Engineer

負荷テスト

模擬的な通信を受けるシステムの動作を測るテスト

どのようなものか

負荷テストは、予定した量の模擬通信をシステムへ送り、それを受け持つ間の動作を測ります。応答時間、失敗したリクエスト数、維持できるスループット、サーバーが使う資源などです。スクリプトが多数の利用者を同時に演じ、k6は、それぞれを仮想ユーザーと呼びます。負荷テストを行うという活動に対し、一回のテストは、負荷の形、実行時間、合格条件を定めた一度の実行です。

広い意味でも、狭い意味でも使う言葉です。Grafana Labsのk6文書は、かける負荷で分類します。スモークテストは、少数の仮想ユーザーで、スクリプトが動くことを確かめます。平均負荷テストは、通常の本番の一日を模擬します。ストレステストはその平均を超え、ソークテストは平均負荷を何時間も保ち、スパイクテストは、突然の非常に大きい負荷を数分間与え、限界点テストは、何かに支障が出るまで負荷を上げ続けます。「負荷テスト」が模擬通信の試験全般を指すため、k6は、狭い意味を平均負荷テストと呼び、通常の一日を試すテストや、ボリュームテストと呼ぶチームもあると記します。ガイドも「これらの試験の名称にさえ合意がない」と率直です。分類も相対的で、一つのアプリケーションに厳しい負荷が、別のものには普通の一日です。

負荷テストは、機能テストではありません。購入手続きの処理が買い物かごの合計を正しく計算するかでなく、現実的な数の利用者が使う間も応答するか、どれほど速いかを問います。両方が必要です。不正なリクエストを高速に送るスクリプトは、拒否する速さだけを測るからです。

概念を判断に生かす

「負荷テストとは？」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。 3社の間で受け渡す代わりに、1つのチームがアイデアのスケッチから、お客様がログインして使える製品まで担当します。

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。 GIS地図アプリの開発を主導し売上を10倍に — 10x 地図製品の開発を主導して実現した売上成長

GIS地図アプリの開発を主導し売上を10倍に お問い合わせ →

なぜ重要なのか

どのシステムにも、処理が追い付かなくなる通信量があり、問題は誰が先に見つけるかです。テストを普通のPythonで書くオープンソースの負荷テストツールLocustは、入門文書でその挙動を示します。ユーザーを増やすと、例では約20人のところで、応答時間が急増し、さらに人が来ても、毎秒のリクエスト数が増えなくなります。文書は、対象が過負荷、または飽和したと呼びます。この折れ曲がる点が、構成図の約束でなく、実装したシステムの処理能力です。顧客の前でなく、事前の試験で見つける方が、はるかに安く済みます。

折れ曲がる点へ向けて増やす過程だけでも、情報があります。k6の平均負荷テストのガイドは、まだ負荷を増やしている間に、最初の結果が出ると説明します。通信量とともに応答時間が悪くなるか、予定の負荷に届く前に失敗するかです。その後の一定負荷の区間では、性能と資源使用が安定するかを調べます。まさにその間に、不安定になるシステムもあるためです。

負荷テストは、性能を意見でなく、合否の条件にもします。k6では、失敗するリクエストが1%未満、応答の95%が200 ms以内など、指標の合否を決めるしきい値をスクリプトに書けます。k6の文書によると、サービスレベル目標を、こうしたしきい値で記すこともよくあります。k6は、すべて満たせば終了コード0、一つでも満たさなければゼロ以外で終了します。グラフを誰も読まずに、サービスを遅くしたリリースを、パイプラインが拒否できます。

実際の活用例

特売を準備する店を考えます。大半の通信は、商品を買えない人からなので、耐えるべき応答は「売り切れ」で、その経路を先に試します。次はk6で書いたテストです。固定人数でなく、指定した率でリクエストを開始し、三分かけて毎秒2,000件まで増やし、二十分保ってから減らします。平均負荷に関するk6のガイドどおり、増加は全体の5から15%、一定区間は少なくとも増加時間の五倍、減少も増加とほぼ同じ長さです。チケット販売、製品公開、季節の特売にk6が勧めるスパイクテストは、その後、ほぼ立ち上がりなしで行う想定です。この試験は、まず一定の率で、売り切れの応答が持ちこたえることを確かめます。

```
import http from 'k6/http';

export const options = {
  scenarios: {
    sold_out: {
      executor: 'ramping-arrival-rate',
      startRate: 0,
      timeUnit: '1s',
      preAllocatedVUs: 400,
      stages: [
        { target: 2000, duration: '3m' },
        { target: 2000, duration: '20m' },
        { target: 0, duration: '3m' },
      ],
    },
  },
  thresholds: {
    http_req_failed: ['rate<0.01'],
    http_req_duration: ['p(95)<200'],
    dropped_iterations: ['count<1'],
  },
};

export default function () {
  http.get('https://shop.example/sale/items/4711');
}
```

