Engineer

Mise en cache

⏭ les règles pour conserver et réutiliser une réponse

Ce que c’est

La mise en cache consiste à enregistrer le résultat d’un travail, comme une réponse HTTP, un résultat de requête ou une page rendue, puis à servir ce résultat enregistré aux requêtes suivantes selon des règles précisant combien de temps il est fiable et ce qui provoque son retrait. Le verbe est mettre en cache, et un résultat conservé de cette manière est en cache. La mise en cache des valeurs renvoyées par une fonction à l’intérieur d’un programme est généralement appelée mémoïsation.

L’espace de stockage qui contient les copies est le cache. La mise en cache est la politique qui l’entoure et répond à trois questions. Pendant combien de temps une copie est-elle fraîche, c’est-à-dire utilisable sans rien demander à personne ? Une fois périmée, comment la vérifie-t-on ? Et lorsque l’original change, qu’est-ce qui supprime ou remplace la copie ?

HTTP répond aux trois questions par des en-têtes. La RFC 9111 appelle une réponse stockée fraîche tant que son âge est inférieur à sa durée de fraîcheur. L’origine fixe celle-ci avec la directive `max-age` de `Cache-Control` ou avec une date `Expires` . Au-delà, la réponse est périmée, et un cache doit normalement la valider avant de la réutiliser. Il envoie une requête conditionnelle portant un validateur conservé, généralement l’ `ETag` de la réponse dans un en-tête `If-None-Match` . Si rien n’a changé, l’origine répond 304 Not Modified, sans corps. La revalidation coûte encore un aller-retour, mais épargne le transfert et le travail de reconstruction du corps.

```
HTTP/1.1 200 OK
Cache-Control: max-age=3600, stale-while-revalidate=60
ETag: "v42"
```

Dans une application, les mêmes questions prennent d'autres formes. Avec le modèle cache-aside, le code lit d'abord le cache puis, s'il n'y trouve rien, lit la base de données et remplit lui-même le cache. Avec le modèle write-through, chaque écriture actualise le cache en même temps que la base : une lecture trouve donc la valeur courante. Une expiration sur chaque entrée, sa durée de vie ou TTL, joue le rôle de `max-age`. Même une base de données peut mettre en cache : une vue matérialisée PostgreSQL stocke le résultat d'une requête comme une table, et ce résultat ne change que lorsque quelqu'un exécute `REFRESH MATERIALIZED VIEW`.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que la mise en cache ? » 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

Chaque règle de mise en cache échange de la fraîcheur contre de la vitesse, et le bon compromis dépend du type de données. Un logo peut avoir un an. Le prix d'un produit peut avoir une minute. Un solde ou un nombre de places doit être confirmé auprès de la source avant que quelqu'un agisse sur cette base. La mise en cache échoue le plus souvent lorsqu'une règle unique est appliquée aux trois.

Pour les fichiers statiques, la réponse est la plus simple. Lorsqu'une compilation inscrit dans le nom de chaque fichier un hachage de son contenu, un fichier modifié reçoit une nouvelle URL. L'ancienne peut donc être mise en cache pendant un an. La RFC 8246 ajoute la directive `immutable`, qui indique au navigateur que la réponse restera identique tant qu'elle est fraîche. Même un rechargement peut ainsi éviter de la revalider. La page HTML qui nomme ces fichiers doit garder une durée de vie courte ou être revalidée à chaque utilisation, sinon les visiteurs continuent à charger les fichiers sous les anciens noms.

Deux directives se confondent facilement. `no-cache` permet de stocker la réponse, mais interdit de la réutiliser sans validation préalable auprès de l'origine ; `no-store` interdit tout stockage. Pour une page qui change souvent,

`no-cache` avec un `ETag` donne un contenu à jour tout en économisant le transfert si rien n'a changé.

L'invalidation est la partie difficile. La RFC 9111 exige qu'un cache invalide une URL lorsqu'une requête non sûre vers elle, comme `POST`, `PUT` ou `DELETE`, réussit. La même section précise que seuls les caches traversés par cette requête sont concernés. Un navigateur, un CDN et un cache applicatif sont trois emplacements distincts. Retirer une copie de l'un laisse les deux autres en place. La plupart des systèmes comptent donc sur l'expiration comme filet de sécurité et utilisent l'invalidation explicite pour être à jour plus tôt.

Les données périmées peuvent aussi être utiles. La RFC 5861 définit `stale-while-revalidate`, qui permet de servir une réponse périmée pendant un nombre indiqué de secondes tout en récupérant une réponse fraîche en arrière-plan, et `stale-if-error`, qui permet de continuer à servir une copie périmée si l'origine renvoie une erreur. Les deux transforment une expiration stricte en expiration souple : le visiteur reçoit immédiatement une réponse, et la mise à jour suit.

