Tema 5

Medir prestaciones

Pedro A. Castillo Valdivieso
Depto Arquitectura y Tecnología de Computadores
Universidad de Granada
pacv@ugr.es

Índice

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.
- Sí son problemas de rendimiento y degradación de recursos.
- Detectar posibles cuellos de botella e ineficiencias

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

Conexiones por segundo

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**.

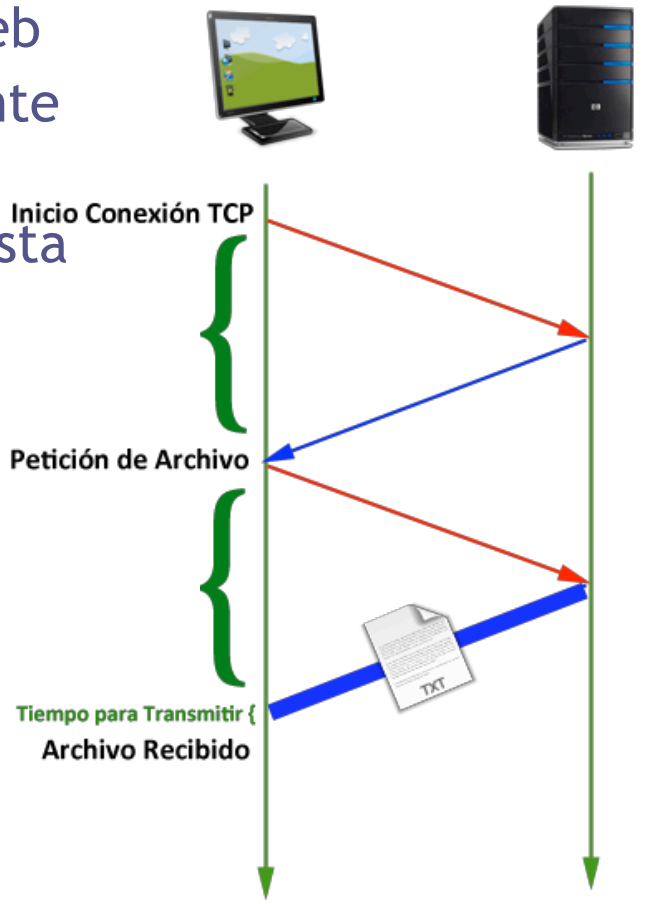
Las aperturas y cierres de conexiones consumen muchos recursos.

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

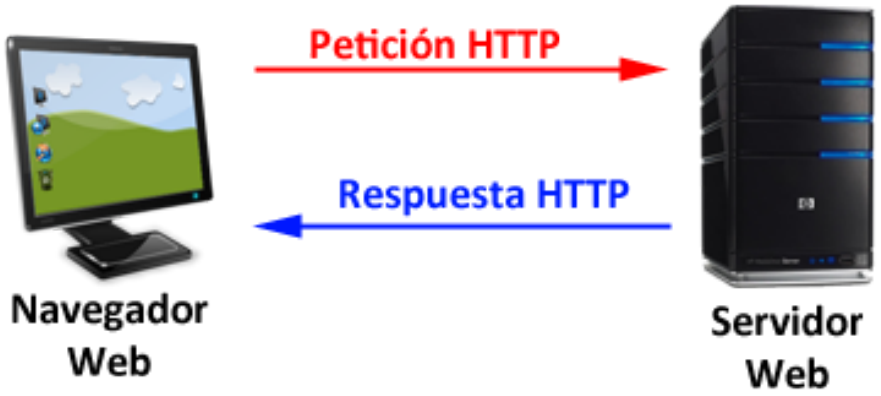
Ahora ya pueden comenzar a enviarse datos desde el servidor al cliente (normalmente será una página web).



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 a la hora de adquirir y configurar un balanceador de carga.



Conexiones por segundo

Ejercicio:

Buscar información sobre cómo calcular el número de conexiones por segundo.

Para empezar, podéis revisar las siguientes webs:

http://bit.ly/1ye4yHz

http://bit.ly/1PkZbLJ

Índice

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

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.

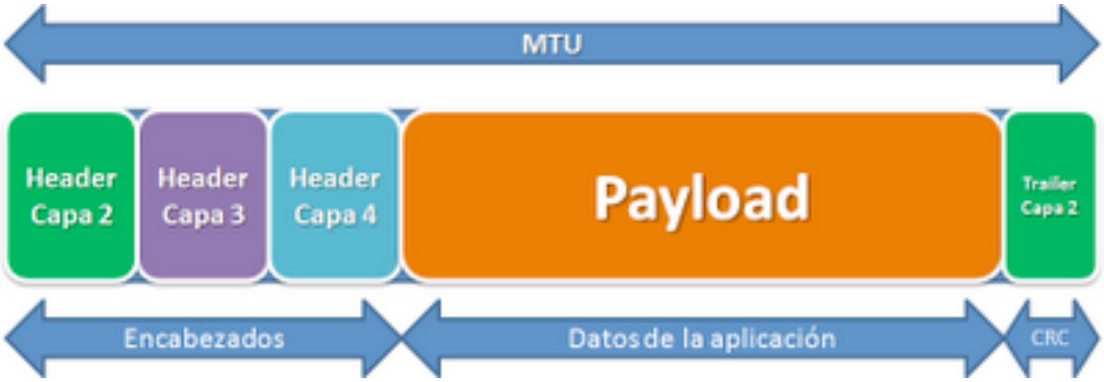
Algunas implementaciones no tienen el hardware o el software adecuado, con lo que quedan limitados a transferencias máximas de 80Mbps.

