf 3 Schichten ergeben 64 Aufrufe ganz unten.

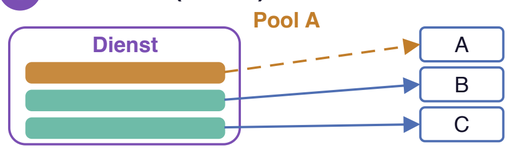
B · Fehler eindämmen

4 Circuit Breaker



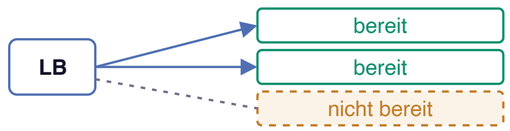
Einsetzen: eine Abhängigkeit fällt aus oder ist langsam. Schnell scheitern, Luft geben, halboffen testen.
Vermeiden: Stichproben winzig sind, ein Breaker viele Hosts abdeckt, oder offen nichts Sinnvolles passiert.

5 Bulkhead (Schott)



Einsetzen: eine langsame Abhängigkeit oder ein Mandant gemeinsame Threads oder Verbindungen leeren könnte.
Vermeiden: Pools so klein werden, dass normale Spitzen warten, oder niemand Abweisungen je Pool misst.

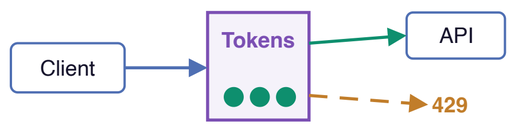
6 Health Checks



Einsetzen: Readiness steuert den Traffic; Liveness erkennt nur hängende Prozesse. Beide billig halten.
Vermeiden: Liveness Abhängigkeiten prüft: ein gemeinsamer Ausfall startet dann die ganze Flotte neu.

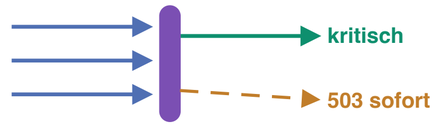
C · Kapazität schützen

7 Rate Limiting



Einsetzen: eine geteilte API vor einzelnen Clients geschützt wird. Kontingent je Client; 429 mit Retry-After.
Vermeiden: es der einzige Überlastschutz ist. Kontingente sind statisch, echte Kapazität nicht.

8 Load Shedding



Einsetzen: der Server über der Kapazität liegt. Früh und billig abweisen, niedrigste Priorität zuerst.
Vermeiden: erst nach der teuren Arbeit abgewiesen wird, oder Health Checks mit abgewiesen werden.

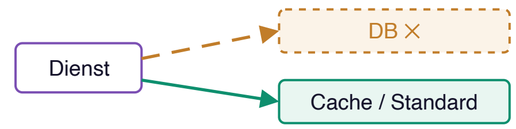
9 Queue-basierter Lastausgleich



Einsetzen: stoßweise asynchrone Arbeit. Begrenzte Queue, Abgelaufenes verwerfen, nach Queue-Alter skalieren.
Vermeiden: ein synchroner Aufrufer hinter einer unbegrenzten Queue wartet: Latenz bleibt verborgen, dann explodiert sie.

D · Weich scheitern, sicher wiederholen

10 Fallback und Degradation



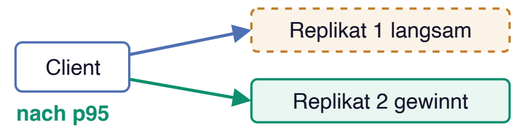
Einsetzen: eine gecachte, Standard- oder reduzierte Antwort besser ist als ein Fehler, z. B. Preise mit Stand.
Vermeiden: der Fallback einen weiteren fragilen Dienst aufruft oder nie im Test ausgelöst wurde.

11 Idempotenzschlüssel



Einsetzen: Schreibzugriffe wiederholt werden. Client sendet eindeutigen Schlüssel; Server speichert und liefert das Ergebnis erneut.
Vermeiden: Schlüssel personenbezogene Daten enthalten oder vor dem Ende der Retries ablaufen.

12 Hedged Requests



Einsetzen: Tail-Latenz bei idempotenten Lesezugriffen. Zweite Kopie nach ~p95 senden, Verlierer abbrechen.
Vermeiden: die Langsamkeit Überlast ist (Hedges erhöhen sie genau dann) oder es ein Schreibzugriff ist.

Legende Anfragefluss · Kontrolle / sicherer Pfad · Fehler oder Überlast · Abgewiesen oder Fallback · 1 Abschnittsnummer im Artikel

conceptualise.de · CC BY 4.0