

Tema 5

Medir prestaciones



José Manuel Soto Hidalgo
Dpto. Arquitectura y Tecnología de Computadores
Universidad de Granada

jmsoto@ugr.es

Índice



- [1. Introducción]
- 2. Conexiones por segundo
- 3. Número de conexiones concurrentes
- 4. Rendimiento, en bits por segundo
- 5. Tipos de tráfico
- 6. Límite en las prestaciones
- 7. Software
- 8. Apache benchmark
- 9. httpperf
- 10. OpenWebLoad
- 11. Siege

Introducción

Medir las prestaciones de nuestro sistema web:

- los servidores finales
- los dispositivos de balanceo

Objetivo: Comprobar si cumplen unos mínimos requisitos de rendimiento.

Aplicar una metodología de test de prestaciones para detectar posibles problemas de rendimiento.

Introducción

Principal necesidad de hacer los tests:

- No son exclusivamente las caídas o errores de programación factores que influyen en el rendimiento.
- Detectar posibles cuellos de botella e ineficiencias.
- Detectar límite del sistema

Limitaciones de los tests:

- Dificultad para hacer pruebas en un entorno de producción.
- No se puede simular el comportamiento de los usuarios.

Introducción

Muy importante **medir las prestaciones** de los dispositivos de balanceo.

Según el sitio web, usaremos **diferentes métricas**:

- conexiones por segundo
- número total de conexiones concurrentes
- rendimiento (en bits por segundo)

Índice



1. Introducción
- 2. Conexiones por segundo**
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Conexiones por segundo

Es una de las métricas más importantes cuando hablamos del rendimiento de servidores web.

Hace referencia al número de **conexiones de entrada** que cierto dispositivo puede manejar por segundo.

También se llama *transacciones por segundo* o *sesiones por segundo*.



Conexiones por segundo

Es un factor determinante, ya que **abrir y cerrar conexiones HTTP resulta muy costoso.**

En el nivel que estamos tratando, es la operación principal.

Para enviar datos hay que llevar a cabo una serie de pasos que pueden llegar a **sobrecargar el dispositivo de red.**

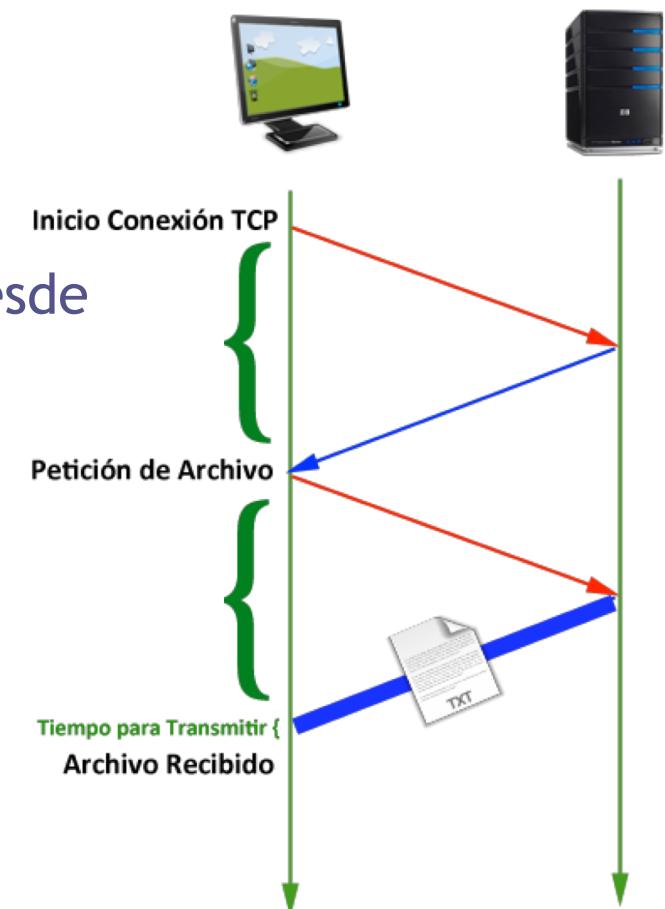
Las aperturas y cierres de **conexiones consumen muchos recursos.**

Conexiones por segundo

Pasos para establecer una conexión HTTP:

- el cliente inicia la conexión HTTP enviando un paquete TCP SYN al puerto 80 del servidor web
- el servidor web envía un paquete ACK al cliente seguido de otro SYN
- el cliente envía un paquete ACK como respuesta

Ahora ya pueden comenzar a enviarse datos desde el servidor al cliente (normalmente será una página web).



Conexiones por segundo

La velocidad a la que se gestionan las aperturas y cierres de conexiones es fundamental.

Si cierto servidor web tiene un tráfico HTTP alto,
conexiones por segundo será la métrica más importante.



Índice



1. Introducción
2. Conexiones por segundo
- [3. Número de conexiones concurrentes]**
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Número de conexiones concurrentes

Métrica para determinar **cuántas sesiones** de usuario TCP puede manejar el balanceador al **mismo tiempo**.

Limitado por la memoria o el procesador del dispositivo.

Varía desde varios miles hasta millones.
Límite teórico (realmente no es tan alta).



Número de conexiones concurrentes

Esto en cuanto a las conexiones TCP...

Sin embargo, para el tráfico UDP (streaming o tráfico DNS) el número de conexiones concurrentes no es un factor que afecte, ya que se trata de un protocolo "sin conexión":

- el receptor no reconoce haber recibido paquetes.

No hay una fase de establecimiento de la conexión.

No se mantiene información de estado.

TCP mantiene información sobre el estado de la conexión para garantizar un servicio fiable de transferencia de datos y control de congestión.

Número de conexiones concurrentes

Hoy en día, las webs de vídeos como Youtube o Vimeo, utilizan TCP ya que algunas organizaciones bloquean el tráfico UDP por cuestiones de seguridad.

También se usa TCP para no colapsar el servidor ya que TCP provee control de congestión.

Índice



1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. **Rendimiento, en bits por segundo**
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Rendimiento, en bits por segundo

Hace referencia a la **velocidad** a la que el balanceador maneja y pasa el tráfico.

