Лекция 12. Тестирование производительности

9 этапов тестирования производительности

Тестирование производительности проводится для обеспечения бесперебойной работы и снижения стоимости владения ПО на всех этапах жизненного цикла решения.

В этой лекции будет описано в какой последовательности проводится данная проверка и какие особенности характерны для каждого из 9 главных этапов:

1. анализ системы и подбор требований;
2. подготовка стратегии;
3. настройка генератора нагрузки;
4. проведение мониторинга серверов и генератора нагрузки;
5. подготовка тестовых данных;
6. разработка нагрузочных скриптов;
7. предварительные запуски тестов;
8. проведение тестирования;
9. анализ результатов и подготовка отчёта.

ЭТАП 1 – АНАЛИЗ СИСТЕМЫ И ПОДБОР ТРЕБОВАНИЙ

Важно, чтобы перед проведением тестирования система была закончена. Если процесс разработки не завершен, то полученные данные отразят недействительные результаты.

Также нужно убедиться, что конфигурации ПО настроены правильно, поэтому после установки приложения обязательно проводится функциональная проверка.

Анализ системы включает изучение ее свойств, особенностей и режима работы. С его помощью инженеры:

- воспроизводят максимально объективные шаблоны поведения пользователей и профиля нагрузки;
- определяют необходимое количество тестовых данных;
- выявляют потенциально «узкие» места ПО;
- настраивают способы мониторинга системы.

Требования – это критерии, которым система соответствует с технической точки зрения.

Основной упор делается на определение критериев успешности проведенных тестов, которые содержатся в SLA (service-level agreement).

Подобранные на начальном этапе требования будут сравниваться с полученными результатами для того, чтобы оценить поведение программного продукта как в целом, так и отдельными блоками, а также найти «узкие» места.

Такими *критериями* будут следующие измеряемые **метрики:**

Метрика	**Определение**
Количество пользователей	Программа, которая при выполнении запросов к приложению действует как реальный пользователь.
Время отклика	Время, затраченное системой на выполнение запроса пользователя.
Запросы в секунду	Измерение количества запросов, отправляемых на целевой сервер.
Транзакции в секунду	Измерение количества транзакций, отправляемых на целевой сервер.
Процент ошибок	Процент ошибок от общего числа ответов в единицу времени.
Процессор	Процентное соотношение времени, которое процессор тратит на выполнение рабочего потока.
Оперативная память	Количество МВ в физической памяти, свободное для всех программ и процессов.
Жесткий диск	Информация о процентах использования дискового пространства.

В некоторых случаях анализ системы и подбор требований будут проходить на протяжении всего жизненного цикла проекта. Если приложение уже находится в эксплуатации, то анализ целевой аудитории и наблюдение за их действиями позволит определить, какие части приложения наиболее критичны, а также как будет происходить работа с продуктом.

ЭТАП 2 – ПОДГОТОВКА СТРАТЕГИИ

Стратегия разрабатывается на основе детального *анализа* ПО и описывает расширенный подход к тестированию производительности.

Анализ включает всю собранную информацию о системе, целях тестирования, требованиях к ПО, *конфигурации тестового стенда*, мониторинге, сценариях поведения пользователей, *профиле и модели нагрузки*, *инструментах тестирования производительности*, планируемых запусках тестов, виде предоставления результатов**.**

Конфигурация тестового стенда

На результаты нагрузочного тестирования влияют такие факторы, как конфигурация тестового стенда, загруженность сети, наполненность БД и др.

Поэтому для получения достоверных результатов проверка производительности проводится на отдельном окружении с параметрами и конфигурацией, которые максимально приближены к характеристикам реальной системы.

Части тестового стенда	**Характеристики**
Аппаратное обеспечение	• центральный процессор (количество ядер, тип, частота); • оперативная память (объём, тип); • жёсткий диск (тип, скорость).

