Engineer

Cache

en gemt kopi en browser må genbruge i stedet for at spørge

Hvad det er

En cache er et lager, der holder kopier af data tæt på dem, der læser dem, så en gentaget læsning besvares fra kopien i stedet for fra det langsommere sted, hvor dataene egentlig bor. I flertal hedder det caches; ét gemt element er en cache-post, og dataene i den er en cachelagret kopi.

En cache er aldrig kilden til sandheden. Den fil, række eller det serversvar, den kopierer, forbliver autoritativ, og hver post kan smides væk og hentes igen. Det adskiller en cache fra en replika, som replikering med vilje holder komplet og i takt, og fra en buffer, der holder data på vej, som måske kun findes i bufferen. Linux' page cache er begge dele: den gemmer læste fildata og tager imod skrivninger først som beskidte sider, til de når disken.

Caches stables mellem brugere og data. Kernens page cache ligger under hver fil, et program læser. PostgreSQL holder sin egen pulje af sider, `shared_buffers`, oven på den. En browser gemmer HTTP-svar i en privat cache for én person; et CDN eller en reverse proxy holder en delt cache for mange. En sides script kan gemme svar i Cache Storage gennem en service worker, adskilt fra browserens HTTP-cache. Og en applikation kan gemme værdier, den har beregnet, i et nøgle-værdi-lager som Redis.

Ordforrådet er det samme i hvert lag. En læsning, der finder sin post, er et hit; en, der ikke gør, er en miss og går videre til kilden. Andelen af læsninger, der rammer, er hitraten. Hver post findes ved sin nøgle, som for en HTTP-cache mindst består af forespørgslens metode og URL. Et fuldt lager smider poster ud for at få plads, det hedder eviction, og lagerets eviction-politik vælger, hvad der ryger; least recently used, LRU, er den almindelige.

Brug begrebet i en beslutning

Brug »Hvad er en cache?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. At finde de forespørgsler og manglende indeks bag et system, der er blevet tungere hvert år, og rette dem, der betyder noget.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. 30 dages eldata på 180 ms — 180 ms til at aggregere en måneds måleraflæsninger

30 dages eldata på 180 ms Kontakt os →

Hvorfor det betyder noget

En cache er som regel den billigste fart, et system kan købe, uden ny hardware eller ny datamodel. Regningen kommer som hukommelse, som kopier på steder, der er svære at nå, og som kopier, der holder op med at være sande. Tre beslutninger om selve lageret afgør det meste.

Den første er, hvor det sidder. Jo nærmere kopien er læseren, jo mere sparer den: en kopi i browseren springer både netværket og serveren over. Den er til gengæld længere fra ejeren. En Redis-instans kan tømmes med én kommando; en kopi på en besøgendes telefon bliver liggende under de regler, der blev sendt med den.

Den anden er størrelsen, fordi hukommelse, der gives til en cache, tages fra noget andet. PostgreSQL's dokumentation angiver standardværdien for `shared_buffers` til typisk 128 MB. På en dedikeret server med 1 GB hukommelse eller mere foreslår den at begynde ved 25 %, og fordi PostgreSQL også læner sig op ad styresystemets cache, vil mere end 40 % ifølge dokumentationen næppe virke bedre end en mindre mængde. To caches deler én maskine.

Den tredje er, hvem der deler lageret. RFC 9111 skelner mellem en privat cache for én bruger og en delt cache, der gemmer svar til genbrug for flere. Et personligt svar i et delt lager er den vej, én persons side finder frem til en anden, så en delt cache må ikke genbruge svaret på en forespørgsel med en `Authorization`-header, medmindre svaret udtrykkeligt tillader det.

Én regel følger af alle tre: en cache skal kunne tømmes uden tab. Mister man data ved at tømme den, er den blevet en database uden en databases garantier. Hvor længe en post er god nok at servere, og hvad der fjerner den før tid, hører til caching, praksissen; lagerets egen opgave er at holde kopier, finde dem ved nøgle og afgive plads, når den slipper op.

Sådan ser det ud i praksis

Kernens page cache er det nederste lag. En vært med 464 MB, hvor en forge og en webserver deler hukommelsen med den, har en dead man's switch, der slår alarm ved at tie: hvert femtende minut, med jitter, måler et script tilgængelig hukommelse og ledig disk og pinger kun en ekstern tjeneste, så længe begge ligger over deres tærskler. Tilgængelig er det rigtige mål, for på en sund Linux-maskine ligger den frie hukommelse tæt på nul, når kernen bruger ledig hukommelse som cache. Scriptet læste den rigtige kolonne, men seks variabelnavne omkring det sagde »free«. Den, der stolede på navnene og holdt dem op mod en tærskel på ti procent, ville se tolv megabyte fri på 464 MB, erklære tjekket for defekt og ændre ét tegn i det. Resultatet ville være en alarm, der konstant slår ud, og som nogen har gjort tavs inden for en uge. Navngivningen er siden rettet.