Todos los dispositivos tienen una serie de factores que acaban limitando las prestaciones, basados en la estructura interna (hardware y software).

Algunos desarrolladores de balanceadores de carga sólo soportan Fast Ethernet, limitándolos así a 100Mbps.

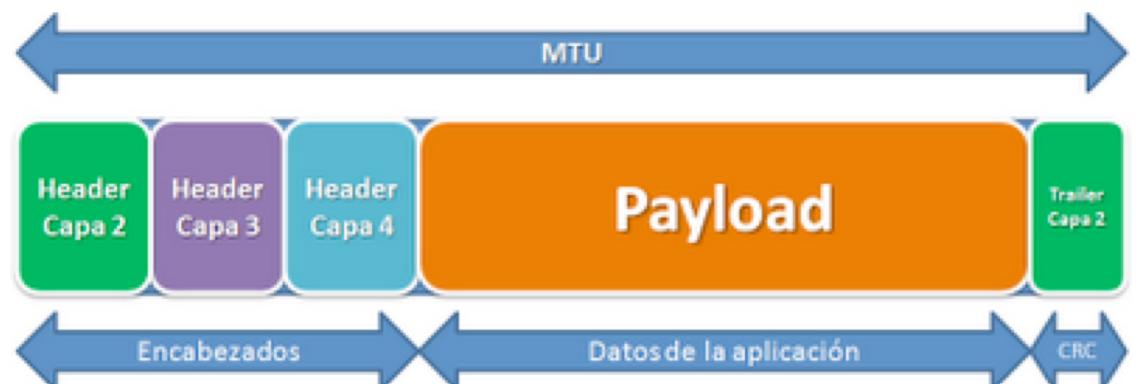
Algunas implementaciones no tienen el hardware o el software adecuado, con lo que quedan limitados a transferencias máximas de 80Mbps.

Rendimiento, en bits por segundo

Se mide en bits por segundo. Es combinación de las variables "*tamaño del paquete*" y "*paquetes por segundo*".

El paquete típico tiene un tamaño máximo (MTU, Maximum Transmittable Unit) de 1.5KB.

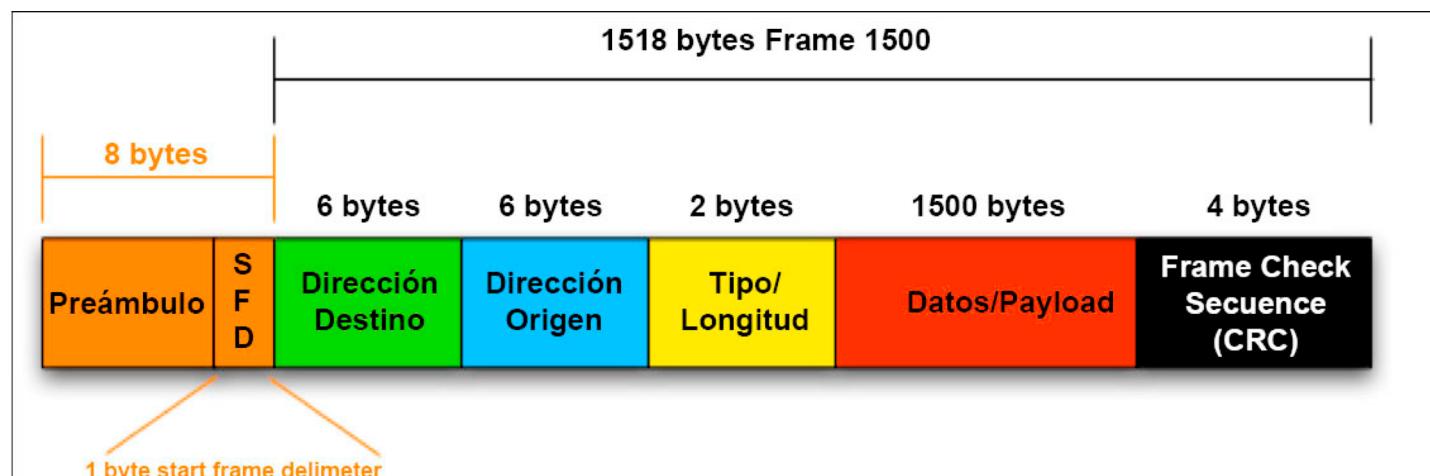
Si hay que enviar más datos, se trocean en paquetes de este tamaño máximo.



Rendimiento, en bits por segundo

Ejemplo:

- un acceso por HTTP usando el método GET a un recurso de 100 bytes podrá servirse en un solo paquete.
- un acceso por GET a un archivo de 32KB necesitará 21 paquetes, con 1.5KB de información útil (*payload*) en cada uno.

