Engineer

Cache

en sparad kopia en webbläsare får återanvända utan att fråga

Vad det är

En cache, på svenska ibland cacheminne eller mellanlager, är ett lager som håller kopior av data nära den som läser dem, så att en upprepad läsning besvaras från kopian i stället för från den långsammare plats där datan egentligen bor. En enskild post i den kallas cachepost, och datan i posten är en cachad kopia.

En cache är aldrig sanningskällan. Filen, raden eller serversvaret som den kopierar förblir det som gäller, och varje post kan kastas och hämtas på nytt. Det skiljer en cache från en replik, som replikeringen med flit håller komplett och i takt, och från en buffert, som håller data under transport som kanske inte finns någon annanstans ännu. Sidcachen i Linux är både och: den behåller fildata som har lästs och tar emot skrivningar först, och markerar de sidorna som smutsiga tills de har nått disken.

Mellan användare och data staplas cachelagren på varandra. Systemkärnans sidcache (page cache) ligger under varje fil ett program läser. PostgreSQL har en egen pool av sidor, `shared_buffers`, ovanpå den. En webbläsare sparar HTTP-svar i en privat cache för en person, medan ett CDN eller en omvänd proxy håller en delad cache för många. Ett skript på en sida kan lägga svar i Cache Storage genom en service worker, skilt från webbläsarens HTTP-cache. Och en applikation kan spara värden som den har räknat fram i ett nyckel-värdelager som Redis.

Orden är desamma i varje lager. En läsning som hittar sin post är en träff; en som inte gör det är en miss och fortsätter till ursprunget. Andelen läsningar som träffar är träffkvoten. Varje post hittas med sin nyckel, som för en HTTP-cache åtminstone består av anropsmetoden och URL:en. Ett fullt lager kastar ut poster

för att få plats, på engelska eviction, och dess utkastningspolicy avgör vilka som får gå. Den vanligaste är LRU, least recently used: den post som använts längst tillbaka åker ut först.

Använd begreppet i ett beslut

Använd ”Vad är en cache?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Hittar databasfrågorna och de saknade indexen bakom ett system som har blivit tyngre för varje år, och åtgärdar dem som spelar roll.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. 30 dagars eldata på 180 ms — 180 ms för att aggregera en månads mätaravläsningar

30 dagars eldata på 180 ms Hör av er →

Varför det spelar roll

En cache är oftast den billigaste fart ett system kan köpa, billigare än ny hårdvara eller ett nytt databasschema. Notan kommer i form av minne, av kopior på ställen som är svåra att nå och av kopior som slutar vara sanna. Tre beslut om lagret avgör det mesta av den.

Det första är var cachen sitter. Ju närmare läsaren kopian ligger, desto mer sparar den: en kopia i webbläsaren slipper både nätverket och servern. Den ligger också längre från ägaren. En Redis-instans kan tömmas med ett enda kommando, men en kopia i en besökares telefon ligger kvar under de regler som skickades med den.

Det andra är storleken, eftersom minne som ges till en cache tas från något annat. PostgreSQL-dokumentationen anger att standardvärdet för `shared_buffers` vanligtvis är 128 MB. På en dedikerad server med 1 GB minne eller mer föreslår den att börja på 25 %, och eftersom PostgreSQL också förlitar sig på operativsystemets cache skriver den att mer än 40 % sannolikt inte fungerar bättre än en mindre mängd. Två cachelager delar på en maskin.

Det tredje är vem som delar lagret. RFC 9111 skiljer mellan en privat cache, för en användare, och en delad cache som lagrar svar för att återanvändas av fler än en. Ett personligt svar i ett delat lager är precis hur en persons sida når en annan, och därför får en delad cache inte återanvända svaret på ett anrop som bar en `Authorization` -header, om inte svaret uttryckligen tillåter det.

Ur alla tre följer en regel: en cache ska gå att tömma utan risk. Om data går förlorade när den töms har den blivit en databas utan en databas garantier. Hur länge en post är tillräckligt färsk för att serveras, och vad som tar bort den i förtid, hör till cachning, själva praktiken. Lagrets eget jobb är att hålla kopior, hitta dem via nyckeln och lämna ifrån sig utrymme när det blir trångt.

Så ser det ut i praktiken

Från systemkärnan via applikationen till besökarens egen enhet ligger cachen här i tre lager.

