Engineer

Redis

一時的なデータをメモリーに保持するデータストア

どのようなものか

Redisは、データセット全体を、型のある構造としてメモリーに保持するデータベースサーバーです。複数のアプリケーションのプロセスが、ネットワーク越しに、短いコマンドを一つずつ使って、読み書きできます。プロジェクト自身は、データ構造サーバーと説明しています。無償で入手できる版はRedis Open Sourceと呼び、Redis Ltdは、商用ライセンスのRedis CloudとRedis Softwareも販売しています。

一つのキーは、一つの型の値を一つ持ちます。文字列は単なるバイト列で、カウンターにも使います。ハッシュはフィールドと値からなるレコード、リストは到着順を保つもの、集合は重複しない要素を持つもの、ソート済み集合はスコアで要素を順位付けするものです。型の一覧には、ほかにも、ストリーム、JSON文書、地理空間索引、時系列、ベクトル集合、BloomフィルターやHyperLogLogのような確率的データ構造があります。

コマンドは、これらの型を直接操作します。`INCR`は、カウンターの値を一つ増やして新しい値を返し、`SADD`は集合へ要素を入れ、`SPOP`は一つ取り出します。どのキーにも有効期限を設けられ、過ぎるとRedis自身が削除します。Redisは、そのようなキーをvolatileと呼びます。複数の手順を一つとして行う必要があれば、サーバー内でLuaスクリプトを動かせます。Redisは、その実行中に、ほかの処理が動かないことを保証します。

データの置き場所はメモリーで、ディスクへの保存は任意です。Redisは、RDBファイルと呼ぶ、ある時点のスナップショットを書けます。また、すべての書き込みを追記専用ファイル（AOF）へ記録し、起動時に再生することもでき、両方を使っても、どちらも使わなくてもかまいません。レプリカはプライマリーを複製し、読み取りに応答できます。メモリーの上限`maxmemory`と、キーの削除ポリシーが、割り当てた容量をデータが超えたときの動きを決めます。

ライセンスは、Redis 8で変わりました。このリリース以降、Redis Open Sourceは、RSALv2、SSPLv1、AGPLv3の三つから選べます。最初の二つは、ソースコードを公開するsource-availableライセンスで、オープンソースの選択肢はAGPLv3です。

概念を判断に生かす

「Redis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

Webアプリケーションが、一つのプロセスだけで動くことは、あまりありません。ロードバランサーの背後に複数のインスタンスがあり、それぞれにメモリーがあります。ログイン試行の回数、セッション、キャッシュしたページなど、そこに置いたものは、たまたまリクエストを処理したプロセスにしか存在しません。そのようにプロセスごとに回数を数える制限は、別のインスタンスへ到達すれば、すり抜けられます。Redisは、その種類の状態を、全インスタンスが共有する一か所に置き、リクエストごとに参照できる速さで提供します。

アトミックなコマンドは、速度と同じくらい重要です。二つのインスタンスが、それぞれカウンターを読み、一つ足して書き戻すと、同じ数を書いてしまい、一回分の増加が消える場合があります。 `INCR` は、読み取りと書き込みを、サーバー内で一つの手順として行うため、百件のリクエストが同時に来れば、カウンターは、正確に百だけ増えます。Luaスクリプトは、短い連続した処理にも、同じ保証を広げます。まだ空きがあるか確かめて枠を取る、といった、分けてはいけない検査と変更を、一つとして実行します。

三つ目は、有効期限です。レート制限の時間枠、セッション、使い捨てのコードは、時間が来たら消えるべきものです。キーに有効期限を設ければ、片付けは、定期実行タスクではなく、保存先の仕事になります。

代償は、永続性で、選ぶ前に数値で把握する価値があります。追記専用ファイルを使い、既定の、毎秒一回fsyncする方針では、クラッシュで約一秒分の書き込みを失う場合があります。通常、数分ごとに取るスナップショットだけなら、ドキュメントは、「直近数分間のデータを失うことに備えるべき」としています。

レプリケーションは既定で非同期なので、プライマリーが書き込みに成功応答を返した後、どのレプリカにも届く前に止まることがあります。そのため、再構築できる状態や、失っても支障のない状態には適しますが、注文、決済、残高の唯一の保存先には適しません。設計で最も重要な問いは、Redisに届かない場合や、覚えさせたものを忘れた場合に、アプリケーションが何をするかです。

実際の活用例

企業と専門家を載せる公開ディレクトリーでは、22個のレートリミッターが、Redisに状態を置きます。簡単にすり抜けられる、単一プロセス内の制限ではなく、全インスタンスにまたがって効かせるためです。ログイン試行、検索、一括エクスポートでは、不正利用になる頻度が大きく違うため、各リミッターを、保護する経路に合わせて作っています。決め手は、Redisが使えないときの動作です。リミッターは、障害時に閉じ、通信を素通りさせず、拒否します。その選択には、目に見える代償があります。Redisの問題が、利用者に見える問題になりますが、意図して受け入れています。別の選択をすれば、Redisの問題が、課金の問題になるためです。