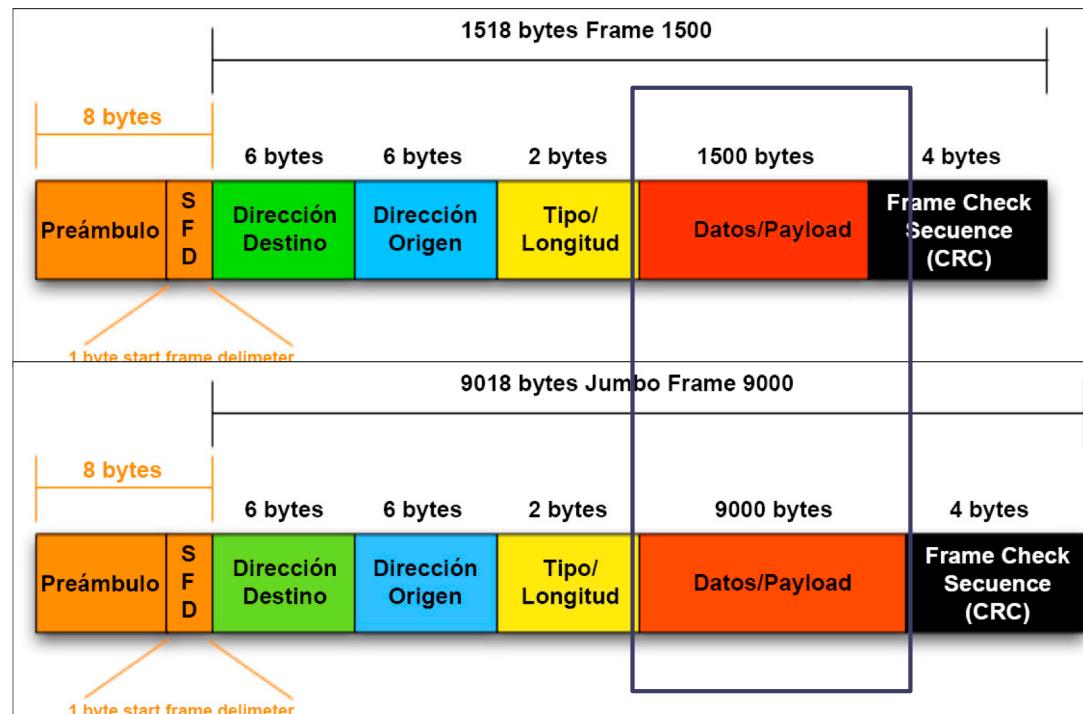


¿Y para grandes transferencias?

¡Jumbo Frame!

<http://www.mundonas.com/2013/05/jumbo-frames.html>

“con la salida de **las redes Gigabit** se implementó la posibilidad de modificar el tamaño de esos paquetes para optimizar tanto tráfico como transferencia de información”



¿Y para grandes transferencias?

¡Jumbo Frame! - Ventajas

- Al ser paquetes más grandes requieren de menor dedicación de CPU para su procesado en tarjetas de red, routers, etc, además de llegar antes
- Se minimiza el uso de bytes para la asignación de los datos que acompañan a la MTU, dedicando más bytes a datos enviados.
- Aliviamos la carga de protocolo y menor fragmentación de la información por la red.
- Se mejora la carga en los programas que gestionan el tráfico de red, como son los Firewalls (Cortafuegos).

¿Y para grandes transferencias?

¡Jumbo Frame! - Desventajas

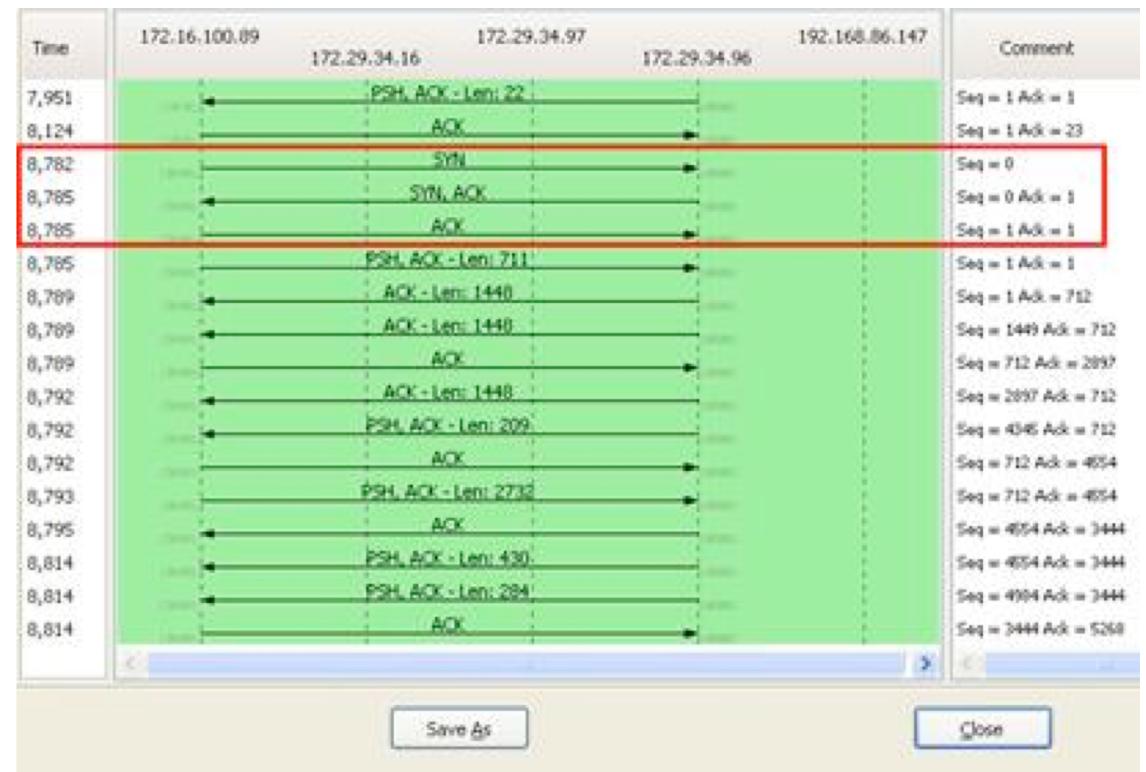
- Al aumentar el tamaño de los paquetes aumenta la latencia puesto que el tiempo de envío de los mismos es mayor, no siendo conveniente en redes con fuerte streaming.
- En redes de baja/media calidad provoca que si un paquete no pasa la comprobación de errores debe volverse a reenviar, siendo mayor la información que se reenvía y el tiempo que se pierde en la tarea.
- No es muy recomendable forzar el uso de Jumbo Frames (paquetes de información grandes) con redes que manejan paquetes pequeños (P2P) puesto que no se rinde al 100% de la configuración, es conveniente probar la administración por parte del router de esta mezcla.

