Engineer

Cache

ein Zwischenspeicher für wiederverwendbare Datenkopien

Was es ist

Ein Cache ist ein Speicher, der Datenkopien nahe bei denjenigen hält, die sie lesen. Bei einem wiederholten Lesezugriff liefert er deshalb die Kopie, statt die Daten von ihrem eigentlichen, langsameren Speicherort abzurufen. Der Plural lautet Caches; ein einzelnes gespeichertes Objekt ist ein Cache-Eintrag, und die darin enthaltenen Daten sind eine zwischengespeicherte Kopie.

Ein Cache ist nie die maßgebliche Datenquelle. Die Datei, Zeile oder Serverantwort, die er kopiert, bleibt maßgeblich. Jeder Eintrag darf verworfen und erneut abgerufen werden. Das unterscheidet einen Cache von einem Replikat, das durch Replikation gezielt vollständig und auf demselben Stand gehalten wird, und von einem Puffer, der Daten auf dem Transportweg hält, die womöglich noch nirgendwo sonst existieren. Der Linux-Seitencache ist beides: Er hält gelesene Dateidaten und nimmt Schreibvorgänge zuerst entgegen. Bis die Daten auf der Platte ankommen, markiert er diese Seiten als geändert.

Zwischen Nutzern und Daten liegen mehrere Cache-Schichten. Der Seitencache des Kernels liegt unter jeder Datei, die ein Programm liest. PostgreSQL legt darüber seinen eigenen Seitenpuffer, `shared_buffers`. Ein Browser hält HTTP-Antworten in einem privaten Cache für eine Person; ein CDN oder Reverse Proxy hält einen gemeinsamen Cache für viele. Das Skript einer Seite kann über einen Service Worker Antworten in Cache Storage halten, getrennt vom HTTP-Cache des Browsers. Eine Anwendung kann außerdem selbst berechnete Werte in einem Schlüssel-Wert-Speicher wie Redis halten.

Die Begriffe sind auf jeder Ebene dieselben. Ein Lesezugriff, der seinen Eintrag findet, ist ein Treffer, ein Hit. Findet er ihn nicht, ist das ein Miss, und der Zugriff geht weiter zur ursprünglichen Quelle. Der Anteil der Lesezugriffe mit Treffern ist die Trefferquote. Jeder Eintrag wird anhand seines Schlüssels gefunden. Bei

einem HTTP-Cache besteht dieser mindestens aus Anfragemethode und URL. Ein voller Speicher verwirft Einträge, um Platz zu schaffen. Das heißt Verdrängung oder Eviction, und die Verdrängungsstrategie entscheidet, was gehen muss. Häufig wird der am längsten nicht verwendete Eintrag entfernt, nach dem Verfahren Least Recently Used, kurz LRU.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Cache?“ 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

Ein Cache ist meist die günstigste Beschleunigung, die ein System kaufen kann, günstiger als neue Hardware oder ein neues Schema. Die Rechnung kommt in Form von Speicherbedarf, Kopien an schwer erreichbaren Orten und Kopien, die nicht mehr stimmen. Drei Entscheidungen über den Speicher klären das meiste davon.

Die erste ist sein Standort. Je näher die Kopie beim Leser liegt, desto mehr spart sie: Eine Kopie im Browser erspart sowohl das Netzwerk als auch den Server. Zugleich liegt sie weiter vom Betreiber entfernt. Eine Redis-Instanz lässt sich mit einem Befehl leeren; eine Kopie auf dem Telefon eines Besuchers bleibt dort unter den Regeln, die mit ihr gesendet wurden.

Die zweite ist die Größe, denn Speicher für einen Cache fehlt an anderer Stelle. Die PostgreSQL-Dokumentation nennt für `shared_buffers` einen typischen Standardwert von 128 MB. Für einen dedizierten Server mit mindestens 1 GB Speicher empfiehlt sie 25 % als Ausgangspunkt. Weil PostgreSQL auch vom Cache des Betriebssystems abhängt, bringt ein Anteil von mehr als 40 % laut Dokumentation wahrscheinlich keinen Vorteil gegenüber einem kleineren. Zwei Caches teilen sich eine Maschine.

