Engineer

Caching

⏭ das Speichern von Antworten mit Regeln zur Wiederverwendung

Was es ist

Caching speichert das Ergebnis einer Arbeit, etwa eine HTTP-Antwort, ein Abfrageergebnis oder eine gerenderte Seite, und gibt dieses gespeicherte Ergebnis an spätere Anfragen weiter. Regeln legen fest, wie lange ihm vertraut werden darf und wodurch es ausgemustert wird. Das Verb lautet zwischenspeichern, und ein so aufbewahrtes Ergebnis ist zwischengespeichert. Werden innerhalb eines Programms die Rückgabewerte einer einzelnen Funktion gespeichert, heißt das meist Memoisierung.

Der Speicher für die Kopien ist der Cache. Caching ist die Strategie darum herum, und diese beantwortet drei Fragen. Wie lange ist eine Kopie frisch, also ohne Nachfrage zur Auslieferung geeignet? Wie wird sie geprüft, sobald sie veraltet ist? Und was entfernt oder ersetzt die Kopie, wenn sich das Original ändert?

HTTP beantwortet alle drei Fragen in Headern. RFC 9111 nennt eine gespeicherte Antwort frisch, solange ihr Alter unter ihrer Gültigkeitsdauer liegt. Der Ursprungsserver legt diese mit der Direktive `max-age` in `Cache-Control` oder mit einem `Expires`-Datum fest. Danach ist die Antwort veraltet und muss vor der Wiederverwendung normalerweise vom Cache validiert werden. Dieser sendet eine bedingte Anfrage mit einem aufbewahrten Validator, meist dem `ETag` der Antwort in einem `If-None-Match`-Header. Hat sich nichts geändert, antwortet der Ursprungsserver mit 304 Not Modified ohne Rumpf. Die erneute Validierung kostet weiterhin einen Netzwerkumlauf, spart aber die Übertragung und die Arbeit für die erneute Erzeugung des Rumpfs.

```
HTTP/1.1 200 OK
Cache-Control: max-age=3600, stale-while-revalidate=60
ETag: "v42"
```

Innerhalb einer Anwendung nehmen dieselben Fragen andere Formen an. Bei Cache-Aside liest der Code zuerst den Cache. Bei einem Fehlzugriff liest er die Datenbank und füllt den Cache selbst. Bei Write-Through aktualisiert jeder Schreibvorgang den Cache zusammen mit der Datenbank, sodass ein Lesezugriff den aktuellen Wert findet. Die Lebensdauer bis zum Ablauf eines Eintrags, seine Time to Live oder TTL, übernimmt die Rolle von `max-age`. Selbst eine Datenbank kann zwischenspeichern: Eine materialisierte Sicht in PostgreSQL speichert das Ergebnis einer Abfrage als Tabelle. Dieses Ergebnis ändert sich erst, wenn jemand `REFRESH MATERIALIZED VIEW` ausführt.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Caching?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die Abfragen und fehlenden Indizes hinter einem von Jahr zu Jahr schwerfälligeren System finden und die entscheidenden Probleme beheben.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 30 Tage Stromdaten in 180 ms — 180 ms für die Aggregation eines Monats an Zählerständen

30 Tage Stromdaten in 180 ms Kontakt aufnehmen →

Warum es wichtig ist

Jede Caching-Regel tauscht Aktualität gegen Geschwindigkeit. Die passende Abwägung hängt von der Datenart ab. Ein Logo darf ein Jahr alt sein, ein Produktpreis eine Minute. Ein Kontostand oder die Zahl verfügbarer Plätze sollte an der Quelle bestätigt werden, bevor jemand daraufhin handelt. Caching scheitert meist dann, wenn für alle drei dieselbe Regel gilt.

