Engineer

レート制限

一定時間内に一つのクライアントが送れるリクエスト数の上限

どのようなものか

レート制限は、一つのクライアントが、ある時間内にサービスへ送れるリクエスト数の上限です。サービス自身か、その手前に置くものが適用します。どの制限も、三つの問いに答えます。誰をクライアントとするか、リクエストをどう数えるか、超過したリクエストをどうするかです。最後の問いに対応するHTTPステータスコード、429 Too Many Requestsは、2012年4月のRFC 6585で定義されました。同RFCは、最初の二つを意図的に決めていません。サーバーは、リソースごと、サーバー全体、サーバーのグループ全体で数えてもよく、認証情報や、状態を保持するCookieでクライアントを識別してもかまいません。

実際に使うキーは、いくつかの候補に絞られます。クライアントのIPアドレス、ログインしたアカウント、APIキー、あるいは、そのどれかとパスの組み合わせです。組み合わせれば、ログインフォームと商品ページに違う制限を設けられます。超過したリクエストへの対応には、三種類あります。一つは、429で拒否する方法です。できれば `Retry-After` ヘッダーを付け、秒数かHTTP日付で待ち時間を伝えます。二つ目は、リクエストを待たせ、少し後に処理する方法です。突発的な集中を拒否せず、ならせます。三つ目は、量そのものが攻撃である場合に、応答せず接続を切る方法です。RFC 6585も、その対応を提案しています。大量のリクエストすべてに、丁寧な429を書くにも、リソースがかかるためです。

数え方によって、境界での制限の動きが決まります。固定ウィンドウは、時計の分や時間の区切りごとに数え、境界でゼロから数え直します。処理が軽く、説明しやすい一方、クライアントは、一つの区間の最後の一秒に全枠を使い、次の区間の最初の一秒にも全枠を使えます。一分100件なら、二秒で200件です。スライディングウィンドウは、常に現在から同じ長さだけ過去を振り返るため、上限をすり抜けるために使える区間の境界がありません。HAProxyのスティックテーブルは、この種類のカウンターを持ち、マニュアルの一例では、30秒のスライディングウィンドウで接続率を数えます。nginxが使うリーキーバケットは、一定の平均レートでリクエストを通し、それより速く来たものを小さいキューに置

き、あふれるものを拒否します。トークンバケットは逆で、トークンを一定の速度で上限までため、リクエストごとに一つ使います。しばらく通信しなかったクライアントは、たまった分を一度に使えます。

概念を判断に生かす

「レート制限とは？」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。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

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。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

公開したエンドポイントは、誰が、どの頻度で呼ぶかを選べません。カタログを複製するスクレーパー、再試行のループから抜けられないスクリプト、漏れたパスワードをログインフォームで試す攻撃者、急いで夜間処理を書いた取引先は、サーバーから見ると同じです。少数の送信元から、想定容量を超えるリクエストが来ます。制限がなければ、最も多く呼ぶ相手が、ほかの全員に残る容量を決めます。制限があれば、一つのクライアントは自分の分しか使えず、その相手が待つ間も、ほかの利用者向けのサービスは通常の速度で続きます。

ほかよりはるかに高くつくリクエストもあります。商品ページはキャッシュから一ミリ秒で返せても、検索、エクスポート、有料の外部APIを呼ぶ処理は、呼び出すたびにCPU時間を使うか、費用が発生します。不正利用する側は、自分には費用がかからないからこそ、その経路を狙います。一つの全体共通の制限を、エクスポートを守れるほど低くすると、通常の閲覧まで妨げます。そのため、役立つレート制限はパスごとに設け、各経路のコストと正当な使い方に応じて上限を選びます。