Программное обеспечение	• ОС (версия и пакеты обновления); • сервер приложения (версия и патчи); • СУБД (версия и тип).
Сетевое оборудование	• топология сети; • пропускная способность; • протокол передачи данных.
Приложение	• архитектура; • БД (структура, данные); • ПО, необходимое для работы приложения.

Разработка профиля и модели нагрузки

Во время проведения тестирования производительности собирается статистика по использованию ПО, которая нужна для создания **профиля нагрузки** – модели поведения пользователей.

Профиль отражает процентное распределение совершаемых в системе операций между пользователями разных ролей. После этого вычисляется начальная точка и размер шага для увеличения интенсивности выполненных операций.

Модель нагрузки для одного и того же теста бывает быть разной. Например, можно использовать модель нагрузки «+1 пользователь каждые 5 секунд» либо добавить всех пользователей сразу.

Выбор инструмента для нагрузочного тестирования

Использование инструментов нагрузочного тестирования позволяет записать процесс обмена данными между сервером и браузером на уровне HTTP(S), WS, TCP и др. и воспроизвести этот процесс с учётом логики корреляции динамических параметров.

Запуск разработанных скриптов в несколько потоков помогает сгенерировать необходимую нагрузку на систему.

Для тестирования производительности используются Apache JMeter, HP Loadrunner, Microsoft Visual Studio, TSUNG.

Остановимся на этих инструментах подробнее:

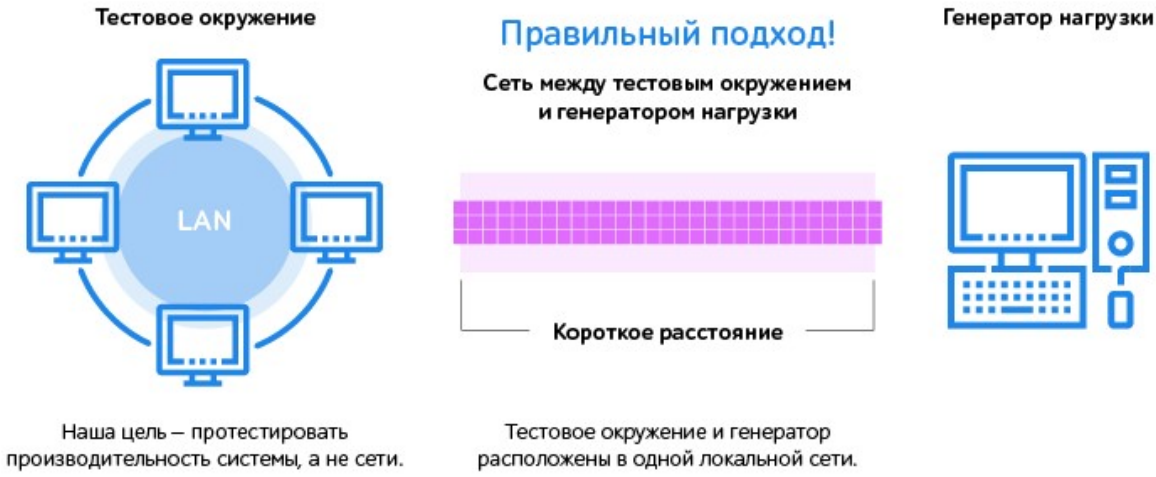
Инструмент	Преимущества	Недостатки
	• Проведение тестов со сложной логикой и корреляцией динамических параметров. • Тестирование производительности веб-приложений, включая API, веб-службы и подключение к БД. • Возможность воспроизведения поведения нескольких пользователей с параллельными потоками.	• Проблематичное воспроизведение трафика AJAX. • Ограниченные возможности создания отчётов. • Ресурсоёмкость при запуске тестов (требуется много оперативной памяти и ЦПУ).
	• Распределенная архитектура (VuGen, Controller, Load Generator, Analysis). • Наличие набора инструментов для работы с различными протоколами. • Генерация отчёта и создание шаблона для отчёта. • Подробное логирование действий каждого виртуального пользователя.	• Лицензионное ограничение количества виртуальных пользователей. • Отсутствие частых обновлений.
	• Хранение результатов тестов в БД MS SQL Server и их удобная структуризация. • Отсутствие ограничений по количеству виртуальных пользователей при наличии лицензии.	• Ограниченные возможности декларативных тестов для реализации логики в скриптах. • Доступность только декларативных тестов для SharePoint.
	• Поддержка HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP, XMPP. • Удобство в тестировании API-запросов. • Распределение нагрузки на кластере из клиентских машин.	• Отсутствие GUI. • Невозможность автоматически подгружать подзапросы, которые содержатся на странице HTML. • Сложности в работе с WS, если необходимо получить ответ и использовать его в дальнейшем.

