Engineer

Lasttest

Test des Systemverhaltens unter simuliertem Datenverkehr

Was es ist

Ein Lasttest schickt eine geplante Menge simulierten Datenverkehrs an ein System und misst dessen Verhalten unter dieser Last: wie lange Antworten dauern, wie viele Anfragen scheitern, welchen Durchsatz es aufrechterhält und welche Ressourcen die Server dafür verbrauchen. Der Verkehr kommt aus einem Skript, das viele Nutzer zugleich simuliert. Das Werkzeug k6 nennt jeden davon einen virtuellen Nutzer. Lasttests durchzuführen ist die Tätigkeit; ein Lasttest ist ein einzelner Durchlauf mit einem Lastprofil, einer Dauer und Kriterien zum Bestehen.

Der Begriff wird weit und eng verwendet. Die von Grafana Labs veröffentlichte k6-Dokumentation ordnet Tests nach ihrer Last. Ein Smoke-Test lässt eine Handvoll virtueller Nutzer laufen, um zu zeigen, dass das Skript funktioniert. Ein Test mit durchschnittlicher Last bildet einen gewöhnlichen Tag im Produktionsbetrieb nach. Ein Stresstest geht über diesen Durchschnitt hinaus, ein Dauerlasttest hält die durchschnittliche Last über Stunden, ein Lastspitzentest erzeugt für wenige Minuten einen plötzlichen sehr hohen Ansturm, und ein Test bis zur Belastungsgrenze erhöht die Last weiter, bis etwas nachgibt. Weil „Lasttest“ jeden Test mit simuliertem Datenverkehr meinen kann, nennt k6 die engere Art Average-Load-Test und merkt an, dass manche Teams Day-in-Life-Test oder Volumentest sagen. Der Leitfaden spricht die Begriffsfrage offen an: „Nicht einmal über die Namen dieser Testarten besteht Einigkeit.“ Auch die Kategorien sind relativ: Was eine Anwendung stresst, ist für eine andere ein gewöhnlicher Tag.

Ein Lasttest ist kein Funktionstest. Er fragt nicht, ob die Kasse den Warenkorb richtig zusammenrechnet, sondern ob sie noch antwortet und wie schnell, während eine realistische Zahl von Nutzern sie verwendet. Beide brauchen einander: Ein Skript, das fehlerhafte Anfragen mit hoher Geschwindigkeit sendet, misst nur, wie schnell das System Nein sagt.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Lasttest?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Ein Team führt eine Idee von der Skizze bis zu einem Produkt, bei dem Ihre Kunden sich anmelden können, statt sie zwischen drei Dienstleistern weiterzureichen.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. GIS-Kartenprodukt zu 10-fachem Umsatz geführt — 10x Umsatzwachstum durch die Leitung des Kartenprodukts

GIS-Kartenprodukt zu 10-fachem Umsatz geführt Kontakt aufnehmen →

Warum es wichtig ist

Jedes System hat ein Verkehrsaufkommen, mit dem es nicht mehr Schritt hält. Die Frage ist nur, wer es zuerst findet. Locust, ein quelloffenes Werkzeug für Lasttests mit Tests in gewöhnlichem Python, zeigt den Verlauf im Schnelleinstieg. Fügt das Werkzeug Nutzer hinzu, wird ein Punkt erreicht, im Beispiel bei ungefähr 20 Nutzern, an dem die Antwortzeiten so schnell steigen, dass die Zahl der Anfragen pro Sekunde nicht mehr zunimmt, obwohl immer mehr Nutzer hinzukommen. Die Dokumentation bezeichnet das getestete System als überlastet oder gesättigt. Dieser Knick ist die tatsächliche Kapazität des gebauten Systems, unabhängig von den Versprechen des Architekturdiagramms. Ihn bei einer Probe zu finden ist viel günstiger als vor Kunden.

Schon der Anstieg bis zum Knick ist aufschlussreich. Laut k6-Leitfaden zu Tests mit durchschnittlicher Last zeigt sich das erste Ergebnis bereits während der Laststeigerung: ob die Antwortzeit mit wachsendem Verkehr schlechter wird und ob das System schon vor Erreichen der vollen Last scheitert. Das Plateau zeigt anschließend, ob Leistung und Ressourcenverbrauch stabil bleiben. Manche Systeme werden genau in diesem Abschnitt unberechenbar.