Für statische Dateien ist die Antwort am klarsten. Schreibt ein Build einen Hash des Inhalts jeder Datei in ihren Namen, erhält eine geänderte Datei eine neue URL. Die alte URL kann deshalb ein Jahr lang gecacht werden. RFC 8246 ergänzt die Direktive `immutable`: Sie teilt dem Browser mit, dass die Antwort während ihrer Gültigkeitsdauer gleich bleibt. Selbst beim Neuladen darf deshalb die erneute Validierung entfallen. Die HTML-Seite, die diese Dateien nennt, muss kurzlebig bleiben oder bei jeder Verwendung neu validiert werden. Sonst laden Besucher die Dateien weiterhin unter den alten Namen.

Zwei Direktiven werden leicht verwechselt. `no-cache` erlaubt dem Cache, die Antwort zu speichern, verbietet aber ihre Wiederverwendung ohne vorherige Validierung beim Ursprungsserver. `no-store` verbietet die Speicherung

vollständig. Für eine häufig geänderte Seite liefert `no-cache` mit einem `ETag` aktuelle Inhalte und spart trotzdem die Übertragung, wenn sich nichts geändert hat.

Die schwierige Hälfte ist die Invalidierung. RFC 9111 verlangt von einem Cache, eine URL zu invalidieren, wenn eine Anfrage mit einer nicht sicheren Methode wie `POST`, `PUT` oder `DELETE` an diese URL erfolgreich ist. Derselbe Abschnitt weist darauf hin, dass dies nur die Caches erreicht, die diese Anfrage durchläuft. Browser, CDN und Anwendungscache sind drei getrennte Orte. Eine Kopie an einem davon zu entfernen lässt die beiden anderen bestehen. Die meisten Systeme verlassen sich deshalb auf den Ablaufzeitpunkt als Absicherung und verwenden ausdrückliche Invalidierung, um früher aktuell zu werden.

Auch veraltete Daten können nützlich sein. RFC 5861 definiert `stale-while-revalidate`. Damit darf ein Cache eine veraltete Antwort für eine festgelegte Anzahl Sekunden ausliefern, während er im Hintergrund eine frische abruft. `stale-if-error` erlaubt ihm, eine veraltete Kopie weiter auszuliefern, wenn der Ursprungsserver einen Fehler zurückgibt. Beide machen aus einer harten Ablaufgrenze eine weiche: Der Besucher erhält sofort eine Antwort, die Aktualisierung folgt.

Wie es in der Praxis aussieht

Viele Nutzer einer Plattform sind Seeleute und Außendienstmitarbeiter auf See, mit Telefonen und oft offline. Für sie machte Caching die Web-App installierbar und auf See nutzbar. Der Service Worker wird über next-pwa konfiguriert. Workbox-Laufzeitcaching liefert Schriften, Bilder, Audio, Video und CDN-Assets nach dem Prinzip Cache First aus: vom Gerät, wenn dort eine Kopie liegt, und nur bei einem Fehlzugriff aus dem Netzwerk. Das sind Dateien, die von einer Anfrage zur nächsten gleich bleiben. HTTP-Cache-Control-Header begleiten sie. Weil Service Workers HTTPS benötigen, wurden Installation und Offlineverhalten auf echten Geräten über HTTPS getestet. Das Ergebnis ist eine App, die sich installieren lässt, offline weiterarbeitet und bei erneuten Besuchen aus dem Cache statt aus dem Netzwerk geladen wird. Eine Cache-First-Regel überspringt die Aktualitätsprüfung vollständig, solange ein Eintrag existiert. Deshalb passt sie zu Dateien, deren Inhalt an ihre Adresse gebunden ist.

Am anderen Ende eines Systems liegt eine zehnmal schneller gewordene PostgreSQL-Datenbank, die Nutzer, Mandanten, geografische Daten und tägliche Strommesswerte hält. Caching war eine Maßnahme unter mehreren. Neben

gezielten Indizes, GiST für räumliche Abfragen und zusammengesetzten Indizes für Mandanten- und Datumsfilter sowie zeitbasierter Partitionierung der Tabelle mit den Strommesswerten wurden einige Abfrageergebnisse als materialisierte Sichten gehalten. Außerdem wurden Abfragecaching und Verbindungspooling eingerichtet, um Overhead zu senken. Die Antwortzeiten verbesserten sich um den Faktor 10, kritische Operationen liefen in Millisekunden und der Ressourcenverbrauch sank um 40 %. Diese Verbesserungen gehören zur gesamten Arbeit. Welchen Anteil daran das Caching hatte, wurde nicht gesondert gemessen.