		• Недостаточно подробные отчёты.

ЭТАП 3 – НАСТРОЙКА ГЕНЕРАТОРА НАГРУЗКИ

Для качественного проведения тестирования производительности инструмент устанавливается на **генератор нагрузки** – виртуальную или физическую машину, расположенную максимально близко к серверу(-ам) приложения. Это снижает искажения при подаче нагрузки, вызванные задержками сети.

Для создания большого объёма нагрузки ресурсов одной машины может быть недостаточно, поэтому проводится **распределённое нагрузочное тестирование**.



ЭТАП 4 – ПРОВЕДЕНИЕ МОНИТОРИНГА СЕРВЕРОВ И ГЕНЕРАТОРА НАГРУЗКИ

Выделенные на первом этапе метрики используются для оценки параметров производительности ПО и определения тех областей приложения, которые создают «узкие» места.

Информация, полученная при мониторинге всех серверов и ресурсов системы, используется при анализе результатов тестирования производительности.

Мониторинг можно проводить с помощью специальных **утилит** для отслеживания потребления аппаратных ресурсов ПО.

Выделим наиболее эффективные утилиты:

- **Nmon**

Инструмент сохраняет и отображает краткую информацию о памяти, жестком диске, сети, файловых системах и не сильно загружает процессор (1-2%).

Nmon используется в двух режимах:

1. Онлайн-режим. После запуска nmon на экране командной строки раз в 2 секунды будет выводиться информация о мониторинге в реальном времени.
2. Режим сохранения информации. Данные за время мониторинга сохраняются в CSV-файле, который используется как набор данных для построения графиков и анализа результатов вручную.

- **Performance Monitor**

Для запуска достаточно открыть Performance Monitor на машине и создать новый сборщик информации в Data Collector Set.

После того как мониторинг всех серверов и ресурсов закончен, создаётся BLG-файл. Он преобразуется в CSV-файл, с помощью которого анализируются результаты.

Для упрощения построения графиков также используется специальный инструмент PAL (Performance Analysis of Logs) – он позволяет структурировать информацию и автоматически строит графики.

- **Zabbix**

Данное программное обеспечение мониторинга работоспособности и состояния сервера строится по принципу «сервер-агент». Сервер собирает всю информацию, поэтому можно просматривать историю мониторинга, настраивать необходимые метрики и добавлять правила.

- **Grafana**

ПО способно анализировать информацию из БД временных рядов для Graphite, InfluxDB и OpenTSDB с мощными функциями визуализации.

- **New Relic**

Используя этот сервис отслеживания производительности, можно легко просматривать и анализировать огромные объёмы данных и получать информацию о действиях в реальном времени.

Профилирование БД

Для того чтобы проверить, какие запросы, хранимые процедуры и триггеры снижают быстроту работы БД, проводится профилирование. Оно позволяет быстро выполнить диагностику и не тратить время на поиск ошибок, а сразу приступить к их исправлению.

- **PostgreSQL** — свободная объектно-реляционная система управления базами данных. Для целей тестирования производительности подходит профайлер pgBadger.

pgBadger — это анализатор логов Postgre SQL, который предоставляет общую статистику по запросам, информацию по медленным запросам, данные о соединениях и сессиях и т. д.

- **SQL Server Profiler** — инструмент для отслеживания, воссоздания и устранения неполадок MS SQL Server. Он используется для создания и обработки запросов, воспроизведения и анализа результатов их выполнения.