À quoi cela ressemble en pratique

De nombreux utilisateurs d'une plateforme sont des marins et du personnel de terrain en mer, sur téléphone et souvent hors ligne. Pour eux, la mise en cache a rendu l'application Web installable et utilisable en mer. Le service worker est configuré par next-pwa. Le cache d'exécution de Workbox sert polices, images, audio, vidéo et ressources CDN en donnant la priorité au cache : depuis l'appareil lorsqu'une copie existe, depuis le réseau seulement sinon. Ce sont des fichiers qui restent identiques d'une requête à l'autre, accompagnés d'en-têtes HTTP de contrôle du cache. Les service workers nécessitant HTTPS, l'installation et le fonctionnement hors ligne ont été testés en HTTPS sur de véritables appareils. Le résultat est une application qui s'installe, continue à fonctionner hors ligne et se recharge depuis le cache plutôt que depuis le réseau lors des visites suivantes. Une règle donnant la priorité au cache ne vérifie pas du tout la fraîcheur tant qu'une entrée existe. C'est pourquoi elle convient aux fichiers dont le contenu est lié à l'adresse.

À l'autre bout d'un système se trouve une base PostgreSQL rendue dix fois plus rapide, contenant utilisateurs, organisations clientes, données géographiques et relevés électriques quotidiens. La mise en cache était une mesure parmi plusieurs. En parallèle d'index ciblés — GiST pour les requêtes spatiales et composites pour les filtres sur l'organisation cliente et la date — et du

partitionnement temporel de la table de relevés électriques, certains résultats de requête ont été conservés comme vues matérialisées. La mise en cache des requêtes et un pool de connexions ont aussi été mis en place pour réduire la surcharge. Les temps de réponse se sont améliorés d'un facteur 10, les opérations critiques s'exécutaient en millisecondes et l'utilisation des ressources a baissé de 40 %. Ces gains résultent de l'ensemble du travail : la contribution de la mise en cache n'est donc pas mesurée séparément.

Où les problèmes surviennent

Après un déploiement, certains visiteurs reçoivent encore l'ancienne version. La réponse ne précisait pas de durée de fraîcheur, et les caches en ont donc estimé une. La RFC 9111 les autorise à utiliser une fraction du temps écoulé depuis `Last-Modified` et cite 10 % comme valeur habituelle. Un fichier modifié pour la dernière fois un mois auparavant peut donc être réutilisé environ trois jours sans vérification. Envoyez un `Cache-Control` explicite pour chaque réponse, placez un hachage du contenu dans les noms des fichiers statiques et donnez une durée de vie courte au HTML qui les désigne.

Une purge vide le CDN, et les navigateurs gardent l'ancienne copie. Une purge retire la copie du CDN et rien d'autre. Chaque navigateur ayant stocké la réponse la conserve jusqu'à l'expiration de sa propre durée de vie. Donnez aux réponses publiques un `max-age` court pour les navigateurs et un `s-maxage` plus long, appliqué seulement par les caches partagés. La copie qui peut être purgée est ainsi celle qui vit le plus longtemps. Lorsqu'une copie du navigateur doit changer immédiatement, changez l'URL.

Une page brièvement absente le reste. La RFC 9110 rend une réponse 404 susceptible d'être mise en cache de façon heuristique. Un cache peut donc conserver une telle réponse sans information de fraîcheur pendant une durée qu'il estime lui-même. Un lien suivi juste avant la publication d'une page peut continuer à renvoyer 404 après sa création. Donnez aux réponses d'erreur une durée de vie courte explicite et purgez l'URL à la mise en ligne de la page.

Une ancienne valeur est réinscrite après une mise à jour. Avec cache-aside, un lecteur ne trouve rien dans le cache et lit la ligne. Avant qu'il stocke la valeur, un processus d'écriture actualise la ligne et supprime l'entrée. Le lecteur stocke ensuite ce qu'il avait lu, et l'ancienne valeur reste jusqu'à sa suppression. Supprimez l'entrée après la validation de la transaction et donnez un TTL à chaque entrée, pour qu'une valeur périmée finisse par expirer même si une

invalidation a été manquée ou perturbée par une exécution concurrente. Dans Redis, un simple `SET` supprime toute expiration antérieure. Fixez donc le TTL dans la même commande :

```
SET price:42 "19.90" EX 300
```

Pour aller plus loin

- RFC 9111, section 3 : stockage des réponses dans les caches — les conditions de stockage d'une réponse, puis les règles de fraîcheur, de revalidation et d'invalidation.
- RFC 9110, section 13 : requêtes conditionnelles — `ETag`, `If-None-Match` et la réponse 304 qui rendent la revalidation peu coûteuse.
- RFC 5861 : extensions HTTP de Cache-Control pour le contenu périmé — les deux directives permettant de servir volontairement des réponses périmées.
- RFC 8246 : réponses HTTP immuables — la directive `immutable` pour les fichiers dont l'URL change avec le contenu.
- Workbox : stratégies de mise en cache — priorité au cache, priorité au réseau et stale-while-revalidate dans un service worker, avec du code.
- PostgreSQL : vues matérialisées — conserver un résultat de requête dans la base et l'actualiser à la demande.

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 que la mise en cache ? » 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/caching/