<https://www.opentext.com/what-is/performance-engineering>

<https://testguild.com/performance-trend-2020/>

Performance engineering is proactive, continuous, and end-to-end application performance testing and monitoring. It allows seamless collaboration between teams, tools, and processes through continuous feedback loops. Here, it’s not just testers who are responsible for quality assurance but developers, performance engineers, product owners, and business analysts as well.

[Todd DeCapua](https://twitter.com/appperfeng), author of [*Effective Performance Engineering*](https://www.oreilly.com/library/view/effective-performance-engineering/9781492030164/), has said that performance engineering means understanding how all the parts of the system fit together, knowing which metrics matter, and building in performance quality from the first design.

The best approach to achieving chaos is to test how the system responds under stress proactively.

The idea of chaos engineering is that it compares what you *believe*will happen in your distributed system against the reality of what actually happens.

To learn how to build resilient business software systems, you can use a chaos test tool to break things in your environment on purpose to see if it actually fails the way you believed it would.

## Performance in CI/CD

So, where does that performance fit into a DevOps pipeline?

[Paola Rossaro](https://twitter.com/prossaro) Co-Founder and CTO at Nouvola believes there is a trend underway for performance tests to become part of your acceptance tests.

Continuous testing is really a key component of continuous integration and continuous delivery and creating testing at speed.

### What is the difference between Performance Testing and Performance Engineering?

Classic performance testing is effectively a subset of performance engineering. It usually entails running a single round of OpenText™ [load testing](https://www.microfocus.com/products/loadrunner-professional/overview) as part of the post-development quality assurance (QA) cycle. Performance testing involves checking the speed, reliability, scalability, stability, response time, and resource use of an application under the anticipated workload. Before we get into the differences between performance engineering and performance testing, we’ll first take a look at performance testing in isolation and why it’s, by itself, no longer sustainable.

Now, let’s delve into the key differences between performance engineering vs performance testing.

* First, performance testing is a quality check of the application’s load handling and responsiveness. It establishes how well the system will bear a production load and anticipates the issues that could come up during heavy load conditions. Performance engineering seeks to design the application from the start with performance metrics in mind and facilitate the discovery of issues early in development.
* Second, performance testing is a QA process that usually takes place when a round of software development is complete. Performance engineering is a continuous process that is embedded in all phases of the software development cycle – from design, to development, and into the end-user experience.
* Third, performance testing is conducted by the QA team while performance engineering involves RND and QA.

<https://www.dynatrace.com/resources/ebooks/javabook/performance-engineering/>

<https://www.radview.com/blog/performance-engineering-explained/>

A key aspect of Performance Engineering is the inclusion of Performance Tuning.

Endurance

he capacity of something to last or to withstand wear and tear.

* Establishing the testing performance metrics and criteria: memory usage, CPU usage, response time, throughput, etc.

## What is Performance Tuning?

Performance Tuning is essentially the step that is taken after identifying performance issues during the Performance Testing phase. After detecting the exact problems that the system has and getting to their root cause, you can start fixing these bottlenecks. In other words, Performance Tuning is the modification of the system to perform better and faster.

Performance Tuning is primarily the rewriting of application code, but can also involve load balancing and distributed computing to caching strategies

## Performance Engineering: A Cultural Shift That Goes Beyond Performance Testing and Tuning

However, Performance Engineering is a cultural shift that goes beyond performance testing and performance tuning. Here are additional aspects of Performance Engineering and best practices you must adopt to integrate the performance mindset into every facet of your software development life cycle, from beginning to end:

* **Integrate Performance Into Your User Stories –**Performance cannot be an afterthought today. With Performance Engineering, you can bake in the required metrics and acceptance criteria early in the development process, right from the initial design. Also, these requirements need to be integrated into user stories and to your Definition of Done (DoD).

<https://www.blazemeter.com/blog/performance-engineering>

## What is Performance Engineering?

Performance engineering is a systematic approach for developing software applications to ensure they meet the expected performance objectives. It is a discipline that is focused on the architectural design, coding and implementation choices that engineers make, including their technologies, practices, processes and frameworks.

<https://www.blazemeter.com/blog/performance-engineering>

Here are examples of free resources you can monitor:

* OS: CPU average idle
* Web server: Waiting requests
* App server: Free worker threads
* Messaging: Enqueue/dequeue wait time
* Database: Free connections in thread pool

# https://www.perfmatrix.com/performance-engineering-tutorial/**What is Performance Engineering?**

August 1, 2023 by [PerfMatrix](https://www.perfmatrix.com/author/gd-performance-tester/" \o "View all posts by PerfMatrix)

Software Performance Engineering is the systematic approach to constructing software that meets performance objectives. It is more related to building a tuned software application at the early stage of the software development life cycle whose performance satisfies all the non-functional requirements during the performance testing phase. On the other hand, [Software Performance Testing](http://perfmatrix.blogspot.in/2016/09/basic-of-performance-testing.html) is the act of evaluating the software system for its performance and finding the bottlenecks in the system. It is more related to quality assurance.

**Software Performance Testing:** To test and certify the quality of the application with respect to its performance.

**Software Performance Engineering:** To plan and build quality software without or minimum performance bottlenecks.

Business requirements drive performance engineering. It is a methodology to optimize the performance of the application from the earliest design stages. The involvement of a Performance Engineer in the initial phase reduces the last-minute application tuning activities. In short, the main aim of Performance Engineering is to provide better business value for the organization by discovering potential issues early in the development cycle to reduce the cost of fixing the bottlenecks in the later stage.