Die dritte ist, wer den Speicher teilt. RFC 9111 unterscheidet einen privaten Cache für einen Nutzer von einem gemeinsamen Cache, der gespeicherte Antworten für mehrere Nutzer wiederverwendet. Eine persönliche Antwort in einem gemeinsamen Speicher lässt die Seite einer Person bei einer anderen landen. Deshalb darf ein gemeinsamer Cache die Antwort auf eine Anfrage mit einem `Authorization`-Header nicht wiederverwenden, sofern die Antwort dies nicht ausdrücklich erlaubt.

Aus allen drei Entscheidungen folgt eine Regel: Ein Cache muss gefahrlos geleert werden können. Gehen dabei Daten verloren, ist er zu einer Datenbank ohne deren Garantien geworden. Wie lange ein Eintrag noch gut genug zum Ausliefern ist und wodurch er früher entfernt wird, gehört zum Caching, also zum Verfahren. Die Aufgabe des Speichers selbst ist es, Kopien zu halten, sie anhand eines Schlüssels zu finden und bei Platzmangel Speicher freizugeben.

Wie es in der Praxis aussieht

Auf einem Host mit 464 MB, der seine eigene Forge und seinen Webserver betreibt, entscheidet der Seitencache des Kernels darüber, was ein Alarm messen sollte. Ein Totmannschalter, der durch Verstummen Alarm auslöst läuft dort alle fünfzehn Minuten mit einer zufälligen Zeitverschiebung: Ein Skript misst verfügbaren Speicher und freien Plattenplatz und sendet nur dann ein Lebenszeichen an einen externen Dienst, wenn beide über ihren Grenzwerten liegen. Verfügbar ist das richtige Maß, weil freier Speicher auf einem gesunden Linux-System nahe null liegt: Der Kernel verwendet ungenutzten Speicher als Cache. Das Skript las die richtige Spalte, doch sechs Variablennamen darum herum sagten „free“. Wer diesen Namen bei einem Grenzwert von zehn Prozent vertraut, sieht zwölf Megabyte freien Speicher bei 464 MB, hält die Prüfung für defekt und ändert ein einziges Zeichen. Zurück bleibt ein ständig auslösender Alarm, der binnen einer Woche stummgeschaltet wird. Die Benennung ist inzwischen korrigiert.

Eine Ebene darüber liegt ein Anwendungscache vor einer Datenbank. Die Notiz zur Datenbankoptimierung vor dem Sharding schaltet einen solchen Cache vor die Datenbank, wenn der Bestand eines Artikels vollständig reserviert ist. Sind alle Einheiten eines Artikels reserviert, kann die Datenbank sagen, wann die früheste Reservierung ausläuft: zehn Minuten nach ihrer Vergabe. Die Notiz speichert „vollständig reserviert“ mit genau diesem Ablaufzeitpunkt im Cache. Bis dahin weist der Cache Besucher ab, statt diese Arbeit der Datenbank zu überlassen. Wird eine Reservierung früher freigegeben, etwa wegen einer abgelehnten Karte oder eines verlassenen Warenkorbs, wird der Eintrag gelöscht

und die Einheit sofort wieder angeboten. Der Cache darf ablehnen; nur die Datenbank verkauft, über ein bedingtes `UPDATE`. Ein veralteter Eintrag kann für einige Sekunden einen Verkauf kosten, aber niemals eine Einheit zweimal verkaufen.

Die äußerste Schicht ist das eigene Gerät des Besuchers. Dort legt diese Website keinen skriptverwalteten Speicher an. Ihre Seite über Cookies erklärt, dass sie weder IndexedDB noch Cache Storage nutzt und keinen Service Worker installiert. Der Browser kann weiterhin Antworten in seinem normalen HTTP-Cache speichern, der von einer durch die Website verwalteten Offlinekopie getrennt ist.