限りある在庫を売る場合、Redisを、データベースの手前の速いゲートとして置けます。Luaスクリプトで減らすRedisのカウンターが、在庫数とほぼ同数の購入者を通し、残りをマイクロ秒単位で拒否します。スクリプトは、確認と減算を一つの手順にします。ただし、ゲートが約束する範囲は、見た目より小さいものです。Redisの複製は非同期なので、フェイルオーバー後の新しいプライマリーは、通した人の一部を忘れ、少し多く通す可能性があります。各販売を最終的に決めるのは、データベースの条件付き `UPDATE` で、照合処理が、データベースの数値からゲートを設定し直します。番号付きの座席のように、数が少なく、個々を区別する在庫では、IDをRedisの集合に置き、競合するカウンターを使わず、`SPOP` で一つずつ渡せます。

最も一般的な役割は、単純なキャッシュです。Bicepで定義するAzureの構成では、テスト（testing）、ステージング（staging）、本番（product）のどの環境も、同じテンプレートから構築します。それぞれに、キャッシュ用のRedis、PostgreSQL Flexible Server、apiとWebサイトを動かすContainer Apps、WAFポリシー付きのFront Doorがあります。キャッシュも、ほかの構成と一緒にテンプレートで宣言するため、変更は、環境のほかの変更と同じく、プルリクエストになります。

問題が起きるところ

Redisの問題の多くは、速くても忘れることのある保存先を、永続的なものとして扱うか、時間について伝えたことを覚えていると思い込むことから起きます。

期限が来たら消えるはずのキーが、残り続けます。 `SET` は、値を置き換え、それまでの有効期限を捨てるため、単純な `SET` でキャッシュ値を更新すると、有効期間が十分間の項目が、恒久的なものになります。

```
> SET session:42 "v1" EX 600
OK
> SET session:42 "v2"
OK
> TTL session:42
(integer) -1
```

有効期限が必要な書き込みには、毎回指定するか、6.0.0から使える `KEEPTTL` を加え、以前の期限を保ちます。同じ問題は、 `INCR` の後に、別の `EXPIRE` を送るカウンターにも潜みます。二つ目のコマンドが届かなければ、キーは、いつまでも消えません。両方を一つの `MULTI` と `EXEC` のブロックで送るか、一つのLuaスクリプトに含めます。

読み取りは動くのに、書き込みがエラーを返し始めます。 データが `maxmemory` に達し、キーの削除ポリシーが `noeviction` か、有効期限のあるキーが一つもないデータセットに対する `volatile-` の方針になっています。後者も同じ動作になります。用途に合わせて選びます。どのキーを消してもよい純粋なキャッシュなら、 `allkeys-` の方針を使い、キャッシュ用のキーと、残すべきキーが、一つのインスタンスを共有する場合にだけ、 `volatile-` を使います。そのように混在させる場合、ドキュメントは、二つのインスタンスへ分けることを検討するよう勧めています。

成功応答を受けた書き込みが、再起動やフェイルオーバー後にありません。 fsyncまでの間にクラッシュしたか、書き込みが届く前にレプリカが昇格しました。Redisは、そのような設計なので、対策は、何を保存するかにあります。残らなければならないものは、永続的にコミットするデータベースへ保存し、その手前に、ゲートかキャッシュとしてRedisを置き、在庫のノートと同じく、データベース側から両者を照合します。

すべてのクライアントが、同時に止まります。 長いLuaスクリプトが動き、Redisは、その全実行時間にわたり、そのサーバーのほかの処理をすべてブロックします。スクリプトを短く、処理量に上限のあるものにし、大きなコレクションを走査するループを入れないようにします。

障害時に、リミッターが通信を通してしまいます。 クライアントのコードが、Redisのエラーを「上限に達していない」と扱い、保存先に届かないときにこそ、ゲートが開きます。障害時の動作を意図して決め、Redisを止めてテストします。高くつく経路を守るリミッターは、前述の22個と同じように、拒否すべきです。その代償として、Redisの問題が利用者に見えます。

参考資料

- Redisでの開発 — 試せるコマンド、言語ごとのクライアントライブラリー、データ型を紹介する、プロジェクトの入門です。
- Redisの永続化 — スナップショット、追記専用ファイル、fsyncの方針、それぞれで失い得るデータです。
- Redisのレプリケーション — レプリカがプライマリーに追従する方法と、データを失う余地が残る理由です。
- キーの削除ポリシー — `maxmemory` 、すべてのキー削除ポリシー、選択の助言です。
- EXPIRE — キーの有効期限と、それを消すコマンド、保つコマンドです。
- Luaスクリプト — サーバーでスクリプトをアトミックに実行する方法と、その間にブロックするものです。

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

「Redis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/redis/