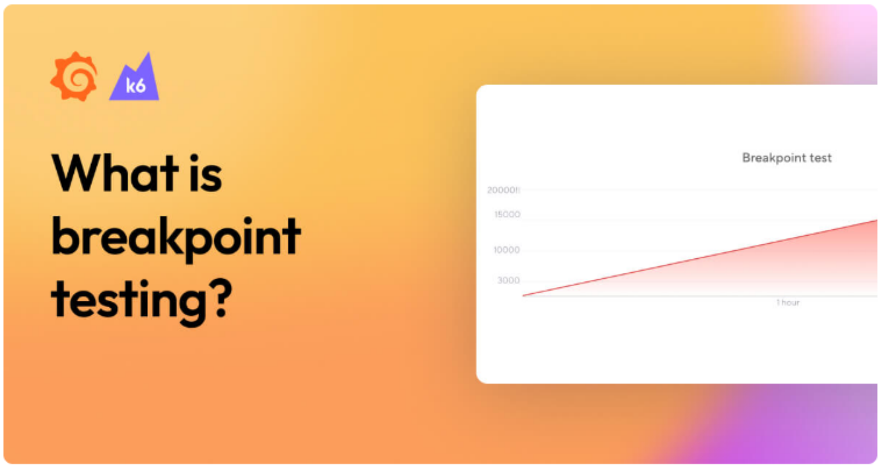


Breakpoint testing: A beginner's guide

Grafana Labs Team • 30 Jan, 2024 • 5 min

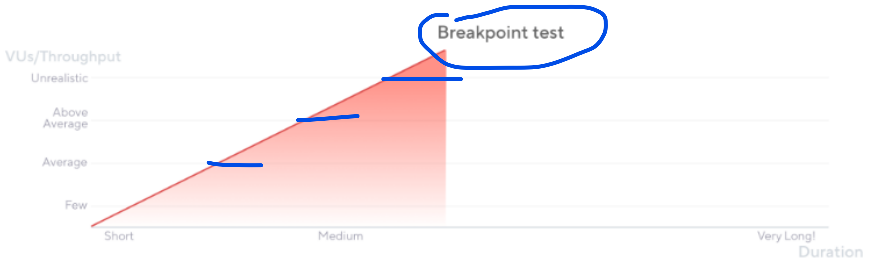
This content was originally published on k6.io.
30 Jan 2024

Breakpoint testing is a type of load testing that aims to find system limits. Reasons you might want to know the limits include:

- To tune or care for the system's weak spots to relocate those higher limits at higher levels.
- To help plan remediation steps in those cases and prepare for when the system nears those limits.

In other words, knowing where and how a system starts to fail helps prepare for such limits.

A breakpoint ramps to unrealistically high numbers. This test commonly has to be stopped manually or automatically as thresholds start to fail. When these problems appear, the system has reached its limits.



The breakpoint test is another type of load testing with no clear naming consensus. In some testing conversation, it's also known as *capacity*, *point load*, and *limit testing*.

When to run a breakpoint test

Teams execute a breakpoint test whenever they must know their system's diverse limits. Some conditions that may warrant a breakpoint test include the following:

- The need to know if the system's load expects to grow continuously
- If current resource consumption is considered high
- After significant changes to the code-base or infrastructure.

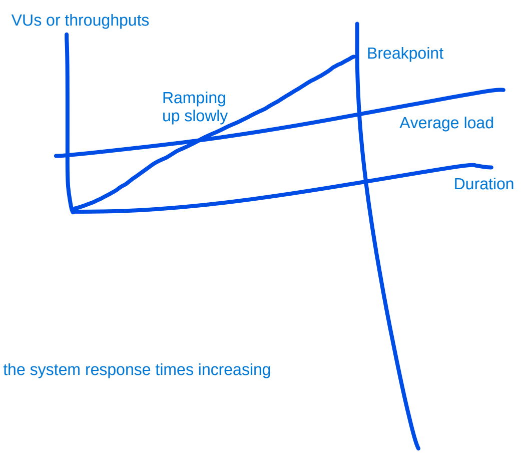
How often to run this test type depends on the risk of reaching the system limits and the number of changes to provision infrastructure components. New resources are provisioned

Once the breakpoint runs and the system limits have been identified, you can repeat the test after the tuning exercise to validate how it impacted limits. Repeat the test-tune cycle until the team is satisfied.

Considerations

- **Avoid breakpoint tests in elastic cloud environments**

- **Avoid breakpoint tests in elastic cloud environments.**
  An elastic cloud environment changes to reflect the needs of an organization, adjusting resources such as CPU, memory, and storage consumption as needed. An elastic environment may grow during breakpoint tests, finding only the limit of your cloud account bill. If you are running a breakpoint test on a cloud environment, *turning off elasticity on all the affected components is strongly recommended*.



