* **[Лабораторная работа №9. Нагрузочное тестирование web-сервера.](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html)**
  + [**Основные теоретические сведения**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id1)
    - [**Термины и определения**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id2)
    - [**Цели нагрузочного тестирования**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id3)
    - [**Виды нагрузочного теста**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id4)
    - [**Использование Apache benchmark tool**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#apache-benchmark-tool)
    - [**Использование httperf**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#httperf)
    - [**Использование Siege**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#siege)
    - [**Балансировщик нагрузки Nginx**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#nginx)
  + [**Задания к лабораторной работе**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id5)
    - [**Нагрузочное тестирование веб-сервера с Apache**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#apache)
    - [**Нагрузочное тестирование веб-сервера с Nginx.**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id6)
    - [**Нагрузочное тестирование веб-серверов Apache с балансировщиком нагрузки.**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id7)
    - [**Нагрузочное тестирование веб-серверов Nginx с балансировщиком нагрузки.**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id8)
  + [**Вопросы к лабораторной работе**](https://xn--80aqobguv5e.xn--p1ai/%D0%BE%D0%B8%D0%B1/lr9.html#id9)

**Лабораторная работа №9. Нагрузочное тестирование web-сервера.**

**Основные теоретические сведения**

**Цель:** С помощью систем нагрузочного тестирования определить производительность web-серверов Apache и Nginx, добиться отказа в обслуживании.

**Термины и определения**

**Нагрузочное тестирование** - это автоматизированный процесс, имитирующий одновременную работу определенного количества пользователей на каком-либо общем ресурсе.

**Приложение** - тестируемое прикладное программное обеспечение.

**Виртуальный пользователь** - программный процесс, который циклически выполняет моделируемые операции.

**Итерация** - один повтор в цикле операции.

**Интенсивность выполнения операции** - частота выполнения операций в единицу времени, в тестовых скриптах задается интервалом времени между итерациями.

**Нагрузка** - совокупное количество попыток выполнить операции на общем ресурсе. Создается или пользовательской (клиентской) активностью или нагрузочными скриптами.

**Производительность** - количество выполняемых приложением операций в единицу времени.

**Масштабируемость приложения** - пропорциональный рост производительности при увеличении нагрузки.

**Профилем нагрузки** называется набор операций с заданными интенсивностями, полученными на основе статистики.

**Нагрузочной точкой** называется рассчитанное (либо заданное Заказчиком) количество виртуальных пользователей в группах, выполняющих операции с определенными интенсивностями.

**Тест производительности, бенчмарк (англ. benchmark)** — контрольная задача, необходимая для определения сравнительных характеристик производительности компьютерной системы.

Успешное прохождение ряда тестов является свидетельством стабильности системы в штатном и в разогнанном режимах.

**Цели нагрузочного тестирования**

1. Оценка работоспособности и производительности приложения на этапе разработки и при передаче в эксплуатацию
2. Оценка работоспособности и производительности приложения на этапе выпуска новых релизов, патч-сетов
3. Оптимизация производительности приложения, включая оптимизацию программного кода и настройку серверов
4. Подбор соответствующей для данного приложения аппаратной и программной платформы, а также нужной конфигурации сервера

**Виды нагрузочного теста**

**Нагрузочный (Load-testing)** – определяет работоспособность системы при некотором строго заданном уровне нагрузки (планируемой, рабочей).

**Устойчивости (Stress)** – используется для проверки параметров системы в экстремальных условиях и условиях сверхнагрузки, основная задача во время испытания - нарушить нормальную работу системы. Позволяет определить минимальные системные требования для работы приложения, оценить предельные возможности системы и факторы, которые ограничивают эти возможности. В рамках теста также определяется возможность системы сохранить целостность данных при возникновении внештатных аварийных ситуаций.

**Производительности (Performance)** – комплексный тест, включает в себя предыдущие два режима тестирования и предназначен для общей оценки всех показателей системы.