Lasttests machen aus einer Meinung über Leistung außerdem eine verbindliche Qualitätsprüfung. In k6 kann ein Skript Grenzwerte deklarieren, also Bestehens- oder Fehlschlagskriterien für seine Messgrößen: etwa weniger als 1 % fehlgeschlagene Anfragen oder 95 % der Antworten innerhalb von 200 ms. Laut k6-Dokumentation halten Tester mit Grenzwerten häufig ihre Service Level Objectives fest. Werden alle eingehalten, endet k6 mit Code 0; scheitert einer, endet es mit einem Code ungleich null. Eine Pipeline kann dadurch eine Veröffentlichung verweigern, die den Dienst langsamer gemacht hat, ohne dass jemand einen Graphen liest.

Wie es in der Praxis aussieht

Betrachten Sie einen Shop vor einer Verkaufsaktion. Der meiste Datenverkehr während der Aktion kommt von Menschen, die den Artikel nicht bekommen werden. Die Antwort, die standhalten muss, lautet deshalb „ausverkauft“, und dieser Pfad sollte zuerst geprobt werden. Der folgende Test ist in k6 geschrieben. Er startet Anfragen mit einer festgelegten Rate statt einer festgelegten Nutzerzahl, steigert sie über drei Minuten auf 2.000 Anfragen pro Sekunde, hält sie zwanzig Minuten und senkt sie danach wieder ab. Der Anstieg folgt k6s Empfehlungen für Tests mit durchschnittlicher Last: Er umfasst 5 bis 15 % des Tests, das Plateau dauert mindestens fünfmal so lange wie der Anstieg, und das Absenken ungefähr genauso lange wie der Anstieg. Ein Lastspitzentest, den k6 für Ticketverkäufe, Produkteinführungen und saisonale Verkaufsaktionen empfiehlt, würde anschließend fast ohne Anstieg folgen. Dieser Test weist zuerst nach, dass die Ausverkauft-Antwort bei einer konstanten Rate standhält.

```
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');
}
```

Führen Sie ihn mit `k6 run sold-out.js` aus. Die Zahl virtueller Nutzer ist eine Schätzung, für die die k6-Dokumentation eine Formel angibt: die mediane Iterationsdauer multipliziert mit der Rate, zuzüglich einer Reserve. Bei 2.000 Anfragen pro Sekunde und ungefähr einer Zehntelsekunde je Anfrage sind 200 virtuelle Nutzer gleichzeitig beschäftigt; 400 lassen Spielraum. Werden Antworten langsamer, dauert jede Iteration länger, der Pool erschöpft sich, und k6 erfasst jede Iteration ohne freien virtuellen Nutzer in `dropped_iterations`. Deshalb gibt es den dritten Grenzwert: Eine ausgelassene Iteration bedeutet, dass der Test nicht die behauptete Last gesendet hat. k6 weist darauf hin, dass das System selbst langsamer werden könnte, wenn der Generator Iterationen auslässt. Die beiden anderen Grenzwerte lassen den Lauf scheitern, wenn mindestens 1 % der Anfragen fehlschlagen oder das 95. Perzentil bei 200 ms oder darüber liegt.

Das Szenario stammt aus der Notiz dieser Website über den genau einmaligen Verkauf der letzten Eintrittskarte. Sie hält fest, dass bei einer Million Käufern und zehntausend Einheiten 990.000 Anfragen scheitern müssen. Die Ablehnung muss deshalb die günstigste Antwort des Systems sein. Ihr Abschnitt zum Nachweis des Entwurfs umfasst Lasttests der Ablehnungen: Der Ausverkauft-Pfad muss schnell bleiben, wenn er 99 % des Datenverkehrs ausmacht, und der Rand des Netzwerks muss ihn abfangen. Ein Test, der nur einen erfolgreichen Kauf probt, misst das 1 %, das für die Kapazität am wenigsten zählt.

Wo es schiefgeht

Der Test wartet auf das System, das er misst. In dem, was k6 das geschlossene Modell nennt, beginnt jeder virtuelle Nutzer seine nächste Iteration erst, wenn die vorige beendet ist. Wird der Server langsamer, werden die virtuellen Nutzer ebenfalls langsamer. Die Ankunftsrate sinkt, und das System erhält einen sanfteren Test als geplant. Diesen Effekt bezeichnet ein Teil der Literatur laut k6 als Coordinated Omission. Echte Besucher warten nicht aufeinander. Geht es darum, ob ein System eine bestimmte Anfragerate bewältigt, verwenden Sie einen Executor mit offenem Modell wie `constant-arrival-rate` oder `ramping-arrival-rate`. Er startet Iterationen unabhängig von der Antwortzeit nach Zeitplan.