Análisis del tráfico con Wireshark

<https://bit.ly/2XQ87EK>

- *análisis de una captura de tráfico realizada con Wireshark*
- *estudio de casos con la herramienta de simulación de redes Cisco Packet Tracer*

Ej: Establecimiento de conexión
(handshake de tres vías)



Análisis del tráfico con Wireshark

<https://bit.ly/2XQ87EK>

Ej: Establecimiento de conexión (handshake de tres vías)

```
# Frame 208: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
# Ethernet II, Src: Hewlett_P_00:0e:84 (00:0e:84:3a:27:80), Dst: Cisco_3a:27:80 (00:0e:84:3a:27:80)
# Internet Protocol Version 4, Src: 172.16.100.89 (172.16.100.89), Dst: 172.29.34.96 (172.29.34.96)
# Transmission Control Protocol, Src Port: netplay-port1 (3640), Dst Port: http-alt (8080), Seq: 0, Len: 0
    Source port: netplay-port1 (3640)
    Destination port: http-alt (8080)
    [Stream index: 22]
    Sequence number: 0 (relative sequence number)
    Header length: 44 bytes
# Flags: 0x02 (SYN)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0... .... = ECN-Echo: Not set
    ..0.... = Urgent: Not set
    ...0.... = Acknowledgement: Not set
    ....0... = Push: Not set
    ....0.. = Reset: Not set
# ACK=1, = Syn: Set
# [Expert Info (Chat/Sequence): Connection establish request (SYN): server port http-alt]
# [Message: Connection establish request (SYN): server port http-alt]
# [Severity Level: chat]
# [Group: Sequence]
    ....0 = Fin: Not set
    window size: 65535
# [Expert Info (Sequence): Destination host lost]
0000  00 0e 84 3a 27 80 18 a9  05 a0 cf ec 08 00 45 00  ...;....E.
0010  00 40 df 65 40 00 80 06  00 00 ac 10 64 59 ac 1d  ;.0.0...dy..
0020  22 60 0e 38 1f 90 82 f4  2f 90 00 00 00 00 b0 02  ..8..../
0030  ff ff 76 cc 00 00 02 04  05 b4 01 03 03 01 01 01  ..v......
0040  08 04 00 00 00 00 00 00  00 00 01 01 04 02  ......




```

Análisis del tráfico con Wireshark

<https://bit.ly/2XQ87EK>

Ej: Establecimiento de conexión (handshake de tres vías)

```
# Frame 209: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
# Ethernet II, Src: Cisco_3a:27:80 (00:0e:84:3a:27:80), Dst: HewlettP_a0:cf:ec (18:a9:05:a0:cf:ec)
# Internet Protocol, Src: 172.29.34.96 (172.29.34.96), Dst: 172.16.100.89 (172.16.100.89)
# Transmission Control Protocol, src Port: http-alt (8080), Dst Port: netplay-port1 (3640), Seq: 0, Ack: 1, Len: 0
    Source port: http-alt (8080)
    Destination port: netplay-port1 (3640)
    [Stream index: 22]
    Sequence number: 0      (relative sequence number)
    Acknowledgement number: 1      (relative ack number)
    Header length: 36 bytes
    Flags: 0x12 (SYN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0... .... = ECN Echo: Not set
        ..0.... = Urgent: Not set
        ...1.... = Acknowledgement: Set
        ....0... = Push: Not set
        ....0.. = Reset: Not set
    # ... .1. = Syn: Set
        [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port http-alt]
            [Message: Connection establish acknowledge (SYN+ACK): server port http-alt]
            [Severity level: Chat]
            [Group: Sequence]
            ....0 = Fin: Not set
    Window size: 0x0001
0010  00 38 36 ff 40 00 7c 06 e8 d9 ac 1d 22 60 ac 10 .86.0.|. .....
0020  64 59 1f 90 0e 38 ef 14 2a fe 82 f4 2f 91 90 b dy...8.. *.....
0030  20 00 1c 03 00 00 02 04 05 b4 04 02 08 0a 04 9e ..... .
0040  42 15 00 00 00 00 B.....
```

Análisis del tráfico con Wireshark

<https://bit.ly/2XQ87EK>

Ej: Establecimiento de conexión (handshake de tres vías)

Análisis del tráfico con Wireshark

<https://bit.ly/2XQ87EK>

No.	Time	Source	Destination	Protocol	Info
153	7.029025	172.16.100.89	128.223.1.183	DNS	Standard query A www.bancochile.cl
Frame 153: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)					
x Ethernet II, Src: HewlettP_a0:cfc:ec (18:a9:05:a0:cfc:ec), Dst: Cisco_3a:27:80 (00:0e:84:3a:27:80)					
x Internet Protocol, Src: 172.16.100.89 (172.16.100.89), Dst: 128.223.1.183 (128.223.1.183)					
x User Datagram Protocol, Src Port: 61375 (61375), Dst Port: domain (53)					
Domain Name System (query)					
{response id: 154}					
transaction ID: 0xf341					
Flags: 0x0100 (Standard query)					
0... = Response: Message is a query					
,000 0.... = Opcode: Standard query (0)					
.... .0. = Truncated: Message is not truncated					
.... .1. = Recursion desired: do query recursively					
.... .0. = Z: reserved (0)					
.... .0.... = Non-authenticated data: unacceptable					
Questions: 1					
Answer RRs: 0					
Authority RRs: 0					
Additional RRs: 0					
Queries					
www.bancochile.cl: type A, class IN					
Name: www.bancochile.cl					
Type: A (Host address)					
Class: IN (0x0001)					
0000	00:00:00:44:27:80	18:09:01:49:0f:ec	00:00:45:00:00:00		
0010	00:00:00:44:27:80	00:00:00:45:00:00	10:00:59:80:00:00		
0020	01:07:ef:bf:00:35	00:25:46:44:73:41	01:00:00:00:00:01		
0030	00:00:00:00:00:00	02:77:77:04:62:62	00:00:00:00:00:01		
0040	03:68:69:62:61:02	63:63:6c:00:00:01	00:00:00:00:00:03		