Wo es schiefgeht

Nach einer Bereitstellung erhalten manche Besucher noch die alte Version. Die Antwort enthielt keine ausdrückliche Gültigkeitsdauer, also schätzten die Caches selbst eine. RFC 9111 erlaubt ihnen, einen Bruchteil der Zeit seit `Last-Modified` zu verwenden, und nennt 10 % als typische Einstellung. Eine vor einem Monat zuletzt geänderte Datei kann deshalb etwa drei Tage ohne Nachfrage wiederverwendet werden. Senden Sie mit jeder Antwort ein ausdrückliches `Cache-Control`, schreiben Sie einen Inhaltshash in statische Dateinamen und halten Sie das darauf verweisende HTML kurzlebig.

Ein Purge leert den CDN-Cache, Browser behalten aber die alte Kopie. Ein Purge entfernt die Kopie im CDN und nichts anderes. Jeder Browser, der die Antwort gespeichert hat, behält sie bis zum Ende ihrer eigenen Lebensdauer. Geben Sie öffentlichen Antworten ein kurzes `max-age` für Browser und ein längeres `s-maxage`, das nur gemeinsame Caches anwenden. Dann lebt die löschbare Kopie am längsten. Muss sich eine Browserkopie sofort ändern, ändern Sie die URL.

Eine kurzzeitig fehlende Seite bleibt verschwunden. RFC 9110 erlaubt heuristisches Caching einer 404. Ein Cache darf deshalb eine solche Antwort ohne Angaben zur Gültigkeit für eine selbst geschätzte Dauer behalten. Wer einen Link kurz vor Veröffentlichung einer Seite aufruft, kann weiterhin 404 erhalten, nachdem die Seite existiert. Geben Sie Fehlerantworten eine ausdrückliche kurze Lebensdauer und löschen Sie die URL aus dem Cache, sobald die Seite live geht.

Nach einer Aktualisierung wird ein alter Wert zurückgeschrieben. Bei Cache-Aside findet ein Leser keinen Eintrag und liest die Zeile. Bevor er den Wert speichert, aktualisiert ein Schreiber die Zeile und löscht den Eintrag.

Anschließend speichert der Leser seinen gelesenen Wert, und der alte Wert bleibt, bis etwas ihn entfernt. Löschen Sie den Eintrag nach dem Commit der Transaktion und geben Sie jedem Eintrag eine TTL. Ein durch eine Race Condition oder eine versäumte Invalidierung veralteter Eintrag läuft dadurch von selbst ab. In Redis verwirft ein einfaches `SET` jeden früheren Ablaufzeitpunkt. Setzen Sie die TTL deshalb im selben Befehl:

```
SET price:42 "19.90" EX 300
```

Weiterführende Lektüre

- RFC 9111, Abschnitt 3: Antworten in Caches speichern — wann eine Antwort gespeichert werden darf, gefolgt von den Regeln für Aktualität, erneute Validierung und Invalidierung.
- RFC 9110, Abschnitt 13: Bedingte Anfragen — `ETag`, `If-None-Match` und die 304-Antwort, die erneute Validierung günstig machen.
- RFC 5861: HTTP-Cache-Control-Erweiterungen für veraltete Inhalte — die beiden Direktiven, mit denen ein Cache bewusst veraltete Antworten ausliefern darf.
- RFC 8246: Unveränderliche HTTP-Antworten — die Direktive `immutable` für Dateien, deren URL sich mit dem Inhalt ändert.
- Workbox: Caching-Strategien — Cache First, Network First und Stale While Revalidate in einem Service Worker, mit Code.
- PostgreSQL: Materialisierte Sichten — ein Abfrageergebnis in der Datenbank halten und auf Nachfrage aktualisieren.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Caching?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/caching/