Engineer

Nettlesercache

en lagret kopi en nettleser kan gjenbruke uten å spørre

Hva det er

En nettlesercache er den cachen de fleste møter først: kopiene nettleseren tar vare på, så en side den har hentet før, kan vises igjen uten en ny tur over nettet. Prinsippet er det samme på alle lag. En cache er et lager som holder kopier av data nær den som leser dem, så en gjentatt lesing besvares fra kopien i stedet for fra det tregere stedet der dataene egentlig bor. På norsk kalles den også hurtigbuffer, men i et utviklerteam sier man nesten alltid cache. Ett lagret element er en oppføring i cachen, og dataene i den er en cachet kopi.

Cachen er aldri fasiten. Filen, raden eller serversvaret den kopierer, er fortsatt den gjeldende versjonen, og hver oppføring kan kastes og hentes på nytt. Det skiller en cache fra en replika, som replikering med vilje holder komplett og i takt, og fra en buffer, som holder data underveis som kanskje ikke finnes noe annet sted ennå. Sidecachen i Linux er begge deler: den holder fildata som er lest, og tar imot skrivinger først, merket som skitne sider til de har nådd disken.

Cacher stables i lag mellom brukerne og dataene. Sidecachen i systemkjernen ligger under hver fil et program leser. PostgreSQL holder sin egen pool av sider, `shared_buffers`, oppå den. En nettleser holder HTTP-svar i en privat cache for én person, mens et CDN eller en omvendt proxy holder en delt cache for mange. Et skript på en side kan legge svar i Cache Storage gjennom en service worker, adskilt fra nettleserens HTTP-cache. Og en applikasjon kan holde verdier den har regnet ut, i et nøkkel-verdi-lager som Redis.

Ordene er de samme på hvert lag. En lesing som finner oppføringen sin, er et treff (*hit*). En lesing som bommer (*miss*), går videre til kilden. Andelen lesinger som treffer, er treffraten. Hver oppføring finnes med nøkkelen sin, som for en

HTTP-cache minst er forespørselsmetoden og URL-en. Et fullt lager kaster ut oppføringer for å få plass (*eviction*), og utkastingspolitikken velger hva som går. Minst nylig brukt, LRU, er den vanligste.

Bruk begrepet i en beslutning

Bruk «Hva er en nettlesercache?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Finne spørringene og de manglende indeksene bak et system som har blitt tyngre for hvert år, og rette dem som betyr noe.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. 30 dagers strømdata på 180 ms — 180 ms for å aggregere en måned med måleravlesninger

30 dagers strømdata på 180 ms Ta kontakt →

Hvorfor det er viktig

En cache er som regel den billigste farten et system kan kjøpe, billigere enn ny maskinvare eller et nytt databaseskjema. Regningen kommer som minne, som kopier på steder som er vanskelige å nå, og som kopier som ikke lenger stemmer. Tre valg om selve lageret avgjør det meste.

Det første er hvor den står. Jo nærmere kopien er leseren, desto mer sparer den: en kopi i nettleseren hopper over både nettverket og serveren. Den er også lenger unna eieren. En Redis-instans kan tømmes med én kommando, mens en kopi på en besøkendes telefon blir liggende der etter de reglene som ble sendt med den.

Det andre er størrelsen, fordi minne som gis til en cache, tas fra noe annet. Dokumentasjonen til PostgreSQL oppgir at standardverdien for `shared_buffers` typisk er 128 MB. På en dedikert server med 1 GB minne eller mer foreslår den å starte på 25 %, og fordi PostgreSQL også lener seg på operativsystemets cache, sier den at mer enn 40 % neppe virker bedre enn en mindre mengde. To cacher deler én maskin.

Det tredje er hvem som deler lageret. RFC 9111 skiller en privat cache, for én bruker, fra en delt cache som lagrer svar for gjenbruk av flere. Et personlig svar i et delt lager er måten én persons side havner hos en annen på. Derfor skal en delt cache la være å gjenbruke svaret på en forespørsel som hadde en `Authorization` -header, med mindre svaret uttrykkelig tillater det.

Én regel følger av alle tre: det må være trygt å tømme en cache. Mister du data når den tømmes, har den blitt en database uten en databases garantier. Hvor lenge en oppføring er god nok til å serveres, og hva som fjerner den tidlig, hører til caching, selve praksisen. Lagerets egen jobb er å holde kopier, finne dem etter nøkkel og gi fra seg plass når den blir trang.

Slik ser det ut i praksis

Cachen dukker opp på tre ulike lag i arbeidet her, fra systemkjernen til den besøkendes egen enhet.