- **Increase the load gradually.**
  A sudden increase may make it difficult to pinpoint why and when the system starts to fail.

- **System failure could mean different things to different teams.**
  You might want to identify each of the following failure points:

  - Degraded performance: The response times increased, and user experience decreased.
  - Troublesome performance: The response times get to a point where the user experience severely degrades.
  - Timeouts: Processes are failing due to extremely high response times.
  - Errors: The system starts responding with HTTP error codes.
  - System failure: The system collapsed.

All are measured by the system response times increasing

- **You can repeat this test several times.**
  Once you have tuned or made adjustments to your system after a breakpoint test, repeating the test might let you push the system further.

- **Run breakpoint tests only when the system is known to perform well under all other test types.**
  The breakpoint test might go too far if the system performs poorly with other load testing types.

Breakpoint testing in k6

The breakpoint test is straightforward. The load slowly ramps up to a considerably high level. It has no plateau, ramp-down, or other steps. And it generally fails before reaching the indicated point.

k6 offers two ways to increase the activity: increasing VUs or increasing throughput (open and closed models). Different from other load test types, which should be stopped when the system degrades to a certain point, breakpoint load increases even as the system starts to degrade. That makes it recommendable to use ramping-arrival-rate for a breakpoint test.

Increasing VUs - > Open model (Constant arrival rate, Ramping arrival rate)
Increasing throughput -> Close model

The test keeps increasing load or VUs until it reaches the defined breaking point or system limits, at which point the test stops or is aborted.

Here is a test script for the open source load testing tool Grafana k6:

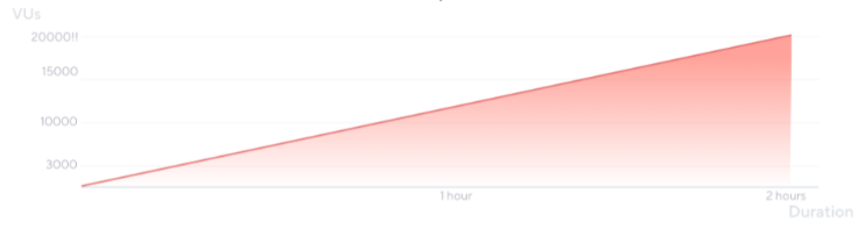
JavaScript

```javascript
import http from 'k6/http';
import {sleep} from 'k6';

export const options = {
  // Key configurations for breakpoint in this section
  executor: 'ramping-arrival-rate', //Assure load increase if the system slows
  stages: [
    { duration: '2h', target: 20000 }, // just slowly ramp-up to a HUGE load
  ],
};

export default () => {
  const urlRes = http.get('https://test-api.k6.io');
  sleep(1);
  // MORE STEPS
  // Here you can have more steps or complex script
  // Step1
  // Step2
  // etc.
```

Breakpoint test



The test must be stopped before it completes the scheduled execution. You can stop the test manually or with a threshold:

- To stop k6 manually in the CLI, press `Ctrl` + `C` in Linux or Windows, and `Command` + `.` in Mac.
- To stop the test using a threshold, you must define abortOnFail as true. For details, refer to the Thresholds .

Breakpoint testing: Results analysis

A breakpoint test must cause system failure. The load test helps identify the failure points of our system and how the system behaves once it reaches its limits.

Once the system limits are identified, the team has two choices: accept them or tune the system.

If the decision is to accept the limits, the test results help teams prepare and act when the system is nearing such limits.

These actions could be:

- Prevent reaching such limits
- Grow system resources
- Implement corrective actions for the system behavior at its limit
- Tune the system to stretch its limits

If the action taken is to tune the system, tune and then repeat the breakpoint test to find where and whether the system limits moved.

Overall, a team must determine the number of repetitions for the breakpoint test, how much the system can be tuned, and how far the system limits can be tuned after each exercise.

Grafana Cloud is the easiest way to get started with Grafana k6 and performance testing. We have a generous forever-free tier and plans for every use case. Sign up for free now!

Tags

k6 Load testing Performance testing

Grafana Cloud Logs

Grafana Cloud Metrics

Grafana Cloud Profiles

Grafana Cloud Synthetic Monitoring

Grafana SLO

Grafana Faro

Grafana Pyroscope

Grafana Beyla

OpenTelemetry

Grafana Tanka

Graphite

GitHub

GrafanaCON 2024

The Golden Grot Awards

Successes

Workshops

Videos

OSS vs Cloud

Load testing

Grafana Cloud Status

Legal and Security Terms of Service Privacy Policy Trademark Policy