Engineer

キャッシュ処理

⏭ 結果を保存し、再利用の期限や検証・削除のルールを決めること

どのようなものか

キャッシュ処理は、HTTP応答、クエリの結果、描画したページなど、何らかの処理結果を保存し、いつまで信頼でき、何をきっかけに使わなくするかというルールに従って、後の要求へその結果を渡す方法です。英語の動詞はto cacheで、こうして保持した結果をcachedと呼びます。プログラム内で一つの関数の戻り値をキャッシュすることは、通常、メモ化と呼びます。

コピーを保持する保存領域が、キャッシュです。キャッシュ処理は、その周囲のポリシーであり、三つの問いを扱います。コピーはいつまで新鮮、つまり他に確認せず応答へ使える状態でしょうか。古くなった後は、どう確認するのでしょうか。そして、元の内容が変わったら、何がコピーを削除・置換するのでしょうか。

HTTPでは、三つすべてをヘッダーで表します。RFC 9111は、保存済み応答の経過時間が鮮度の有効期間より短い間を、新鮮と呼びます。その期間は、オリジンが `Cache-Control` の `max-age` ディレクティブか、 `Expires` の日付で定めます。過ぎると応答は古くなり、通常、キャッシュは再利用前に検証する必要があります。保存していた検証子を付けて条件付き要求を送り、一般には、応答の `ETag` を `If-None-Match` ヘッダーに入れます。何も変わっていなければ、オリジンは本文のない304 Not Modifiedを返します。再検証にも一往復は必要ですが、転送と、本文を組み立て直す処理を省けます。

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

アプリケーション内では、同じ問いが別の形になります。キャッシュアサイドでは、コードが最初にキャッシュを読み、ミスしたらデータベースを読んで、自らキャッシュを埋めます。ライトスルーでは、書き込みのたびに、データベースとともにキャッシュも更新し、読み取りで現在の値を得られるようにします。各エントリーの有効期間、time to live（TTL）が、`max-age` の役割を担います。データベース自体もキャッシュできます。PostgreSQLのマテリアライズドビューは、クエリの結果をテーブルとして保持し、誰かが `REFRESH MATERIALIZED VIEW` を実行したときだけ、その結果が変わります。

概念を判断に生かす

「キャッシュ処理とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 年々重くなってきたシステムのクエリと不足するインデックスを調べ、効果のある箇所を改善します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 30日分の電力データを180 msで処理 — 180 ms 1か月分のメーター値を集計

30日分の電力データを180 msで処理 お問い合わせ →

なぜ重要なのか

どのキャッシュルールも、鮮度と速さを引き換えにしており、適切な釣り合いはデータの種類で異なります。ロゴは一年前のものでも使えます。商品の価格は一分前のものでも使えます。残高や残席数は、人が行動する前に、元のデータで確認すべきです。三つに同じルールを使うことが、キャッシュ処理で最もよくある失敗です。

静的ファイルには、最も明快な方法があります。ビルドで各ファイルの内容のハッシュを名前に入れると、変更後のファイルは新しいURLになるため、古いURLは一年間キャッシュできます。RFC 8246の `immutable` ディレクティブは、新鮮な間は応答が変わらないとブラウザーへ伝えるため、再読み込みでも再検証を省けます。そのファイル名を示すHTMLページは、有効期間を短くするか、利用のたびに再検証する必要があります。そうしなければ、訪問者は古い名前を読み込み続けます。

混同しやすい二つのディレクティブがあります。 `no-cache` は応答の保存を認めますが、オリジンに先に確認せず再利用することを禁じます。 `no-store` は、保存自体を禁じます。頻繁に変わるページでは、 `no-cache` と `ETag` を使えば、最新の内容を届けながら、変わっていない場合の転送を省けます。

難しいのは、無効化です。RFC 9111は、`POST`、`PUT`、`DELETE`など、安全でないメソッドの要求が成功した場合、そのURLのキャッシュを無効にするよう求めています。同じ節は、それが届くのは、その要求が通ったキャッシュだけだとも指摘します。ブラウザー、CDN、アプリケーションキャッシュは、別々の三か所であり、一つからコピーを取り除いても、残り二つには残ります。そのため、多くのシステムは、有効期限を最後の安全策にし、明示的な無効化を、より早く最新の状態にする手段として使います。

古いデータが役立つ場合もあります。RFC 5861は、指定した秒数の間、バックグラウンドで新しい応答を取得しながら古い応答を返せる`stale-while-revalidate`と、オリジンがエラーを返した場合に古いコピーを提供し続けられる`stale-if-error`を定めています。どちらも、厳密な期限を柔軟なものに変えます。訪問者にはすぐに答えを返し、更新は後から行います。