Профилирование веб-серверов

Все системы уникальные, а использование профилирования помогает понять, где стоит увеличить производительность.

JProfiler объединяет профилирование ЦПУ, памяти и потоков, поэтому очень легко узнать, что нужно оптимизировать, устранить или изменить. Этот инструмент может использоваться как для локальных, так и для удаленных сеансов.

Профилирование PHP

Профилирование позволяет выявить участки кода, которые замедляют работу.

- **Xdebug** – это мощный инструмент, который используется для анализа PHP-кода и определения узких мест.
- **XHProf** работает по принципу декомпозиции системы на вызовы функций (методов) и построения статистики в разрезе их ресурсопотребления. В результате можно получить информацию о количестве выделяемой памяти, вызовов функций, времени исполнения и т. д.

ЭТАП 5 – ПОДГОТОВКА ТЕСТОВЫХ ДАННЫХ

Универсального подхода к такой подготовке нет. Часто для корректной работы решения нужно создавать данные в объёме, достаточном для разработки скриптов и проведения тестов.

Как правило, подходящий метод устанавливается на этапе анализа особенностей ПО, когда совместно со стороной разработки обсуждаются типы тестовых данных и их структура.

Выделим 4 способа подготовки данных:

- Код

С помощью скриптов на различных языках программирования (Java, Python и др.) можно создавать пользователей, пароли для пользователей и другие необходимые значения для корректного использования данных. Акцент делается на уникальности имен и их паролях, которые отвечают установленным критериям надежности.

- SQL-запросы

Один из способов заполнения базы данных – SQL-запросы. Такой вариант подходит только в случае, если есть доступ к БД на стороне сервера.

- API-запросы

Если необходимо наполнить базу товарами, информацией о пользователях, то используются API-запросы.

- Интерфейс

Можно создать скрипт, который будет проходить шаги реального пользователя при регистрации. В таком случае есть смысл использовать встроенные функции нагрузочных инструментов, позволяющие провести рандомизацию значений по конкретному алгоритму.

Во время выполнения скрипта пользователи записываются в БД, после чего делается снимок файловой системы для дальнейшего использования полученной информации при запусках тестов.

ЭТАП 6 – РАЗРАБОТКА НАГРУЗОЧНЫХ СКРИПТОВ

Сценарии тестирования производительности создаются с использованием выбранного инструмента.

Процесс разработки скриптов состоит из трех шагов:

1. Изучение сценариев

Необходимо проанализировать подготовленные сценарии, определить, что должен делать каждый из них и каким образом. После того как каждый из шагов будет проверен вручную и никаких ошибок в процессе выполнения не возникнет, можно приступать к записи скрипта.

2. Создание тест-кейсов

Для каждого из сценариев записывается шаблон, в котором нет параметризации, и проводится проверка. После проверки каждый из запросов параметризируется для того, чтобы для разных пользователей один и тот же сценарий отрабатывал корректно.

3. Отладка сценариев

На этой стадии проверяется запуск каждого сценария по X-итерациям для разных пользователей. Если в скрипте каждый тест-кейс отрабатывает корректно, то последний шаг завершается.

ЭТАП 7 – ПРЕДВАРИТЕЛЬНЫЕ ЗАПУСКИ ТЕСТОВ

Предварительные запуски тестов нужны для проверки корректности работы подготовленных нагрузочных скриптов, а также для поиска оптимальной модели нагрузки на систему. Они позволяют определить утилизацию ресурсов генератора(-ов) нагрузки и понять, хватит ли заложенных мощностей генератора(-ов) для проведения полноценных тестов.

Запускать тесты следует с разной моделью нагрузки. После каждого запуска необходимо проанализировать результаты и получить оптимальную нагрузку, которую может выдержать решение.

Таким образом, путем анализа результатов теста и мониторинга ПО устанавливается нагрузка, которая в дальнейшем будет использоваться для проведения тестирования.