Der Lastgenerator ist der Engpass. Gehen dem Testrechner virtuelle Nutzer, CPU oder Speicher aus, beschreiben die Ergebnisse den Generator statt des getesteten Systems. k6 rät in den meisten Fällen von `maxVUs` ab: Während des Tests zusätzliche virtuelle Nutzer anzulegen kostet CPU und Speicher, kann den Generator überlasten und die Ergebnisse verzerren. Dimensionieren Sie

`preAllocatedVUs` vorab mit wenigen lokalen Probeläufen und beobachten Sie `dropped_iterations`. Reicht ein Rechner nicht, verteilt Locust die Last mit einem Masterprozess und vielen Workern auf mehrere Rechner.

Der erste Lauf ist der größte. Die k6-Dokumentation berichtet, dass Tausende ihrer Nutzer Lasttests ausgeführt haben, die ihre Anwendungen oder Staging-Umgebungen schnell zum Absturz brachten. Beginnen Sie mit einem Smoke-Test mit wenigen virtuellen Nutzern, um nachzuweisen, dass Skript und Umgebung funktionieren. Führen Sie danach einen Test mit durchschnittlicher Last aus und erst dann einen Stresstest. k6 nennt diese Abfolge unerlässlich. Probleme bei geringem Aufkommen lassen sich viel leichter finden, bevor die Last hoch ist.

Die Last fährt Achterbahn. Ein Verlauf, der mehrfach steigt und fällt, verschwendet Ressourcen und erschwert es, eine Änderung einem Symptom zuzuordnen. k6 empfiehlt jeweils nur eine Richtung: ansteigen lassen, Plateau halten, absenken. Halten Sie zwischen den Läufen auch Szenario, Daten und Umgebung gleich, damit sich zwei Ergebnisse überhaupt vergleichen lassen.

Niemand hat festgelegt, was Bestehen bedeutet. Einen Lauf, der mit einer Seite voller Diagramme und ohne Kriterien endet, liest jeder Betrachter anders, meist großzügig. Schreiben Sie die Grenzwerte vor dem Lauf anhand des Ziels auf, an dem der Dienst gemessen wird, und lassen Sie den Exitcode entscheiden. Bei langen Läufen kann k6 mit `abortOnFail` abbrechen, sobald ein Grenzwert verletzt ist. `delayAbortEval` verhindert den Abbruch, bevor genügend Messwerte für eine aussagekräftige Bewertung vorliegen.

Eine Testart gilt als ganze Antwort. Ein Test mit durchschnittlicher Last sagt wenig über eine Verkaufsaktion aus, die innerhalb von Sekunden einsetzt. Ein Lastspitzentest sagt wenig über einen Fehler, der erst nach Stunden gleichmäßiger Nutzung sichtbar wird. k6 erklärt ausdrücklich, dass keine einzelne Testart jedes Risiko beseitigt und das Risikoprofil die Mischung bestimmt: Bei Risiken durch lange Nutzung braucht ein System zuerst Dauerlasttests, bei Risiken durch intensive Nutzung Stresstests. Eine Lastspitze kann außerdem Prozesse auslösen, die an einem gewöhnlichen Tag nie laufen. Das Skript muss sie deshalb einschließen.

Weiterführende Lektüre

- Arten von Lasttests — k6s Übersicht über Smoke-Tests, Tests mit durchschnittlicher Last, Stresstests, Dauerlasttests, Lastspitzentests und Tests bis zur Belastungsgrenze mit der jeweils typischen Last und Dauer.
- Tests mit durchschnittlicher Last — wie Anstieg, Plateau und Absenken gestaltet werden und was sich aus jedem Abschnitt lesen lässt.
- Grenzwerte — k6s Bestehens- und Fehlschlagskriterien, ihre Syntax und wie ein verletzter Grenzwert den Exitcode festlegt.
- Offene und geschlossene Modelle — warum ein Test, dessen Nutzer auf Antworten warten, ein langsames System zu günstig darstellt.
- Zuweisung virtueller Nutzer bei vorgegebener Ankunftsrate — wie viele virtuelle Nutzer ein Test mit offenem Modell braucht und was ausgelassene Iterationen bedeuten.
- Was ist Locust? — die Beschreibung eines Python-Lasttestwerkzeugs und seines verteilten Modus durch die Locust-Dokumentation selbst.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Lasttest?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/load-test/