**Результат теста** – представляет собой максимально возможное число пользователей, которые могут единовременно получить доступ к веб-ресурсу, число запросов, которое в состоянии обработать приложение, или время ответа сервера. На основе этой информации, веб-мастер и сетевой администратор смогут заранее выявить слабые места, возникающие из-за несбалансированной работы компонентов (базы данных, маршрутизаторы, кэширующий и прокси-сервер, брандмауэры и др.), и исправить ситуацию, прежде чем система будет запущена в рабочем режиме.

**Использование Apache benchmark tool**

**Apache benchmark** — одна из самых простых утилит, которая применяется для нагрузочного тестирования сайта. Идет в комплекте с веб сервером Apache, в первоначальной настройке не нуждается. Задача, которая ставится перед Apache benchmark — показать, какое количество запросов сможет выдержать веб сервер и как быстро он их обработает.

Пример нагрузки на сервер в 5000 последовательных запросов:

<ab -n 5000 <http://192.168.1.116/index.html>>

Пример нагрузки на сервер в 5000 запросов, но 500 из них будут направлены на сервер одновременно (параллельные запросы):

<ab -n 5000 -c 500 <http://192.168.1.116/index.html>>

**Примечание**

Для выполнения лабораторной работы меняйте значения после -n и -c, чтобы узнать с каким количеством запросов может справиться сервер. На этих примерах выполнены HTML-запросы, для тестирования на PHP-запросы измените цель на index.php (В лабораторной работе №8 есть информация о PHP).

**Использование httperf**

Еще одно консольное приложение, используемое также для создания нужного количества параллельных запросов - httperf.

Его отличие от ab в том, что httperf посылает запросы согласно своим настройкам, невзирая на то, отвечает сервер на них или уже нет. Таким образом можно определить не только какую максимальную нагрузку может выдержать сервер, но и как будет себя вести сервер в момент, когда нагрузка достигла своего пика

Пример запуска 100 запросов от 10 посетителей параллельно:

httperf --port 80 --server <domain> --uri=/ --num-conns=100 --rate=10

**Использование Siege**

Установка:

sudo apt-get install siege

Количество запросов не лимитируется, но можно задавать время в течение которого выполнять тестирование. Пример: 5 пользователей, которые безостановочно загружают главную страницу в течении одной минуты.

siege -c 5 -b -t 1m ip-адрес

**Балансировщик нагрузки Nginx**

Балансировка нагрузки (англ. load balancing) — метод распределения заданий между несколькими сетевыми устройствами (например, серверами) с целью оптимизации использования ресурсов, сокращения времени обслуживания запросов, горизонтального масштабирования кластера (динамическое добавление/удаление устройств), а также обеспечения отказоустойчивости (резервирования).

Установка:

sudo apt-get install nginx

Настройка:

sudo nano /etc/nginx/sites-available/default

upstream web\_backend {

server 192.168.1.113;

server 192.168.1.114;

}

server {

listen 80;

location / {

proxy\_set\_header X­Forwarded­For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://web\_backend;

}

}

После настройки перезапускаем Nginx

sudo service nginx reload

Методы балансировки нагрузки (описываются в начале секции upstream):

* ip\_hash - согласно этому методу запросы от одного и того же клиента будут всегда отправляться на один и тот же backend сервер на основе информации об ip адресе клиента. Не совместим с параметром weight.
* least\_conn - запросы будут отправляться на сервер с наименьшим количеством активных соединений.
* round-robin - режим по умолчанию. То есть если вы не задали ни один из вышеупомянутых способов балансировки - запросы будут доставляться по очереди на все сервера в равной степени.

**Задания к лабораторной работе**

**Нагрузочное тестирование веб-сервера с Apache**

*Для тестирования используются 2 машины – одна с установленным и работающим Apache, вторая будет отсылать запросы и делать выводы о производительности web-сервера.*

Тестирование на PHP-запросы:

* Определить максимальное число параллельных запросов, при котором сервер нас не будет блокировать.
* Провести тест при использовании максимального числа запросов.

Тестирование на HTML-запросы:

* Определить максимальное число параллельных запросов
* Провести тест при использовании максимального числа запросов.