Det første er sidecachen i systemkjernen, og den har lite å gå på hos selskapets egen vert: 464 MB minne, delt mellom en forge og en webserver. Verten har en dødmannsknapp som slår alarm ved å tie: hvert femtende minutt, med jitter, måler et skript tilgjengelig minne og ledig disk og pinger en ekstern tjeneste bare mens begge ligger over tersklene sine. Tilgjengelig minne er riktig mål, fordi ledig minne ligger nær null på en frisk Linux-maskin når kjernen bruker ubrukt minne til cache. Skriptet leste riktig kolonne, men seks variabelnavn rundt det sa «free». Den som stolte på de navnene opp mot en terskel på ti prosent, ville sett tolv megabyte ledig på 464 MB, konkludert med at sjekken var ødelagt og gjort en endring på ett tegn. Resultatet ville vært en alarm som slår ut hele tiden og blir dempet i løpet av en uke. Navngivingen er rettet siden.

Det andre er en applikasjonscache foran en database. Notatet om å optimalisere en database før du deler den opp setter en foran et utsalg som går tomt. Når hver enhet av en vare er reservert, kan databasen si når den tidligste reservasjonen går ut, ti minutter etter at den ble gjort, og notatet cacher «fullt reservert» med akkurat den utløpstiden. Fram til da blir besøkende avvist av cachen og ikke av databasen. En reservasjon som frigis tidlig, av et avvist kort eller en forlatt handlekurv, sletter oppføringen, så enheten er til salgs igjen med en gang. Cachen kan si nei. Bare databasen selger, gjennom en betinget `UPDATE`. En utdatert oppføring kan koste et salg i noen sekunder og kan aldri selge samme enhet to ganger.

Det tredje er den besøkendes egen enhet, der dette nettstedet ikke oppretter noe skriptstyrt lager. Siden om informasjonskapsler sier at det verken bruker IndexedDB eller Cache Storage eller installerer en service worker. Nettleseren kan fortsatt beholde svar i sin vanlige HTTP-cache, som er adskilt fra en kopi for frakoblet bruk som nettstedet administrerer.

Hvor det går galt

Etter en omstart tar kilden alle lesingene på en gang. En kald cache bommer på alt, så en produksjonssetting, en omstart av cachen eller en failover sender hele leselasten til databasen. Et system som klarte seg bare fordi de fleste lesingene traff, kan falle sammen før cachen er fylt opp igjen. Dimensjoner kilden for bom-veien, varm opp cachen før den tar trafikk, og slipp trafikken inn trinnvis. PostgreSQL-utvidelsen `pg_prewarm` gjør dette for buffercachen, og kan notere hva de delte bufferne holdt, og lese det inn igjen etter en omstart.

Cachen vokser til maskinen går tom for minne. Redis setter som standard ingen minnegrense på 64-bitssystemer, så en instans som brukes som cache, vokser med nøkkelrommet til operativsystemet griper inn. Sett en grense og en utkastingspolitikk sammen. Med `noeviction` svarer en full instans med en feil på kommandoer som ville cachet nye data. Det passer et lager som må beholde alt, og overrasker et som skulle glemme. Dokumentasjonen til Redis kaller `allkeys-lru` et godt standardvalg når du ikke har grunn til å foretrekke noe annet:

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

En besøkende får en side på feil språk. Nøkkelen utelater en inndataverdi som verdien avhenger av. En side gjengitt på dansk lagres under URL-en alene, og neste leser får dansk uansett hva hun ba om. Bygg nøkkelen av hver inndataverdi som endrer resultatet, som språk, valuta eller rolle. For HTTP-cacher navngir svarheaderen `Vary` forespørselsheaderne som hører med i nøkkelen, og RFC 9111 lar en cache gjenbruke et lagret svar bare når de headerne stemmer, eller etter at svaret er revalidert.

Den installerte appen kjører gammel kode videre. Oppføringer i Cache Storage har ingen egen utløpstid og lever videre etter en oppdatering av service worker-skriptet. Filer som er endret under samme cachenavn, serveres fortsatt fra de gamle kopiene, og en cache som får nytt navn ved hver utgivelse uten opprydding, legger igjen én til på enheten hver gang. Gi hver cache et navn med versjon, slik Service Workers-spesifikasjonen anbefaler, og slett alle eldre så snart den nye workeren har tatt over.

Les mer

- RFC 9111: HTTP Caching — IETF-standarden for private og delte HTTP-cacher, nøklene deres og hvilke svar en delt cache må la være å gjenbruke.
- Linux kernel: memory management concepts — hvordan sidecachen holder fildata i minnet, og hvorfor sidene regnes som mulige å ta tilbake.
- PostgreSQL: Resource Consumption — `shared_buffers`, den vanlige standardverdien og råd om størrelse som regner med operativsystemets cache.
- Service Workers — W3C-spesifikasjonen for Cache Storage og for hvordan den skiller seg fra nettleserens HTTP-cache.
- Redis: Key eviction — `maxmemory` og hva en full cache gjør under hver utkastingspolitikk.

Ordliste

Velg ut fra deres faktiske rammer

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en nettlesercache?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/cache/