Факультет компьютерных технологий и прикладной математики

Кафедра вычислительных технологий

02.03.02

Информационная безопасность

Лабораторная работа № 7

Тема: Нагрузочное тестирование web-сервера.

Цель работы

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

Указания к работе

Вначале студенты изучают теоретическую часть. Далее каждый студент должен выполнить задания, а также ответить на вопросы к лабораторной работе. За проделанную работу студент может получить оценку от «неудовлетворительно» до «отлично». Для получения оценки «удовлетворительно» студент должен выполнить ВСЕ задания к лабораторной работе. Оценка «хорошо» ставится, если студент ответил на ВСЕ вопросы к лабораторной работе. Оценку «отлично» студент получает, если подготовлен отчёт по лабораторной работе.

ОЦЕНКУ ЗА ПРОДЕЛАННУЮ РАБОТУ МОЖНО ПОВЫСИТЬ ДО СДАЧИ СЛЕДУЮЩЕЙ ЛАБОРАТОРНОЙ РАБОТЫ.

Теоретическая часть

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.

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.

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

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>

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.

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 X-Forwarded-For $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):

1. 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.

2. least_conn – запросы будут отправляться на сервер с наименьшим количеством активных соединений.

3. 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.

Nginx

Запустим тест, который мы запускали для Apache, чтобы можно было сравнить, какой из них работает лучше.

```
[root@lab ~]# ab -n 5000 -c 500 http://localhost:80/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking localhost (be patient)
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
Completed 3500 requests
Completed 4000 requests
Completed 4500 requests
Completed 5000 requests
Finished 5000 requests
Server Software:        nginx/1.10.1
Server Hostname:        localhost
Server Port:            80
Document Path:          /
Document Length:        3698 bytes
Concurrency Level:      500
Time taken for tests:   0.758 seconds
Complete requests:      5000
Failed requests:        0
Write errors:           0
Total transferred:      19660000 bytes
```

```
HTML transferred:       18490000 bytes
Requests per second:    6593.48 [#/sec] (mean)
Time per request:       75.832 [ms] (mean)
Time per request:       0.152 [ms] (mean, across all concurrent requests)
Transfer rate:          25317.93 [Kbytes/sec] received
Connection Times (ms)
min  mean[+/-sd] median   max
Connect:        0    6  11.0      2      53
Processing:     5   19   8.2     17      53
Waiting:        0   18   8.2     16      47
Total:         10   25  17.4     18      79
Percentage of the requests served within a certain time (ms)
50%     18
66%     21
75%     21
80%     22
90%     69
95%     73
98%     75
99%     76
00%     79 (longest request)
[root@lab ~]#
```

Видно, что Nginx обрабатывает **6593 запроса в секунду**.

Задания к лабораторной работе

Часть 1

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:

1) Определить максимальное число параллельных запросов, при котором сервер нас не будет блокировать.

2) Провести тест при использовании максимального числа запросов.

Тестирование на HTML-запросы:

3) Определить максимальное число параллельных запросов

4) 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:

1. Определить максимальное число параллельных запросов, при котором сервер нас не будет блокировать.

2. Провести тест при использовании максимального числа запросов.

3. Сравнить с результатами, полученными при тестировании Apache.

Тестирование на HTML-запросы:

1) Определить максимальное число параллельных запросов.

2) Провести тест при использовании максимального числа запросов.

3) 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:

1. Провести тест при использовании максимального для Apache числа запросов.

2. Провести тест при использовании максимального для Nginx числа запросов.

3. Сравнить с предыдущими результатами и сформировать выводы.

Тестирование на HTML-запросы:

1. Провести тест при использовании максимального для Apache числа запросов.

2. Провести тест при использовании максимального для Nginx числа запросов.

3. 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:

1. Провести тест при использовании максимального для Apache числа запросов.

2. Провести тест при использовании максимального для Nginx числа запросов.

3. Сравнить с предыдущими результатами и сформировать выводы.

Тестирование на HTML-запросы:

1. Провести тест при использовании максимального для Apache числа запросов.

2. Провести тест при использовании максимального для Nginx числа запросов.

3. Сравнить с предыдущими результатами и сформировать выводы.

		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?