Det första är systemkärnans sidcache. En värd på 464 MB som kör en forge och en webbserver har ett dödmansgrepp som larmar genom att tystna: var femtonde minut, med slumpmässig förskjutning, mäter ett skript tillgängligt minne och ledigt diskutrymme och pingar en extern tjänst bara så länge båda ligger över sina tröskelvärden. Tillgängligt är rätt mått, eftersom det lediga minnet på en frisk Linux-maskin ligger nära noll när kärnan använder outnyttjat minne som cache. Skriptet läste rätt kolumn, men sex variabelnamn runt omkring sa ”free”. Den som litade på de namnen mot ett tröskelvärde på tio procent skulle se tolv megabyte ledigt på 464 MB, döma ut kontrollen som trasig och göra en ändring på ett tecken. Kvar blir ett larm som löser ut hela tiden och tystas inom en vecka. Namngivningen har sedan dess rättats.

Det andra är en applikationscache framför en databas. Anteckningen om att optimera databasen innan du shardar den sätter en sådan framför en slutförsäljning. När varje enhet av en vara är reserverad kan databasen säga när den tidigaste reservationen löper ut, tio minuter efter att den gjordes, och anteckningen cachar ”fullt reserverad” med exakt den utgångstiden. Fram till dess är det cachen och inte databasen som avvisar besökarna. En reservation som släpps i förtid, genom ett nekat kort eller en övergiven varukorg, tar bort cacheposten, så att enheten genast är till salu igen. Cachen får säga nej; bara databasen säljer, genom en villkorad `UPDATE` . En inaktuell post kan kosta en försäljning i några sekunder men kan aldrig sälja en enhet två gånger.

Det tredje är besökarens egen enhet, där webbplatsen inte skapar något skriptstyrt lager. Sidan om kakor säger att webbplatsen varken använder IndexedDB eller Cache Storage eller installerar en service worker. Webbläsaren kan fortfarande behålla svar i sin vanliga HTTP-cache, som är skild från en offlinekopia som webbplatsen själv hanterar.

Var det går fel

Efter en omstart tar ursprunget varje läsning på en gång. En kall cache missar på allt, så en driftsättning, en omstart av cachen eller en failover skickar hela läslasten till databasen. Ett system som klarade sig bara för att de flesta läsningar var träffar kan falla ihop innan cachen har fyllts igen. Dimensionera ursprunget för missvägen, värm upp cachen innan den tar trafik och släpp tillbaka trafiken stegvis. PostgreSQL har `pg_prewarm` för just buffertcachen, och tillägget kan spara vad de delade buffertarna innehöll och läsa tillbaka det efter en omstart.

Cachen växer tills maskinen får slut på minne. Redis sätter som standard ingen minnesgräns på 64-bitarssystem, så en instans som används som cache växer med sin nyckelrymd tills operativsystemet griper in. Sätt en gräns och en policy samtidigt. Under `noeviction` svarar en full instans med ett fel på kommandon som skulle cacha ny data, vilket passar ett lager som ska behålla allt och överraskar ett som är tänkt att glömma. Redis-dokumentationen kallar `allkeys-lru` ett bra standardval när det inte finns något skäl att föredra ett annat:

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

En besökare får en sida på fel språk. Nyckeln utelämnar något som värdet beror på. En sida som renderats på danska lagras enbart under sin URL, och nästa läsare får danska oavsett vilket språk den läsaren bad om. Bygg nyckeln av varje indatavärde som ändrar resultatet, som språk, valuta eller roll. För HTTP-cachar anger svarshuvudet `Vary` vilka anropshuvuden som hör till nyckeln, och RFC 9111 låter en cache återanvända ett lagrat svar bara när de huvudena stämmer, eller efter att svaret har validerats på nytt.

Den installerade appen fortsätter att köra gammal kod. Poster i Cache Storage har inget eget utgångsdatum och överlever en uppdatering av service worker-skriptet. Filer som ändrats under samma cachenamn fortsätter att serveras från de gamla kopiorna, och en cache som får nytt namn vid varje release utan städning lämnar ytterligare en kvar på enheten varje gång. Ge varje cache ett versionsnummer i namnet, som Service Workers-specifikationen rekommenderar, och radera alla äldre så snart den nya workern har tagit över.

Läs mer

- RFC 9111: HTTP Caching — IETF-standarden för privata och delade HTTP-cachar, deras nycklar och vilka svar en delad cache ska låta bli att återanvända.
- Linux kernel: memory management concepts — hur sidcachen håller fildata i minnet och varför dess sidor räknas som möjliga att återta.
- PostgreSQL: Resource Consumption — `shared_buffers`, dess vanliga standardvärde och råd om dimensionering som räknar med operativsystemets cache.
- Service Workers — W3C:s specifikation av Cache Storage och av hur den skiljer sig från webbläsarens HTTP-cache.
- Redis: Key eviction — `maxmemory` och vad en full cache gör under varje utkastningspolicy.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en cache?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/cache/