petición al DNS

No.	Time	Source	Destination	Protocol	Info
154	7.029025	172.16.100.89	128.223.1.183	DNS	standard query response CNAME www.ds1b.bancochile.cl A 200.14.133.61
Frame 154: 116 bytes on wire (928 bits), 116 bytes captured (928 bits)					
x Ethernet II, src: Cisco_3a:27:80 (00:0e:84:3a:27:80), dst: HewlettP_a0:cfc:ec (18:a9:05:a0:cfc:ec)					
x Internet Protocol, src: 128.223.1.183 (128.223.1.183), dst: 172.16.100.89 (172.16.100.89)					
x User Datagram Protocol, src Port: domain (53), dst Port: 61375 (61375)					
Domain Name System (response)					
{request id: 153}					
[Time: 0.000765000 seconds]					
transaction ID: 0xf341					
Flags: 0x8180 (Standard query response, No error)					
1... = Response: Message is a response					
,000 0.... = Opcode: Standard query (0)					
.... .0. = Authoritative: server is not an authority for domain					
.... .0. = Truncated: Message is not truncated					
.... .1. = Recursion desired: do query recursively					
.... .1.... = Recursion available: server can do recursive queries					
.... .0. = Z: reserved (0)					
.... .0.... = Answer authenticated: Answer/authority portion was not authenticated by the server					
.... .0.... = Non-authenticated data: unacceptable					
.... .0000 = Reply code: No error (0)					
questions: 1					
Answer RRs: 2					
Authority RRs: 0					
Additional RRs: 0					
queries					
www.bancochile.cl: type A, class IN					
Name: www.bancochile.cl					
Type: A (Host address)					
Class: IN (0x0001)					
Answers					
0020	64:59:00:35:bf:00	52:04:53:11:01:00	00:00:00:00:00:00		
0030	00:00:00:00:00:00	00:00:00:77:04:63	01:00:00:00:00:01		
0040	03:68:69:62:61:02	63:63:6c:00:00:01	00:00:00:00:00:03		
0050	00:00:00:00:00:00	00:00:00:77:04:67	01:00:00:00:00:01		
0060	03:68:69:62:61:02	63:63:6c:00:00:01	00:00:00:00:00:03		

respuesta del DNS

Análisis del tráfico con Wireshark

Ejercicio 5.1:

Instalar Wireshark y observar cómo fluye el tráfico de red en el balanceador de la máquina M3 mientras se le hacen peticiones HTTP y HTTPS. Ejecuta al menos 3 peticiones al balanceador.

Realiza un análisis de una sesión TCP (establecer conexión y cierre) de peticiones HTTP y HTTPS y escribe tus propias conclusiones. Puedes ilustrarlo con capturas de pantalla.

Índice



1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Tipos de tráfico

Hay patrones de tráfico muy comunes:

- HTTP
- FTP o streaming
- tienda web

Patrón de tráfico	Métrica más importante	Segunda métrica más importante	Métrica menos importante
HTTP	Conexiones por segundo	Rendimiento	Total de conexiones concurrentes
FTP/streaming	Rendimiento	Total de conexiones concurrentes	Conexiones por segundo
Tienda web	Total de conexiones concurrentes	Conexiones por segundo	Rendimiento

Tipos de tráfico

Tráfico HTTP:

consume ancho de banda intensivamente y genera muchas conexiones por segundo.

HTTP 1.0, se necesita una conexión para cada objeto.
HTTP 1.1 envía con una sola conexión varios objetos.

Necesidad de hacer las páginas web ligeras, de forma que los usuarios puedan cargarlas rápidamente.

Tipos de tráfico

Tráfico FTP / streaming:

Tras una conexión inicial (ya que usa UDP como protocolo), se envía una gran cantidad de información.

El número de conexiones para este tipo de tráfico es muy bajo comparado con la cantidad de información enviada.

Consumen el ancho de banda máximo rápidamente.

Tipos de tráfico

Tráfico tipo “tienda web”:

La velocidad es el factor más importante.

Buena experiencia de usuario: si el usuario se desespera navegando en la tienda web, gastará poco dinero...

No se necesita un alto ancho de banda, ni tampoco va a haber demasiadas conexiones por segundo.

Sin embargo, el sitio debe dar soporte al máximo de usuarios navegando en sesiones largas al mismo tiempo.



Índice

1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Límite de las prestaciones

Existe un límite de tráfico de red suficientemente alto que produce una degradación grave en las prestaciones.

En unos casos ese límite es más fácil de alcanzar que en otros.

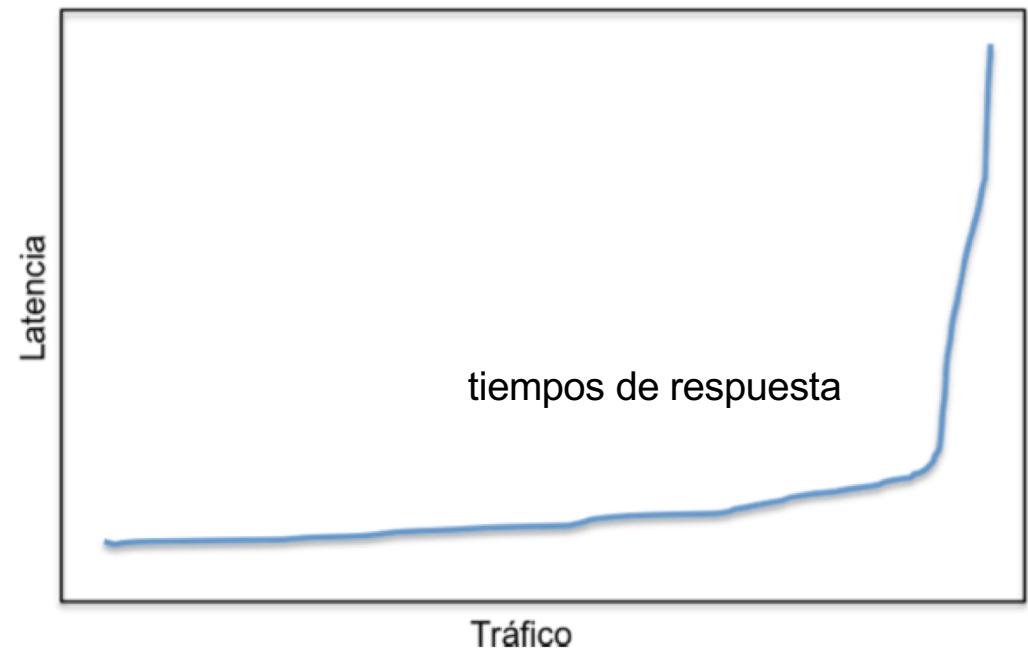
Llegado a ese límite los **tiempos de respuesta** en las conexiones HTTP se degradan completamente, haciendo **imposible la conexión**.

Límite de las prestaciones

Si lo representamos gráficamente:

Esta degradación en las prestaciones se debe a los cuellos de botella.

Hay que estudiar los datos de la monitorización durante las pruebas para determinar las carencias.