`k6 run sold-out.js` で実行します。仮想ユーザー数は推計で、k6の文書は、反復時間の中央値に率を掛け、余裕を足す式を示します。毎秒2,000件、一件が約十分の一秒なら、同時に200人が稼働し、400人なら余裕があります。応答が遅ければ、一反復が長くなり、プールが尽き、空きユーザーがいない各反復を、k6が `dropped_iterations` へ記録します。三つ目のしきい値は、そのためです。落とした反復があれば、試験は主張する負荷を送れていません。k6も、生成器が反復を落とすときは、システム自身が遅くなっている可能性があると記します。残る二つは、リクエストの失敗が1%以上、または第95パーセンタイルが200 ms以上なら、実行を不合格にします。

この想定は、最後のチケットを、厳密に一度だけ売るという、このサイトのノートに基づきます。百万人の買い手に一万個なら、990,000件を拒否するため、拒否が最も安い応答でなければならないと説明します。設計を証明する節には、拒否経路の負荷テストもあります。通信の99%になっても売り切れの応答は速く、ネットワークの端で吸収する必要があります。購入成功だけを試すのは、処理能力を見極めるうえで最も重要度の低い1%だけを測ることです。

問題が起きるところ

測る相手を、試験が待ちます。 k6がクローズドモデルと呼ぶ方式では、各仮想ユーザーは、前の反復が終わるまで次を始めません。サーバーが遅くなると、ユーザーも遅くなり、到着率が下がって、予定より穏やかな試験になります。k6によれば、文献の一部で、これを応答待ちによる測定漏れ（coordinated omission）と呼びます。実際の訪問者は、互いを待ちません。一定のリクエスト率に耐えられるかが問いなら、応答時間にかかわらず、予定どおり反復を始める、`constant-arrival-rate` や `ramping-arrival-rate` などのオープンモデルの実行器を使います。

負荷生成器がボトルネックになります。 テストのマシンで、仮想ユーザー、CPU、メモリーが足りなければ、測るのは対象でなく生成器です。k6は、大半の場合、`maxVUs` を勧めません。実行中のユーザー追加はCPUとメモリーを使い、生成器を過負荷にし、結果を歪め得るためです。少数のローカル試行で、先に `preAllocatedVUs` を決め、`dropped_iterations` を見ます。一台で足りなければ、Locustは、一つのマスタープロセスと多数のワーカーで、複数台へ負荷を分けます。

最初の実行で、最大の負荷をかけます。 k6の文書によると、何千人もの利用者が、アプリケーションやステージング環境をすぐに落とす負荷テストを実行してきました。少数の仮想ユーザーのスモークテストで、スクリプトと環境を確かめ、次に平均負荷、その後で初めてストレステストを行います。k6は、この順序を不可欠とします。低い負荷でも出る問題は、高くする前の方が、ずっと見つけやすいものです。

負荷が、ジェットコースターのように上下します。 何度も上下する形は、資源を無駄にし、どの変化が、どの症状を生んだかを分かりにくくします。k6は、一度に一方向を勧めます。増やし、一定に保ち、減らします。実行間でも、想定、データ、環境を同じにして、二つの結果を比較できるようにします。

合格の意味を決めていません。 基準がなくグラフだけが残る実行結果は、人によって解釈が異なり、たいてい甘く評価されます。サービスの目標に基づき、実行前にしきい値を書き、終了コードで決めます。長い試験では、k6の `abortOnFail` で、しきい値に違反したらすぐに中止できます。`delayAbortEval` は、意味のある数の標本が集まる前の中止を防ぎます。

一種類のテストを、答えのすべてとみなします。 平均負荷テストは、数秒で殺到する特売をほとんど説明せず、スパイクテストは、一定負荷の数時間後にだけ出る不具合を、ほとんど説明しません。k6は、一種類で全リスクを除けず、リス

クの性質が組み合わせを決めると明言します。長時間利用にリスクがあれば、まずソークテストを、集中利用にあれば、ストレステストを必要とします。急増時には、普段は一度も動かない処理が動くこともあり、スクリプトには、それも含めます。

参考資料

- 負荷テストの種類 — スモーク、平均負荷、ストレス、ソーク、スパイク、限界点の各テストを、典型的な負荷と長さで示す、k6の概要です。
- 平均負荷テスト — 増加、一定区間、減少の形と、それぞれから読み取るものです。
- しきい値 — k6の合否基準、その構文、しきい値違反が終了コードを決める方法です。
- オープンモデルとクローズドモデル — 応答を待つユーザーを持つ試験が、遅いシステムの問題を過小評価する理由です。
- 到着率に合わせた仮想ユーザーの確保 — オープンモデルに必要な人数と、開始できずに落とした反復の意味です。
- Locustとは — Pythonで書く負荷テストツールと、その分散実行についての、Locustの文書自身による説明です。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「負荷テストとは？」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/load-test/