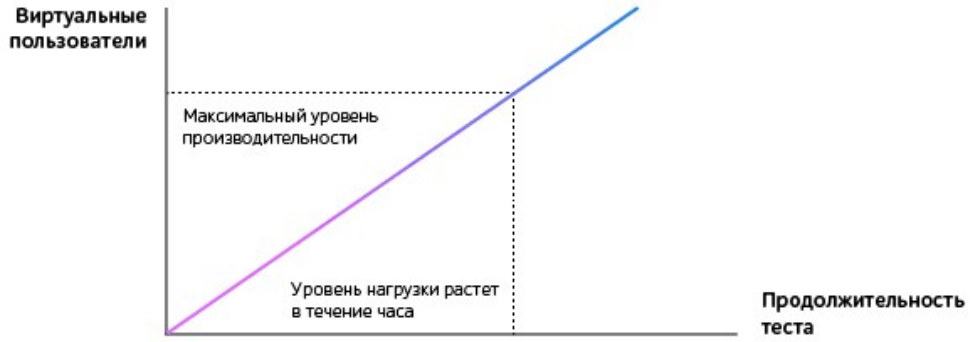
ЭТАП 8 – ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ

При тестировании производительности выполняются следующие виды проверок:

1. Стресс-тест проходит с постепенно увеличивающейся на сервер нагрузкой и возрастает до тех пор, пока не будет достигнут один из критериев остановки теста:

- превышение требуемых значений отклика в несколько раз;
- достижение критического уровня использования аппаратных ресурсов (ЦП>80%, память>90%);
- количество ошибок HTTP превышает 1% от общего числа запросов;
- сбой системного ПО.

Стресс-тест позволяет оценить поведение программного продукта при повышенной интенсивности выполнения операций и установить предельный уровень производительности решения.



2. Нагрузочное тестирование проводится в течение длительного промежутка времени (4-8 ч.).

Если в результате стресс-теста система не выдержала целевую нагрузку, то проверка проходит под нагрузкой 80% от результата максимальной производительности, полученной при проведении стресс-теста.

3. Проверка стабильности проводится с ожидаемым уровнем нагрузки при длительном многочасовом тестировании. При достижении максимального количества пользователей нагрузка на сервер больше не подаётся и остаётся постоянной на протяжении всей проверки. Тестирование стабильности может продолжаться несколько дней.

4. Объемное тестирование используется для оценки производительности работы ПО при увеличении данных, которые хранятся в БД приложения. Для проведения такого теста предварительно необходимо заполнить базу необходимым объёмом информации.

5. Проверка масштабируемости позволяет оценить способность решения увеличивать производительность пропорционально добавлению аппаратных ресурсов системы.

6. При модульном тестировании нагрузка подаётся на отдельные компоненты ПО.

7. Тест «часа пик» проводится для того, чтобы проверить реакцию системы как в момент наибольшей загруженности, так и непосредственно после снижения нагрузки.

8. Тестирование конфигурации оценивает работу программного продукта при разных конфигурациях окружения.

ЭТАП 9 – АНАЛИЗ РЕЗУЛЬТАТОВ И ПОДГОТОВКА ОТЧЁТА

На основе предыдущих этапов создаётся документ, который описывает не только результаты проведенных тестов, но и ход выполнения каждого из них.

Отчёт по тестированию содержит следующую информацию:

- цель проведения тестирования;
- конфигурацию тестового стенда и генератора нагрузки;
- требования к ПО;
- сценарии поведения пользователей, профиль нагрузки;

- статистику по ключевым характеристикам производительности (время отклика, количество запросов в секунду, количество транзакций в секунду);
- данные о максимально возможном количестве одновременно работающих пользователей, при котором решение будет справляться с нагрузкой;
- сведения о количестве и типах ошибок HTTP;
- графики, показывающие зависимость производительности системы от количества одновременно работающих пользователей;
- выводы о производительности системы в целом и о слабых местах, если они будут обнаружены;
- рекомендации по улучшению производительности ПО.