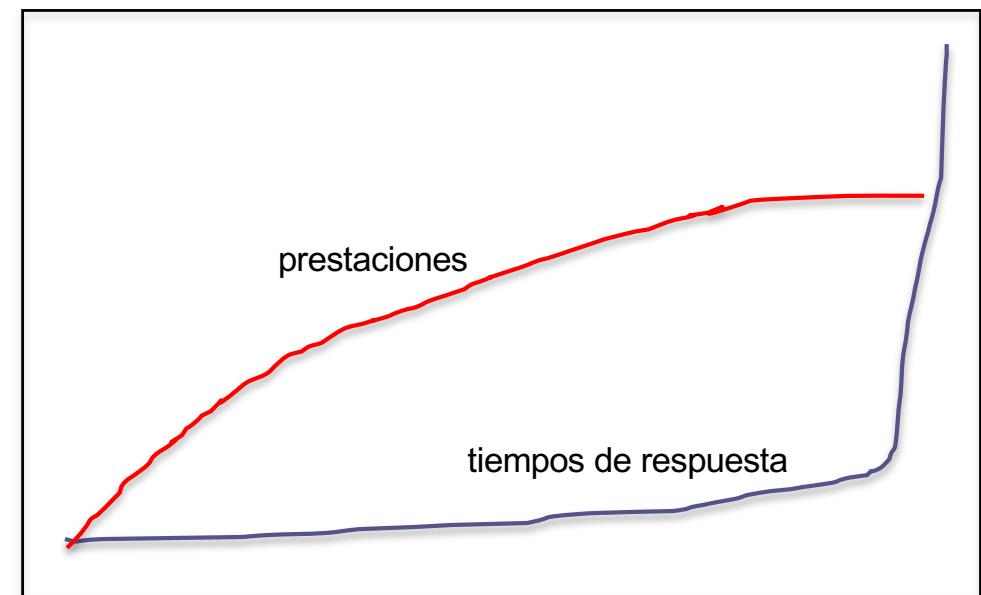


Límite de las prestaciones

Ejemplo 1: curva característica

Al subir la carga, las prestaciones se degradan y dejan de crecer.

Coincide con el incremento de los tiempos de respuesta.

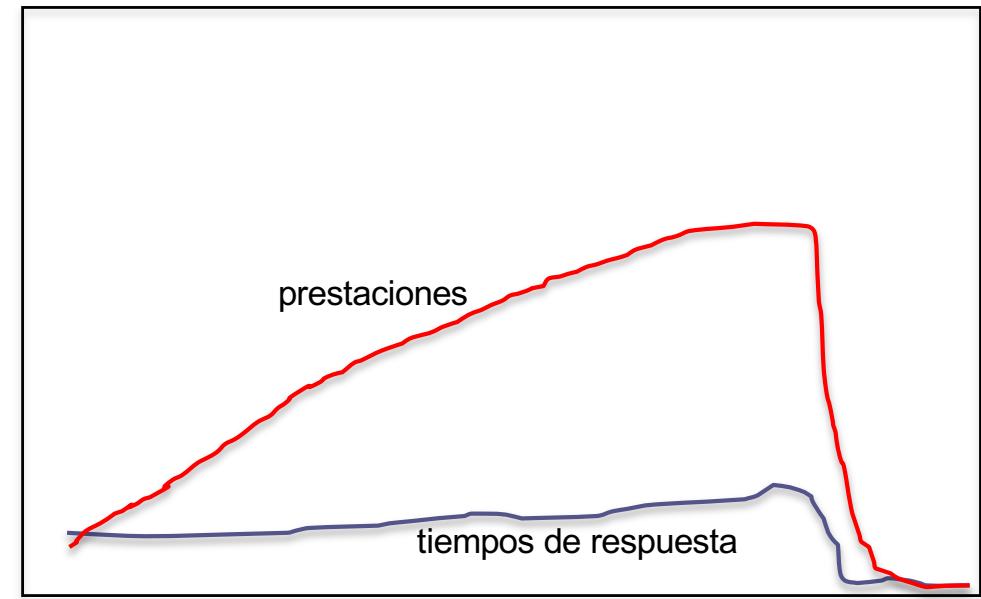


Límite de las prestaciones

Ejemplo 2: cuando ocurre algún problema...

Al fallar algún servicio,
caen las prestaciones.

También los tiempos de
respuesta, al terminar las
transacciones rápidamente
con un error.



En estos casos habría que examinar los *logs* (acceso y error).

Límite de las prestaciones

Cada dispositivo de red puede comportarse de forma diferente, llegando a **cuelgues o reinicios**.

Estos límites son difíciles de alcanzar...
pero a mayor **número de características activas** en un balanceador, será más fácil de alcanzar el límite:



Si es capaz de procesar tráfico a 90Mbps, puede ver reducido su rendimiento a la mitad si le pedimos que haga análisis de URLs y que de soporte de cookies (requieren uso más intensivo de la CPU para inspeccionar los paquetes completos y no solo la cabecera).

Índice



1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software para hacer tests
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Software para los tests

Necesarias **herramientas** para ejecutar en máquinas clientes y **crear una carga HTTP** específica.

Se suelen usar benchmarks como **SPECweb** o **WebStone** para simular un número determinado de clientes:

- <http://www.spec.org/benchmarks.html>
- <http://sourceforge.net/projects/webstone/>

El número de usuarios de un servidor web puede ser del orden de los millones de usuarios, así es que **simular un número pequeño de clientes no es realista**



¿Cómo hacer los tests?

Consideraciones a tener en cuenta cuando vamos a evaluar el rendimiento de un sitio web real:

1. **Primero fijar un número alto de usuarios.** Calcular el tiempo medio cuando hay un alto número de usuarios haciendo peticiones al sitio web.
2. **Después, evaluar cómo se comporta el servidor cuanto tiene el doble de usuarios.** Un servidor que tarda el doble en atender al doble de usuarios será mejor que otro que al doblar el número de usuarios (la carga) pase a tardar el triple.

¿Cómo hacer los tests?

En sistemas críticos, en lugar de usar (o desarrollar) una herramienta para generar la carga para los tests, se le puede encargar a una empresa externa especializada.