制限は、APIの契約の一部でもあります。規則を知るクライアントは、自らペースを調整し、一度も上限に達せずに済みます。拒否されて制限を知るより、双方のコストが下がります。HTTPには、そのための標準ヘッダーがありません。多くのAPIが `X-RateLimit-Limit` 、 `X-RateLimit-Remaining` 、 `X-RateLimit-Reset` を送りますが、同じ名前でも実装ごとに値が違うことを、標準化を目指すIETFのドラフトが記録しています。ドラフトは、枠と時間幅を表す `RateLimit-Policy` 、残りを表す `RateLimit` という二つのフィールドを定義します。第十一版の日付は2026年5月23日で、ほかのInternet-Draftと同じく、標準ではなく、作業中の文書です。

実際の活用例

nginxでは、二つのディレクティブで制限を設けます。一つは `http` ブロックでカウンターを用意し、もう一つは、適用先に置きます。

```
limit_req_zone $binary_remote_addr zone=perip:10m rate=1r/s;
limit_req_status 429;

server {
    location /search/ {
        limit_req zone=perip burst=5 nodelay;
        proxy_pass http://app;
    }
}
```

キーは、クライアントのアドレスをバイナリー形式で表したもので、長さは4バイト（IPv4）か16バイト（IPv6）です。カウンターは、 `perip` という10メガバイトの共有メモリー領域に置きます。ドキュメントによると、一つのキーの状態は128バイト（64ビット環境）で、一メガバイトに約8,000アドレス、この領域には約80,000アドレスが入ります。いっぱいになると、nginxは、最後の利用が最も古いものを追い出します。 `rate=1r/s` は、平均して毎秒一件を許可します。 `burst=5` は、超過したリクエストを五件まで拒否せず待たせ、 `nodelay` は、設定したレートで間隔を空ける代わりに、すぐ処理します。バーストの枠を超えたものは拒否します。nginxは、既定では503を返すため、 `limit_req_status` で429にしています。毎秒一件より低いレートは、一分当たりで書き、 `30r/m` は、二秒に一件です。新しい制限を本番へ適用する前に、 `limit_req_dry_run on` を使うと、通常どおり数えながら、拒否するはずのものを、 `REJECTED_DRY_RUN` として `$limit_req_status` 変数に記録し、実際には何も拒否しません。

応答を受け取る側では、上限に達したクライアントは、即座に再試行するのではなく、応答を読むべきです。IETFのドラフトに従うAPIの429には、たとえば次のような形が考えられます。

```
HTTP/1.1 429 Too Many Requests
Content-Type: application/problem+json
Retry-After: 20
RateLimit-Policy: "default";q=100;w=60
RateLimit: "default";r=0;t=20
```

この方針の上限は100件で、ウィンドウは60秒です。残りはゼロで、20秒後に枠が戻ります。両方ある場合は、ドラフトによると、`Retry-After`が優先します。適切に動くクライアントは、少なくともその時間だけ待ち、小さいランダムな遅延を加えます。`Retry-After`自体がなければ、固定間隔ではなく、指数バックオフへ切り替えます。

障害時に閉じる22個のレートリミッターは、企業と専門家を載せる公開ディレクトリーのために、2025年からの業務で構築したもので、パスごとに分ける規則に従っています。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、それぞれ保護する経路に合わせて作りました。状態はRedisに置くため、プロセスごとではなく、すべてのインスタンスにまたがって制限が効きます。決定的な選択は、Redisが停止したときの動作でした。リミッターは、通信を素通りさせず、拒否します。ページは、代償を率直に述べます。Redisの問題が、利用者に見える問題になります。受け入れた理由も説明しています。ほかの選択肢では、Redisの問題が、課金の問題になるためでした。周囲には、Cloudflare Turnstile、オリジンのロック、再試行した書き込みを二度適用しないための、351行の冪等性ミドルウェアがあります。

