Engineer

Cache de données

une copie gardée près du lecteur pour accélérer les lectures

Ce que c’est

Un cache est un stockage qui garde des copies de données près de ceux qui les lisent. Une lecture répétée reçoit ainsi la copie plutôt que de solliciter l’endroit plus lent où les données résident réellement. Le pluriel est caches ; un élément stocké est une entrée de cache, et les données qu’il contient sont une copie en cache.

Un cache n’est jamais la source de vérité. Le fichier, la ligne ou la réponse du serveur qu’il copie reste la référence faisant autorité, et chaque entrée peut être jetée puis récupérée à nouveau. C’est ce qui distingue un cache d’une réplique, que la réplication maintient volontairement complète et synchronisée, et d’un tampon, qui conserve des données en transit pouvant ne pas encore exister ailleurs. Le cache de pages de Linux joue les deux rôles : il garde les données de fichiers déjà lues et reçoit d’abord les écritures, en marquant ces pages comme modifiées tant qu’elles n’ont pas atteint le disque.

Les caches s’empilent entre les utilisateurs et les données. Le cache de pages du noyau se trouve sous chaque fichier lu par un programme. PostgreSQL conserve au-dessus son propre ensemble de pages, `shared_buffers`. Un navigateur garde les réponses HTTP dans un cache privé pour une personne ; un CDN ou un mandataire inverse conserve un cache partagé pour plusieurs. Le script d’une page peut conserver des réponses dans Cache Storage par un service worker, séparément du cache HTTP du navigateur. Une application peut aussi conserver les valeurs qu’elle a calculées dans un stockage clé-valeur comme Redis.

Le vocabulaire est le même à chaque couche. Une lecture qui trouve son entrée est un succès de cache, ou hit ; celle qui ne la trouve pas est un défaut de cache, ou miss, et poursuit sa route vers l’origine. La proportion de lectures trouvant leur entrée est le taux de succès. Chaque entrée est retrouvée par sa clé, qui

comprend au minimum la méthode de requête et l'URL pour un cache HTTP. Un stockage plein supprime des entrées pour faire de la place : c'est l'éviction. Sa politique d'éviction choisit ce qui part ; l'élimination de l'élément le moins récemment utilisé, ou LRU, est courante.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un cache de données ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Identifier les requêtes et les index manquants dans un système alourdi au fil des ans, puis corriger ce qui compte.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 30 jours de données électriques traités en 180 ms — 180 ms pour agréger un mois de relevés de compteurs

30 jours de données électriques traités en 180 ms Contactez-nous →

Pourquoi c'est important

Un cache est généralement le gain de vitesse le moins cher qu'un système puisse acheter, moins cher qu'un nouveau matériel ou un nouveau schéma. La facture prend la forme de mémoire, de copies dans des lieux difficiles à atteindre et de copies qui cessent d'être exactes. Trois décisions concernant ce stockage règlent l'essentiel de la question.

La première est son emplacement. Plus la copie est proche du lecteur, plus elle économise : une copie dans le navigateur évite le réseau autant que le serveur. Elle est aussi plus loin de son propriétaire. Une instance Redis peut être vidée en une commande ; une copie sur le téléphone d'un visiteur y reste selon les règles envoyées avec elle.

La deuxième est la taille, car la mémoire donnée à un cache est prise ailleurs. La documentation PostgreSQL indique une valeur par défaut de `shared_buffers` généralement égale à 128 Mo. Sur un serveur dédié disposant de 1 Go de mémoire ou plus, elle suggère de commencer à 25 %. Comme PostgreSQL dépend aussi du cache du système d'exploitation, elle précise qu'une part supérieure à 40 % a peu de chances de fonctionner mieux qu'une quantité plus petite. Deux caches partagent une seule machine.

La troisième est le partage du stockage. La RFC 9111 distingue un cache privé, pour un utilisateur, d'un cache partagé, qui réutilise les réponses stockées pour plusieurs utilisateurs. Une réponse personnelle dans un stockage partagé est le

chemin par lequel la page d'une personne arrive chez une autre. Un cache partagé ne doit donc pas réutiliser une réponse à une requête portant un en-tête `Authorization`, sauf si la réponse l'autorise explicitement.

Une règle découle des trois : vider un cache doit être sans danger. Si le vider fait perdre des données, il est devenu une base de données sans les garanties d'une base de données. La durée pendant laquelle une entrée reste assez bonne pour être servie et ce qui la retire plus tôt relèvent de la mise en cache, la pratique. Le rôle propre du stockage consiste à garder des copies, les retrouver par clé et libérer de la place lorsqu'elle manque.

À quoi cela ressemble en pratique

Sur un hôte de 464 Mo qui exécute sa propre forge et son serveur Web, le cache de pages du noyau détermine ce qu'une alarme doit mesurer. Un dispositif d'homme mort qui donne l'alarme en cessant d'émettre s'y exécute toutes les quinze minutes, avec une légère variation de l'intervalle : un script mesure la mémoire disponible et l'espace disque libre, puis contacte un service externe uniquement tant que les deux dépassent leurs seuils. La mémoire disponible est la bonne mesure, car sur une machine Linux saine, la mémoire libre est proche de zéro, le noyau utilisant la mémoire inutilisée comme cache. Le script lisait la bonne colonne, mais six noms de variables autour d'elle disaient « free ». Une personne faisant confiance à ces noms face à un seuil de dix pour cent verrait douze mégaoctets libres sur 464 Mo, jugerait le contrôle défectueux et changerait un caractère. L'alarme franchirait alors son seuil en permanence et serait mise en sourdine en une semaine. Les noms ont depuis été corrigés.

