**Тестирование производительности программного обеспечения**

**Тестирование производительности** в [инженерии программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) — [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой [нагрузкой](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0). Также может служить для проверки и подтверждения других атрибутов качества системы, таких как [масштабируемость](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C), [надёжность](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B4%D1%91%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и потребление ресурсов.

Тестирование производительность - стремится учитывать производительность на стадии моделирования и проектирования системы, перед началом основной стадии кодирования.

В тестировании производительности различают следующие направления:

* Нагрузочное (Load)
* Стресс (stress)
* Тестирование стабильности
* Конфигурационное (configuration)

Возможны два подхода к тестированию производительности программного обеспечения[[1]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8#cite_note-_fa394dea5f0029c6-1):

* В терминах рабочей нагрузки: программное обеспечение подвергается тестированию в ситуациях, соответствующих различным [сценариям использования](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B9_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F);
* В рамках [бета-тестирования](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), когда система испытывается реальными конечными пользователями.

**Нагрузочное тестирование**

[Нагрузочное тестирование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — это простейшая форма тестирования производительности. Нагрузочное тестирование обычно проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой. Этой нагрузкой может быть, например, ожидаемое количество одновременно работающих пользователей приложения, совершающих заданное число транзакций за интервал времени. Такой тип тестирования обычно позволяет получить время отклика всех самых важных бизнес-транзакций. В случае наблюдения за базой данных, [сервером приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9), сетью и т. д., этот тип тестирования может также идентифицировать некоторые узкие места приложения.

**Стресс-тестирование**

Стресс-тестирование обычно используется для понимания пределов пропускной способности приложения. Этот тип тестирования проводится для определения надёжности системы во время экстремальных или диспропорциональных нагрузок и отвечает на вопросы о достаточной производительности системы в случае, если текущая нагрузка сильно превысит ожидаемый максимум.

**Тестирование стабильности**

Тестирование стабильности проводится с целью убедиться в том, что приложение выдерживает ожидаемую нагрузку в течение длительного времени. При проведении этого вида тестирования осуществляется наблюдение за потреблением приложением памяти, чтобы выявить потенциальные утечки. Кроме того, такое тестирование выявляет деградацию производительности, выражающуюся в снижении скорости обработки информации и/или увеличении времени ответа приложения после продолжительной работы по сравнению с началом теста.

**Конфигурационное тестирование**

Конфигурационное тестирование — ещё один из видов традиционного тестирования производительности. В этом случае вместо того, чтобы тестировать производительность системы с точки зрения подаваемой нагрузки, тестируется эффект влияния на производительность изменений в конфигурации. Хорошим примером такого тестирования могут быть эксперименты с различными методами балансировки нагрузки. Конфигурационное тестирование также может быть совмещено с нагрузочным, стресс или тестированием стабильности.

**Определение целей тестирования производительности**

Многие тесты на производительность делаются без попытки осмыслить их реальные цели. Перед началом тестирования всегда должен быть задан бизнес-вопрос: «Какую цель мы преследуем, тестируя производительность?». Ответы на этот вопрос являются частью [технико-экономического обоснования](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%BE-%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) (или [business case](https://en.wikipedia.org/wiki/en:business_case" \o "w:en:business case)) тестирования. Цели могут различаться в зависимости от технологий, используемых приложением, или его назначения, однако, они всегда включают что-то из нижеследующего:

**Параллелизм / Пропускная способность**

Если конечными пользователями приложения считаются пользователи, выполняющие логин в систему в любой форме, то в этом случае крайне желательно достижение параллелизма. По определению это максимальное число параллельных работающих пользователей приложения, поддержка которого ожидается от приложения в любой момент времени. Модель поведения пользователя может значительно влиять на способность приложения к параллельной обработке запросов, особенно если он включает в себя периодически вход и выход из системы.

**Время ответа сервера**

Эта концепция строится вокруг времени ответа одного узла приложения на запрос, посланный другим.

**Время отображения**

Время отображения — одно из самых сложных для приложения для нагрузочного тестирования понятий, так как в общем случае они не используют концепцию работы с тем, что происходит на отдельных узлах системы, ограничиваясь только распознаванием периода времени в течение которого нет сетевой активности. Для того, чтобы замерить время отображения, в общем случае требуется включать функциональные тестовые сценарии в тесты производительности, но большинство приложений для тестирования производительности не включают в себя такую возможность.

**Требования к производительности**

Очень важно детализировать требования к производительности и документировать их в каком-либо плане тестирования производительности.

**Тестирование производительности** может проводиться с использованием глобальной сети и даже в географически удаленных местах, если учитывать тот факт, что скорость работы сети Интернет зависит от местоположения. Оно также может проводиться и локально, но в этом случае необходимо настроить сетевые маршрутизаторы таким образом, чтобы появилась задержка, присутствующая во всех публичных сетях. Нагрузка, прилагаемая к системе, должна совпадать с реальным положением дел. Так например, если 50 % пользователей системы для доступа к системе используют сетевой канал шириной 56К, а другая половина использует оптический канал, то компьютеры, создающие тестовую нагрузку на систему должны использовать те же соединения (идеальный вариант) или эмулировать задержки вышеуказанных сетевых соединений, следуя заданным профайлам пользователей.

**Основные показатели производительности**

**Потребление ресурсов центрального процессора (CPU, %)**

Метрика, показывающая сколько времени из заданного определённого интервала было потрачено процессором на вычисления для выбранного процесса.

**Потребление оперативной памяти (Memory usage, Mb)**

Метрика, показывающая количество памяти, использованной приложением.

**Потребление сетевых ресурсов**

Эта метрика не связана непосредственно с производительностью приложения, однако её показатели могут указывать на пределы производительности системы в целом.

**Работа с дисковой подсистемой (I/O Wait)**

Работа с дисковой подсистемой может значительно влиять на производительность системы, поэтому сбор статистики по работе с диском может помогать выявлять узкие места в этой области. Большое количество чтений или записей может приводить к простаиванию процессора в ожидании обработки данных с диска и в итоге увеличению потребления CPU и увеличению времени отклика.

**Время выполнения запроса (request response time, ms)**

Время выполнения запроса приложением остаётся одним из самых главных показателей производительности системы или приложения. Это время может быть измерено на серверной стороне, как показатель времени, которое требуется серверной части для обработки запроса; так и на клиентской, как показатель полного времени, которое требуется на [сериализацию / десериализацию](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F" \o "Сериализация), пересылку и обработку запроса. Надо заметить, что не каждое приложение для тестирования производительности может измерить оба этих времени.