Algunas empresas ofrecen su herramienta y realizan los tests:

- Micro Focus Intl. - Segue Software (SilkPerformer)
- HP (LoadRunner)
- Micro Focus Intl. - Compuware (QALoad)
- Rational (SiteLoad)
- Radview (WebLoad)

Tipos de pruebas

Tenemos que elegir correctamente el tipo de pruebas:

- Humo (Smoke): pruebas preliminares para comprobar que el sistema está listo para los siguientes tests.
- Carga (Load): cargas lo más parecidas a la real. Se ejecutan en periodos cortos (1h). Para determinar los tiempos de respuesta que tendrán los usuarios.
- Capacidad (Capacity): actividad creciente hasta detectar el punto de saturación.

Tipos de pruebas

Tipos de pruebas (II):

- Estrés (Stress): para analizar el efecto de aplicar de forma continuada una carga por encima de la capacidad del sistema.
- Sobrecarga (Overload): aplicar fuertes picos de carga durante cortos periodos.
- Estabilidad (Stability): cargas lo más similares posibles a la real, aplicadas durante 1 día o 1 semana.

Durante los tests, monitorización

Durante la sesión de pruebas, recoger mediciones que nos indiquen lo que está ocurriendo en el sistema en cada momento y como reacciona éste en función de la carga introducida:

- Medidas de la calidad de servicio ofrecida por el sistema a los usuarios (estadísticas proporcionadas por la misma **herramienta de simulación de carga**)
- Medidas relativas al consumo de recursos del sistema (utilizando las **herramientas del sistema operativo**)

Software para los tests

Diversas herramientas para comprobar el rendimiento de servidores web. Línea de comandos y de interfaz gráfica:

- **Apache Benchmark** <http://httpd.apache.org/docs/2.2/programs/ab.html>
- **Siege** <https://www.joedog.org/siege-home/>
- **Weighttp** <https://redmine.lighttpd.net/projects/weighttp/wiki>
- httpperf
- OpenWebLoad
- The Grinder
- OpenSTA
- JMeter
- Webstone (Mindcraft) <http://mindcraft.com/webstone/>

Las de línea de comandos sobrecargan menos las máquinas.

Software para los tests

Estas herramientas permiten comprobar el rendimiento de cualquier servidor web (Apache, MS Internet Information Services -IIS-, nginx, Cherokee, Tomcat, lighttpd, thttpd, etc).

Comprobar el rendimiento del hardware, software o de alguna modificación que le hayamos hecho.

Índice



1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

Apache Benchmark

ab no simula con total fidelidad el uso del sitio web que pueden hacer los usuarios habitualmente.

Pide la misma página repetidamente. Los usuarios reales no solicitan siempre la misma página.

Las medidas dan una idea aproximada del rendimiento del sitio, pero no reflejan el rendimiento real.

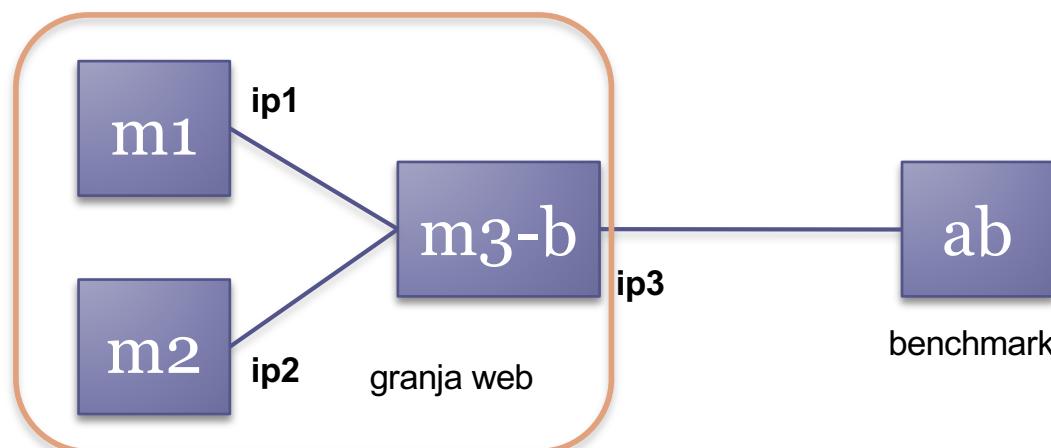
Va bien para testear cómo se comporta el servidor antes y después de modificar cierta configuración.

Teniendo los datos del “estado base”, podemos comparar cómo afecta una nueva configuración.

Apache Benchmark

Debemos ejecutar el benchmark en otra máquina diferente a la que hace de servidor web.

Ambos procesos no deben consumir recursos de la misma máquina (veríamos un menor rendimiento).



Sin embargo, al hacerlo remotamente, introducimos cierta latencia debido a las comunicaciones.

Apache Benchmark

Cada vez que ejecutemos el test obtendremos resultados ligeramente diferentes.

Esto es debido a que en el servidor hay diferente número de procesos en cada instante, y además la red puede encontrarse más sobrecargada en un momento que en otro.

Lo ideal es hacer al menos 30 ejecuciones, sacar resultados en **media y desviación estándar**, y representarlo gráficamente de forma adecuada.

Apache Benchmark. Ejemplo

Para ejecutar el benchmark, usamos la sintaxis:

```
ab -n 1000 -c 5 http://maquina.com/prueba.html
```

-n 1000 => se solicita mil veces en total la URL

-c 5 => se hacen peticiones de 5 en 5 (conurrencia)

```
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
...
Concurrency Level:          10
Time taken for tests:    0.474 seconds
Complete requests:         1000
Failed requests:           0
Write errors:               0
...
Requests per second:   2109.82 [#/sec] (mean)
Time per request:      4.740 [ms] (mean)
Time per request:         0.474 [ms] (mean, across all concurrent requests)
Transfer rate:             733.49 [Kbytes/sec] received
```



Índice

1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
11. Siege

httpperf

httpperf es una herramienta para medir el rendimiento de sitios web.

Originalmente se desarrolló en los laboratorios de investigación de Hewlett-Packard.

Si tenemos varios clientes, deberíamos hacer la ejecución en todos simultáneamente.

De todas formas, puesto que los tests tardan varios minutos, que la ejecución comience con un segundo de diferencia, no afectará significativamente al resultado final.