Une couche au-dessus, un cache applicatif se place devant une base de données. La note sur l'optimisation d'une base avant son partitionnement horizontal en place un pour gérer un stock entièrement réservé. Lorsque chaque unité d'un article est réservée, la base peut indiquer quand la première réservation expire, dix minutes après sa création. La note met alors l'état « fully reserved » (entièrement réservé) en cache avec exactement cette expiration. Jusque-là, le cache refuse les visiteurs à la place de la base de données. Une réservation libérée plus tôt, par une carte refusée ou un panier abandonné, supprime l'entrée : l'unité est immédiatement remise en vente. Le cache peut refuser ; seule la base vend, au moyen d'un `UPDATE` conditionnel. Pendant quelques secondes, une entrée périmée peut faire perdre une vente, mais ne peut jamais vendre deux fois la même unité.

La couche la plus extérieure est l'appareil du visiteur, où ce site ne crée aucun stockage géré par script. Sa page sur les cookies indique qu'il n'utilise ni IndexedDB ni Cache Storage et n'installe aucun service worker. Le navigateur peut néanmoins conserver des réponses dans son cache HTTP ordinaire, distinct d'une copie hors ligne gérée par le site.

Où les problèmes surviennent

Après un redémarrage, l'origine reçoit toutes les lectures en même temps. Un cache froid ne trouve aucune entrée. Un déploiement, un redémarrage du cache ou un basculement envoie donc toute la charge de lecture à la base. Un système qui ne tenait que parce que la plupart des lectures trouvaient leur entrée peut tomber avant le remplissage du cache. Dimensionnez l'origine pour les lectures qui ne trouvent rien dans le cache, préchauffez le cache avant de lui envoyer du trafic et rétablissez le trafic par étapes. Le module `pg_prewarm` de PostgreSQL le fait pour le cache de tampons. Il peut aussi enregistrer le contenu des tampons partagés et le relire après un redémarrage.

Le cache grandit jusqu'à épuiser la mémoire de la machine. Redis ne fixe aucune limite de mémoire par défaut sur les systèmes 64 bits. Une instance utilisée comme cache grandit donc avec l'ensemble de ses clés jusqu'à l'intervention du système d'exploitation. Fixez ensemble une limite et une politique. Avec `noeviction`, une instance pleine répond par une erreur aux commandes qui mettraient de nouvelles données en cache. Cela convient à un stockage qui doit tout conserver et surprend dans un stockage censé oublier. La documentation Redis présente `allkeys-lru` comme une bonne valeur par défaut en l'absence de raison d'en préférer une autre :

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

Un visiteur reçoit une page dans la mauvaise langue. La clé omet une donnée d'entrée dont dépend la valeur. Une page rendue en danois est stockée sous sa seule URL, et le lecteur suivant reçoit du danois quelle que soit sa demande. Construisez la clé à partir de chaque donnée d'entrée qui change la sortie, comme la langue, la devise ou le rôle. Pour les caches HTTP, l'en-tête de réponse `Vary` nomme les en-têtes de requête qui appartiennent à la clé. La RFC 9111 n'autorise la réutilisation d'une réponse stockée que si ces en-têtes correspondent ou après une nouvelle validation de la réponse.

L'application installée continue à exécuter l'ancien code. Les entrées de Cache Storage n'ont pas d'expiration propre et survivent à une mise à jour du script du service worker. Les fichiers modifiés sous un même nom de cache continuent à être servis depuis les anciennes copies. Un cache renommé à chaque publication sans nettoyage en laisse un de plus sur l'appareil à chaque fois. Donnez un nom versionné à chaque cache, comme le recommande la spécification Service Workers, et supprimez tous les anciens dès que le nouveau worker prend le relais.

Pour aller plus loin

- RFC 9111 : HTTP Caching — la norme IETF des caches HTTP privés et partagés, leurs clés et ce qu'un cache partagé doit laisser intact.
- Noyau Linux : concepts de gestion de la mémoire — la manière dont le cache de pages conserve les données de fichiers en mémoire et pourquoi ses pages sont considérées comme récupérables.
- PostgreSQL : consommation des ressources — `shared_buffers`, sa valeur par défaut habituelle et des conseils de dimensionnement tenant compte du cache du système d'exploitation.
- Service Workers — la spécification W3C de Cache Storage et de ses différences avec le cache HTTP du navigateur.
- Redis : éviction des clés — `maxmemory` et le comportement d'un cache plein sous chaque politique d'éviction.

Glossaire

Choisissez selon vos contraintes réelles

Présentez un jeu de données représentatif, les questions auxquelles il doit répondre et le goulot d'étranglement actuel. Nous pouvons convenir de la manière de mesurer la qualité, le débit et la reprise.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un cache de données ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/cache/