実際の活用例

あるプラットフォームでは、多くの利用者が船員や海上の現場担当者で、スマートフォンを使い、オフラインになることも少なくありません。その人たちに向けて、キャッシュによってウェブアプリがインストール可能になり、海上でも使えるようになりました。サービスワーカーをnext-pwaで設定し、Workboxの実行時キャッシュで、フォント、画像、音声、動画、CDNの素材をキャッシュ優先で提供します。コピーがあれば機器から返し、ミスしたときだけネットワークを使います。これらは要求ごとに変わらないファイルで、HTTPのcache-controlヘッダーも付けます。サービスワーカーにはHTTPSが必要なため、インストールとオフライン動作は、HTTPSを使い、実機でテストされました。結果として、インストールでき、オフラインでも動き続け、再訪時にはネットワークではなくキャッシュから開くアプリになりました。キャッシュ優先のルールは、エントリーがある間、鮮度の確認を完全に省くため、内容とアドレスが結び付いたファイルに適しています。

システムの反対側には、利用者、テナント、地理データ、日々の電力測定値を保持する、十倍高速になったPostgreSQLデータベースがあります。キャッシュは、複数の対策の一つでした。対象を絞ったインデックスとして、空間クエリ向けのGiST、テナント・日付の絞り込み向けの複合インデックスを使い、電力テーブルを時間軸でパーティション分割しました。それとともに、一部のクエリ結果をマテリアライズドビューとして保持し、処理の負担を減らすためにクエリキャッシュとコネクションプールを設定しました。応答速度は10倍になり、重要な処理はミリ秒単位で動き、リソース使用量は40%減りました。この成果は作業全体によるものであり、キャッシュだけの寄与は個別には測定されていません。

問題が起きるところ

デプロイ後も、一部の訪問者に旧版が届きます。 応答に鮮度の明示がなく、キャッシュが推定したためです。RFC 9111は、 `Last-Modified` からの経過時間の一部を使うことを認め、一般的な設定として10%を挙げます。そのため、一か月前に変更したファイルは、問い合わせずに約三日間再利用される場合があります。すべての応答に明示的な `Cache-Control` を付け、静的ファイル名には内容のハッシュを入れ、それを指すHTMLの有効期間は短くします。

CDNをパージしても、ブラウザーに古いコピーが残ります。 パージが取り除くのはCDNのコピーだけです。応答を保存した各ブラウザーは、自分の有効期間が切れるまで保持します。公開応答には、ブラウザー向けに短い `max-age` 、共有キャッシュだけが適用する、より長い `s-maxage` を設定し、削除できるコピーのほうを長く残します。ブラウザーのコピーを直ちに変える必要がある場合は、URLを変えます。

一時的になかったページが、ないままになります。 RFC 9110は、404を推定によってキャッシュできる応答としています。そのため、鮮度の情報がなければ、キャッシュが自分で決めた期間、保持する場合があります。ページの公開直前にたどったリンクが、公開後も404を返し続けることがあります。エラー応答には、短い有効期間を明示し、ページを公開したら、そのURLをパージします。

更新した後で、古い値を書き戻します。 キャッシュアサイドで、読み取り側がミスして行を読みます。その値を保存する前に、書き込み側が行を更新し、エントリーを削除します。その後、読み取り側が先ほど読んだ値を保存するため、古い値が、何かに取り除かれるまで残ります。トランザクションのコミット後にエントリーを削除し、すべてのエントリーにTTLを設け、競合や見落としで無効化がうまくいかなくても、自然に期限が切れるようにします。Redisでは、単純な `SET` は以前の有効期限を捨てるため、同じコマンドでTTLも設定します。

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

参考資料

- RFC 9111、第3節：キャッシュへの応答の保存：応答を保存できる条件と、それに続く鮮度、再検証、無効化のルールを説明します。
- RFC 9110、第13節：条件付き要求：再検証の負担を小さくする `ETag` 、 `If-None-Match` 、304応答を説明します。

- RFC 5861：HTTP Cache-Control Extensions for Stale Content：意図して古い応答を提供できる、二つのディレクティブを説明します。
- RFC 8246：HTTP Immutable Responses：内容が変わるとURLも変わるファイルのための、`immutable` ディレクティブを説明します。
- Workbox：キャッシュ戦略：サービスワーカーでのキャッシュ優先、ネットワーク優先、stale-while-revalidateを、コードとともに説明します。
- PostgreSQL：マテリアライズドビュー：クエリ結果をデータベース内に保持し、必要に応じて更新する仕組みを説明します。

用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「キャッシュ処理とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/caching/