Httpperf. Ejemplo

La sintaxis de ejecución es:

```
httpperf --server maquina.com --uri /prueba.html --port 80 \
--num-conn 5000 --num-call 10 --rate 200 --timeout 5
```

Test del servidor maquina.com, puerto 80. Pedirá, de forma repetida, la página llamada "prueba.html"

Abrirá un total de **5000 conexiones** TCP para hacer con cada una de ellas peticiones HTTP (implica hacer la petición y esperar la respuesta).

Hará **10 peticiones por conexión**, y las hará a **200 conexiones por segundo** (implica 2000 peticiones/seg).

timeout = segundos que el cliente esperará respuesta. Si pasa ese tiempo, considerará que la llamada habrá fallado.

Httpperf. Ejemplo

La salida será similar a:

```
Total: connections 4986 requests 39620 replies 39620 test-duration 29.294 s
```

```
Connection rate: 170.2 conn/s (5.9 ms/conn, <=1022 concurrent connections)
```

```
Connection time [ms]: min 922.1 avg 4346.7 max 8045.6 median 4414.5 stddev 1618.6
```

```
Connection time [ms]: connect 643.6
```

```
Connection length [replies/conn]: 10.000
```

```
Request rate: 1352.5 req/s (0.7 ms/req)
```

```
Request size [B]: 58.0
```

```
Reply rate [replies/s]: min 1195.0 avg 1344.7 max 1393.1 stddev 84.1 (5 samples)
```

```
Reply time [ms]: response 370.3 transfer 0.0
```

```
Reply size [B]: header 167.0 content 2048.0 footer 0.0 (total 2215.0)
```

```
Reply status: 1xx=0 2xx=39620 3xx=0 4xx=0 5xx=0
```

```
CPU time [s]: user 1.35 system 27.95 (user 4.6% system 95.4% total 100.0%)
```

```
Net I/O: 3002.2 KB/s (24.6*10^6 bps)
```

```
Errors: total 1038 client-timo 1024 socket-timo 0 connrefused 0 connreset 0
```

```
Errors: fd-unavail 14 addrunavail 0 ftab-full 0 other 0
```

Httpperf. Ejemplo

El ratio de peticiones (request rate) es menor de 2000 (sale 1352.5 peticiones/seg).

O bien el servidor está saturado y no soporta 2000 peticiones por segundo, o bien el cliente no puede llegar a hacerlas.

Ya sabemos el límite de nuestro servidor.

Ha habido 1038 errores:

- 1024 *timeouts* (tardó más de 5seg en llegar la respuesta a httpperf)
- 14 *fd-unavail*: httpperf intentaba abrir otro descriptor de fichero y no podía (se alcanzó el límite de descriptores abiertos por proceso establecido en el kernel de Linux, que es precisamente 1024).

La línea *Net I/O* muestra 24.6 Mbps (muy por debajo de 100 Mbps). Si estuviese cerca de 90Mbps habría que mejorar el ancho de banda.

Índice



1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
- [10. OpenWebLoad]**
11. Siege

OpenWebLoad

OpenWebLoad es otra herramienta de línea de comandos para medir el rendimiento de servidores web:

```
openload [options] http://maquina.com 10
```

El programa recibe **dos parámetros**, muy similares a los de las herramientas anteriores:

- La URL de la página en el servidor.
- El número de clientes simultáneos que simularemos (es un parámetro opcional y el valor por defecto es 5).

El programa ofrece más opciones:

http://openwebload.sourceforge.net/cmd_parms.html

OpenWebLoad. Ejemplo

Sintaxis de uso:

```
openload maquina.com 10
```

Salida del benchmark:

URL: <http://maquina.com:80/>

Clients: 10

MaTps 355.11, Tps 355.11, Resp Time 0.015, Err 0%, Count 511
MaTps 339.50, Tps 199.00, Resp Time 0.051, Err 0%, Count 711
MaTps 343.72, Tps 381.68, Resp Time 0.032, Err 0%, Count 1111
MaTps 382.04, Tps 727.00, Resp Time 0.020, Err 0%, Count 1838
MaTps 398.54, Tps 547.00, Resp Time 0.018, Err 0%, Count 2385
MaTps 425.78, Tps 670.90, Resp Time 0.014, Err 0%, Count 3072

Total TPS: 452.90

Avg. Response time: 0.021 sec.

Max Response time: 0.769 sec



Índice

1. Introducción
2. Conexiones por segundo
3. Número de conexiones concurrentes
4. Rendimiento, en bits por segundo
5. Tipos de tráfico
6. Límite en las prestaciones
7. Software
8. Apache benchmark
9. httpperf
10. OpenWebLoad
- [11. Siege]**

Siege

Siege es una herramienta de generación de carga HTTP para benchmarking parecida a Apache Benchmark:

```
siege -b -t60S -v http://maquina.com
```

El programa recibe **varios parámetros**:

- -b para indicar que haga los tests de forma continua, y no interactivos.
- -t para indicar el tiempo que queremos que esté en ejecución el programa.
- -v para indicar que genere una salida detallada.
- La URL a la que queremos “atacar”.

Por defecto usará 15 conexiones concurrentes durante ese tiempo indicado.

Siege. Ejemplo

```
** Preparing 15 concurrent users for battle.  
The server is now under siege...  
HTTP/1.1 200 0.03 secs: 9405 bytes ==> GET /  
... ... ... ...  
HTTP/1.1 200 0.02 secs: 9405 bytes ==> GET /  
[error] socket: 208723968 connection refused.: Connection refused  
HTTP/1.1 200 5.63 secs: 9405 bytes ==> GET /  
... ... ... ...  
HTTP/1.1 200 0.01 secs: 9405 bytes ==> GET /  
Lifting the server siege... done.
```

Transactions:	17317 hits
Availability:	99.99 %
Elapsed time:	59.97 secs
Data transferred:	155.35 MB
Response time:	0.05 secs
Transaction rate:	288.76 trans/sec
Throughput:	2.59 MB/sec
Concurrency:	13.95
Successful transactions:	17317
Failed transactions:	1
Longest transaction:	20.44
Shortest transaction:	0.01