Over kernen kan en applikation have sin egen cache foran databasen. Noten om at optimere databasen, før man sharder den sætter én foran en vare, der er ved at blive udsolgt. Når hver enhed er reserveret, kan databasen sige, hvornår den tidligste reservation udløber, ti minutter efter at den blev lavet, og noten cacher »fuldt reserveret« med præcis det udløbstidspunkt. Indtil da afviser cachen de besøgende, så databasen slipper for dem. En reservation, der frigives før tid, af et afvist kort eller en forladt kurv, sletter posten, så enheden straks er til salg igen. Cachen må sige nej; kun databasen sælger, gennem en betinget `UPDATE`. En forældet post kan i værste fald koste et salg i et par sekunder, men aldrig sælge samme enhed to gange.

Yderst ligger den besøgendes egen enhed, hvor dette site ikke opretter noget scriptstyret lager. Dets side om cookies siger, at det hverken bruger IndexedDB eller Cache Storage og ikke installerer en service worker. Browseren kan stadig gemme svar i sin almindelige HTTP-cache, som er adskilt fra en offlinekopi, sitet selv administrerer.

Hvor det går galt

Efter en genstart tager kilden hver læsning på én gang. En kold cache misser på alt, så en deploy, en genstart af cachen eller en failover sender hele læsebelastningen mod databasen. Et system, der kun klarede sig, fordi de fleste læsninger var hits, kan vælte, før cachen er fyldt op igen. Dimensionér kilden til miss-vejen, varm cachen op, før den tager trafik, og lad trafikken komme tilbage i trin. PostgreSQL's `pg_prewarm` gør det for buffercachen og kan registrere, hvad shared buffers indeholdt, og læse det ind igen efter en genstart.

Cachen vokser, til maskinen løber tør for hukommelse. Redis sætter som standard ingen hukommelsesgrænse på 64-bit-systemer, så en instans, der bruges som cache, vokser med sit nøglerum, til styresystemet griber ind. Sæt en grænse og en politik sammen. Under `noeviction` svarer en fuld instans med en fejl på kommandoer, der ville cache nye data, hvilket passer til et lager, der skal beholde alt, og overrasker et, der skal kunne glemme. Redis' dokumentation kalder `allkeys-lru` et godt standardvalg, når intet taler for en anden politik:

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

En besøgende får en side på det forkerte sprog. Nøglen udelader et input, som værdien afhænger af. En side gengivet på engelsk gemmes alene under sin URL, og den næste læser får engelsk, uanset hvad vedkommende bad om. Byg nøglen af alle input, der ændrer resultatet, for eksempel sprog, valuta eller rolle. For HTTP-caches angiver svarheaderen `Vary` de forespørgselsheadere, der hører med i nøglen, og RFC 9111 lader kun en cache genbruge et gemt svar, når de headere stemmer overens, eller efter at svaret er revalideret.

Den installerede app bliver ved med at køre gammel kode. Poster i Cache Storage har ingen udløbstid af sig selv og overlever en opdatering af service worker-scriptet. Ændrede filer under samme cachenavn serveres fortsat fra de gamle kopier, og en cache, der omdøbes ved hver udgivelse uden oprydning, efterlader én mere på enheden hver gang. Giv hver cache et versionsnummer i navnet, som Service Workers-specifikationen anbefaler, og slet alle ældre, når den nye worker tager over.

Læs mere

- RFC 9111: HTTP Caching — IETF-standarden for private og delte HTTP-caches og deres nøgler, også de svar, en delt cache skal lade være med at genbruge.
- Linux kernel: memory management concepts — hvordan page cachen holder fildata i hukommelsen, og hvorfor dens sider tæller som genvindelige.
- PostgreSQL: Resource Consumption — `shared_buffers`, dens standardværdi og råd om størrelse.
- Service Workers — W3C-specifikationen for Cache Storage, og hvordan den adskiller sig fra browserens HTTP-cache.
- Redis: Key eviction — `maxmemory`, og hvad en fuld cache gør under hver eviction-politik.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en cache?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/cache/