Wo es schiefgeht

Nach einem Neustart trifft jeder Lesezugriff gleichzeitig den Ursprungsserver. Ein kalter Cache hat für nichts einen Treffer. Eine Bereitstellung, ein Cache-Neustart oder ein Failover schickt deshalb die gesamte Leselast an die Datenbank. Ein System, das nur dank vieler Treffer zurechtkam, kann ausfallen, bevor sich der Cache wieder füllt. Dimensionieren Sie den Ursprungsserver für Cache-Fehlzugriffe, wärmen Sie den Cache auf, bevor er Datenverkehr übernimmt, und führen Sie den Verkehr schrittweise zurück. PostgreSQLs `pg_prewarm` erledigt das für den Puffercache. Es kann aufzeichnen, was die gemeinsamen Puffer enthielten, und es nach einem Neustart wieder einlesen.

Der Cache wächst, bis der Maschine der Speicher ausgeht. Redis setzt auf 64-Bit-Systemen standardmäßig keine Speichergrenze. Eine als Cache verwendete Instanz wächst deshalb mit ihrer Schlüsselmenge, bis das Betriebssystem eingreift. Legen Sie Grenze und Strategie gemeinsam fest. Unter `noeviction` beantwortet eine volle Instanz Befehle zum Speichern neuer Daten mit einem Fehler. Das passt zu einem Speicher, der alles behalten muss, überrascht aber bei einem, der vergessen soll. Die Redis-Dokumentation nennt `allkeys-lru` einen guten Standard, sofern kein Grund für eine andere Strategie spricht:

```
maxmemory 256mb
maxmemory-policy allkeys-lru
```

Ein Besucher erhält eine Seite in der falschen Sprache. Dem Schlüssel fehlt eine Eingabe, von der der Wert abhängt. Eine auf Dänisch gerenderte Seite wird allein unter ihrer URL gespeichert. Der nächste Leser erhält Dänisch, unabhängig von seiner Anfrage. Bilden Sie den Schlüssel aus jeder Eingabe, die die Ausgabe verändert, etwa Sprache, Währung oder Rolle. Bei HTTP-Caches nennt der Antwortheader `Vary` die Anfrageheader, die zum Schlüssel gehören. RFC 9111 erlaubt die Wiederverwendung einer gespeicherten Antwort nur bei übereinstimmenden Headern oder nach erneuter Validierung.

Die installierte App führt weiter alten Code aus. Einträge in Cache Storage haben keinen eigenen Ablaufzeitpunkt und überdauern eine Aktualisierung des Service-Worker-Skripts. Unter demselben Cachenamen geänderte Dateien werden weiter aus den alten Kopien ausgeliefert. Wird der Cache bei jeder Veröffentlichung ohne Aufräumen umbenannt, bleibt jedes Mal ein weiterer auf dem Gerät zurück. Geben Sie jedem Cache einen Versionsnamen, wie die Service-Workers-Spezifikation empfiehlt, und löschen Sie alle älteren, sobald der neue Worker übernimmt.

Weiterführende Lektüre

- RFC 9111: HTTP Caching — der IETF-Standard für private und gemeinsame HTTP-Caches, ihre Schlüssel und das, was ein gemeinsamer Cache unangetastet lassen muss.
- Linux-Kernel: Konzepte der Speicherverwaltung — wie der Seitencache Dateidaten im Speicher hält und warum seine Seiten als freigebbar gelten.
- PostgreSQL: Ressourcenverbrauch — `shared_buffers`, sein üblicher Standardwert und Dimensionierungshinweise, die den Betriebssystemcache einbeziehen.
- Service Workers — die W3C-Spezifikation von Cache Storage und den Unterschieden zum HTTP-Cache des Browsers.
- Redis: Schlüsselverdrängung — `maxmemory` und das Verhalten eines vollen Caches unter den jeweiligen Verdrängungsstrategien.

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 ein Cache?“ 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/cache/