## Need for Performance Engineering:

Performance Engineering reduces the need to rework and refactor the application in later development cycles and it results in an application that performs better precisely because the performance was an early consideration and an integral part of the design.

In the classical software development approach, performance testing is the last or second last phase before Go-Live. When the development and functional testing phases cross the provided timelines, the deadline pressure comes on PT. At that time, the project either skips performance testing or conducts PT in a hurry without fixing of bottlenecks; results; or performance issues in production. On the other hand, the project may need to pay a fine if they conduct performance testing and bottleneck fixing by postponing the Go-Live date. Both cases break the trust of the client. What happens if somebody (Performance Engineer) focuses on design at the early stage so that minimum or no performance issues are identified at the later stage without missing the timelines? To meet this expectation a “Software Performance Engineering” has to be involved from the day first of software development.

* When a bottleneck is identified during performance testing then the role of the performance tester is to analyse the test result and raise a defect. On the other hand, the job of a performance engineer is to investigate the root cause and propose the solution to resolve the bottleneck.

At last, I would say a performance tester with good performance testing skill become a good performance engineer if he has an interest in bottleneck identification, root cause analysis and application tuning. But to suit this role, he must have good knowledge of the application design and architecture.

<https://www.testingxperts.com/blog/performance-testing-metrics>

Performance testing is a non-functional software testing method used to check the speed, scalability, reliability, responsiveness, and performance of an app/website. Various performance testing methods include a spike, volume, endurance, stress, load, etc

<https://www.testingxperts.com/blog/performance-testing-metrics>

<https://www.testingxperts.com/blog/performance-testing-metrics>

### **CPU utilization:**

It is the percentage of CPU capacity utilized in processing the requests.

### **Memory utilization:**

This metric measures the utilization of the primary memory of the computer while processing any work requests.

### **Response times:**

It is the total time between sending the request and receiving the response. Better the response time, better the performance of website/application.

<https://www.testingxperts.com/blog/performance-testing-guide/#What-are-the-Different-Types-of-Performance-Testing>?

\*\*\*

<https://www.linkedin.com/advice/0/what-key-performance-indicators-kpis-metrics-measuring-1e>

Response time measures the time it takes for a system to process a request and return a response to the user. Throughput measures the amount of data or transactions that a system can handle per unit of time.

esource utilization measures the percentage of resources that a system consumes or allocates during performance testing. Error rate measures the percentage of requests or transactions that result in errors or failures during performance testing

<https://www.blazemeter.com/blog/performance-testing-metrics>

<https://dzone.com/articles/client-side-performance-testing>

To provide better feedback, performance tests are divided into two types: server-side and client-side. When you are testing **server-side code**, you are testing the logic and readiness of your application. When you are testing the **client-side**, you are doing end-to-end testing.

<https://www.opentext.com/what-is/performance-engineering>

<https://www.opentext.com/what-is/performance-testing>

Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload. It’s a key step in ensuring software quality, but unfortunately, is often seen as an afterthought, in isolation, and to begin once functional testing is completed, and in most cases, after the code is ready to release.

The goals of [performance testing](https://www.microfocus.com/solutions/performance-testing) include evaluating application output, processing speed, data transfer velocity, network bandwidth usage, maximum concurrent users, memory utilization, workload efficiency, and command response times.

**Identify the test environment and tools**

Identify the production environment, testing environment, and testing tools at your disposal. Document the hardware, software, infrastructure specifications, and configurations in both test and production environments to ensure coherence. Some performance testing may occur in the production environment but there must be rigorous safeguards that prevent the testing from disrupting production operations.

### Tips for performance testing

Create a testing environment that mirrors the production ecosystem as closely as possible.

Separate the performance testing environment from the UAT environment

### What is the difference between performance testing vs. performance engineering?

Performance testing and performance engineering are two closely related yet distinct terms. Performance Testing is a subset of Performance Engineering, and is primarily concerned with gauging the current performance of an application under certain loads.

To meet the demands of rapid[application delivery](https://www.microfocus.com/solutions/application-delivery), modern software teams need a more evolved approach that goes beyond traditional performance testing and includes end-to-end, integrated performance engineering. Performance engineering is the testing and tuning of software in order to attain a defined performance goal. Performance engineering occurs much earlier in the software development process and seeks to proactively prevent performance problems from the get-go.

<https://www.headspin.io/blog/client-side-performance-testing-metrics-to-consider>

he users are really concerned about responsive time of the websites. If any page takes more than 3 sec to load, it is regarded as slow and leads to a large impact on business.

<https://q-automations.com/client-side-performance/#google_vignette>

The client side relates to performance as seen within the web browser. This includes initial page load time, downloading all of the resources, JavaScript that runs in the browser, and more.

The server side relates to how long it takes to run on the server to execute requests. Optimizing the performance on the server generally revolves around optimizing things like database queries and other application dependencies.

<https://q-automations.com/client-side-performance/>

Client side metrics

## Page Load Time

This metric indicates when the page was completely loaded and started to be interactive. It is fired when a resource and its dependent resources have finished loading.