問題が起きるところ

サーバーごとに、別々に数えます。 プロセスのメモリーに置く制限は、アプリケーションの複製が増えるほど、通せる数も増えます。ロードバランサーの背後に四つのインスタンスがあれば、クライアントを予定の四倍のレートで通し、再起動のたびにカウンターも戻ります。nginxの領域は、一台のマシン内の共有メモリーで、サーバー間で領域を同期させる機能は、有料の契約に含まれます。HAProxyのpeersプロトコルは、インスタンス間でスティックテーブルの項目を複製しますが、マニュアルによると、合算はしません。先ほどのディレクトリーがRedisで行うように、全インスタンスが参照する保存先へカウンターを置くか、すべての通信が通る一か所で制限を適用します。

全員が、同じアドレスから来ます。 リバースプロキシ、CDN、ロードバランサーの背後では、元のアドレスを引き継がなければ、アプリケーションから見えるクライアントアドレスは、全利用者で一つです。IPごとの制限が、全員の通信を一斉に制限します。NATの背後にある、オフィス全体や携帯電話網でも、程度は小さくても同じことが起きます。自身のプロキシだけが設定できるヘッダーから本当のクライアントアドレスを取り、そのアドレスをキーにします。ログイン済みの通信では、アドレスよりアカウントやAPIキーを優先します。

リミッターが、サーバーの故障を報告します。 nginxが、拒否したリクエストへ返す既定の応答は、503 Service Unavailableです。HTTPでは、サーバーが過負荷か、保守で停止していることを意味します。5xx応答をサーバーエラーとするダッシュボードやアラートは、一つのクライアントの過剰利用で誰かを起こし、クライアントは、自分の動作について何も分かりません。原因がクライアント側にあると示す429を返し、拒否を、専用のログレベルで記録します。

制限したクライアントが、同じ秒に集中して戻ってきます。 枠が一定の時刻にリセットされる場合や、拒否した全員へ同じ `Retry-After` を伝える場合、全員が一緒に戻ります。IETFのドラフトは、毎日18:00:00にリセットすると、全員が18:00:00に現れがちだという例を示します。その後の集中が、制限した集中よりひどくなる場合もあります。両側にジッターを加えます。サーバーが知らせる待ち時間を変動させ、クライアントにも、独自のランダムな遅延を加えさせます。

保存先が停止したときの方針を、誰も決めていません。 Redisやデータベースで数えるリミッターは、保存先に到達できないときの動作が必要です。接続エラーを捕まえてリクエストを通すコードは、障害時に開く方を選んでおり、見張っている相手に、回数無制限で通信できる期間を与えます。拒否するコードは、障害時に閉じる方を選び、全員のサービス停止に変えます。どちらも、常に誤りというわけではありません。ただし、先ほどのディレクトリーと同じく、パスごとの設計判断として選ぶべきで、ライブラリーの既定動作に任せるべきではありません。

拒否を数えることで、情報が漏れます。 IETFのドラフトは、401や403の応答が枠を消費すると、悪意のあるクライアントが、エンドポイントを探って、別の利用者の通信について知り得ると警告します。どの応答を数えるかを決め、失敗した認証には独自の制限を設け、認証していないクライアントへ枠の詳細を見せないようにします。

参考資料

- RFC 6585：追加のHTTPステータスコード — 429 Too Many Requestsを定義し、利用者の識別とリクエストの数え方を、サーバーに委ねることを明記しています。
- RFC 9110：HTTPの意味論 — `Retry-After` フィールドと二つの形式、およびサーバーが送る503の意味です。
- HTTPのRateLimitヘッダーフィールド — IETF HTTPAPI作業部会による `RateLimit-Policy` と `RateLimit` のドラフトで、同時の再アクセスと情報漏えいについてのセキュリティー節もあります。

- ngx_http_limit_req_moduleモジュール — `limit_req` 、 `burst` 、 `nodelay` 、 `delay` 、ドライランモード、共有メモリー領域のサイズです。
- HAProxy 3.4設定マニュアル — スティックテーブルとレートカウンター、およびインスタンス間で内容を複製するpeersプロトコルです。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「レート制限とは？」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/rate-limiting/