Провести сравнение результатов и сформировать выводы.

**Нагрузочное тестирование веб-сервера с Nginx.**

*Для тестирования используется 2 виртуальные машины – одна с установленным и работающим Nginx, которой будут отсылаться запросы, другая будет отсылать эти самые запросы и делать выводы о производительности веб-сервера с Nginx.*

**Примечание**

<sudo apt-get nginx> - Установка Nginx

Тестирование на PHP-запросы:

* Определить максимальное число параллельных запросов, при котором сервер нас не будет блокировать.
* Провести тест при использовании максимального числа запросов.
* Сравнить с результатами, полученными при тестировании Apache

Тестирование на HTML-запросы:

* Определить максимальное число параллельных запросов.
* Провести тест при использовании максимального числа запросов.
* Сравнить с результатами, полученными при тестировании Apache

Провести сравнение результатов и сформировать выводы.

**Нагрузочное тестирование веб-серверов Apache с балансировщиком нагрузки.**

*Для тестирования используется 4 машины – две одинаковые с установленным и работающим Apache в качестве веб-серверов, которые соединены с третьей машиной, которая выполняет роль балансировщика нагрузки, на нем работает Nginx, четвертая машина будет отсылать эти запросы серверу и делать выводы о производительности данной связки из балансировщика нагрузки на Nginx и двумя веб-серверами Apache.*

Тестирование на PHP-запросы:

* Провести тест при использовании максимального для Apache числа запросов
* Провести тест при использовании максимального для Nginx числа запросов
* Сравнить с предыдущими результатами и сформировать выводы

Тестирование на HTML-запросы:

* Провести тест при использовании максимального для Apache числа запросов
* Провести тест при использовании максимального для Nginx числа запросов
* Сравнить с предыдущими результатами и сформировать выводы

**Нагрузочное тестирование веб-серверов Nginx с балансировщиком нагрузки.**

*Для тестирования используется 4 виртуальные машины – две одинаковые с установленным и работающим Nginx в качестве веб- серверов, которые соединены с третьей машиной, которая выполняет роль балансировщика нагрузки, на нем работает Nginx, четвертая машина будет отсылать эти запросы серверу и делать выводы о производительности данной связки из балансировщика нагрузки на Nginx и двумя веб-серверами Nginx.*

Тестирование на PHP-запросы:

* Провести тест при использовании максимального для Apache числа запросов
* Провести тест при использовании максимального для Nginx числа запросов
* Сравнить с предыдущими результатами и сформировать выводы

Тестирование на HTML-запросы:

* Провести тест при использовании максимального для Apache числа запросов
* Провести тест при использовании максимального для Nginx числа запросов
* Сравнить с предыдущими результатами и сформировать выводы

| *Итоговая таблица сравнения* | | | | | |
| --- | --- | --- | --- | --- | --- |
|  |  | **Максимальное число запросов** | **Запросы/сек** | **Время, затрачиваемое на запрос, мс** | **% успешных запросов** |
| Apache | PHP |  |  |  |  |
|  | HTML |  |  |  |  |
| LB + Apache | PHP |  |  |  |  |
|  | HTML |  |  |  |  |
| Nginx | PHP |  |  |  |  |
|  | HTML |  |  |  |  |
| LB + Nginx | PHP |  |  |  |  |
|  | HTML |  |  |  |  |

**Вопросы к лабораторной работе**

1. Назначение нагрузочного тестирования?
2. Что такое нагрузка?
3. Как указать ab сделать нагрузку в 10000 запросов, 500 из которых будут направлены одновременно? Перестанет ли ваш сервер принимать входящие подключения?
4. Можно ли протестировать при помощи ab, httperf и Siege другие web-сервера? Назовите примеры.
5. Влияет ли использование скриптовых языков программирования (например, PHP) на производительность web-сервера? Объясните почему.
6. Для чего нужен балансировщик нагрузки?
7. Какие существуют методы балансировки нагрузки в nginx?

**Составьте отчет о выполнении лабораторной работы.**

**Включите в него копии экрана и ответы на вопросы лабораторной работы.**