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, Maximun Transmitable Unit) de 1.5KB.

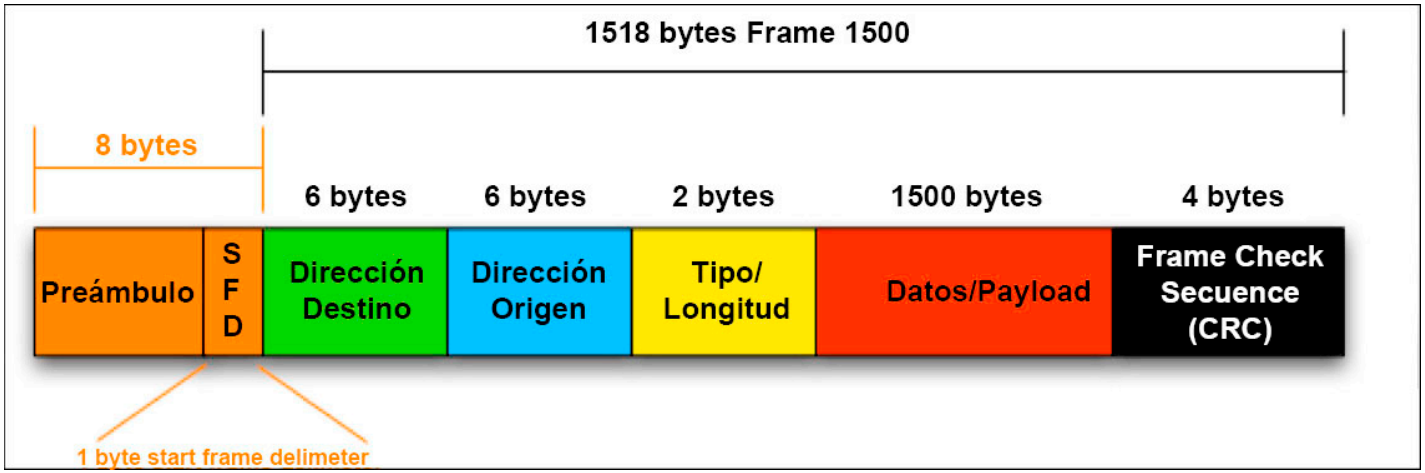
Si hay que enviar más datos, se trocean en paquetes de este tamaño máximo.



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?

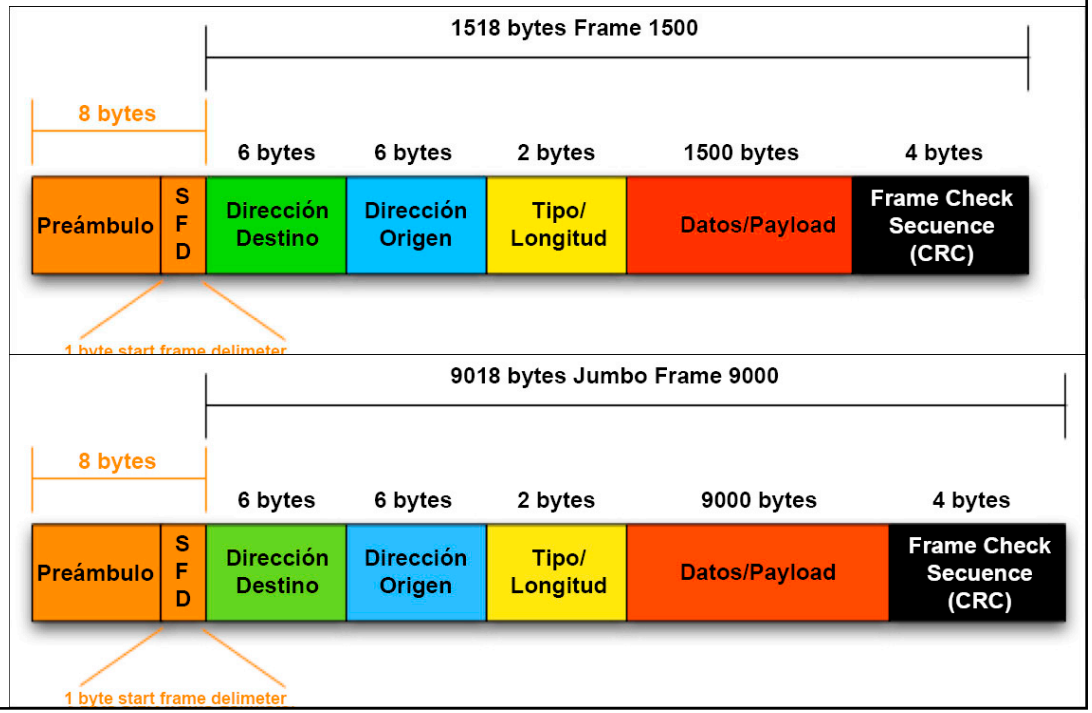
¡Jumbo Frame!

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"*

- menor dedicación de CPU para su procesado en tarjetas de red, routers, etc, además de llegar antes.
- dedicando más bytes a datos enviados

- aumenta la latencia.
- problemas si hay que reenviar...



Análisis del tráfico con Wireshark

http://bit.ly/1g0dkKj

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

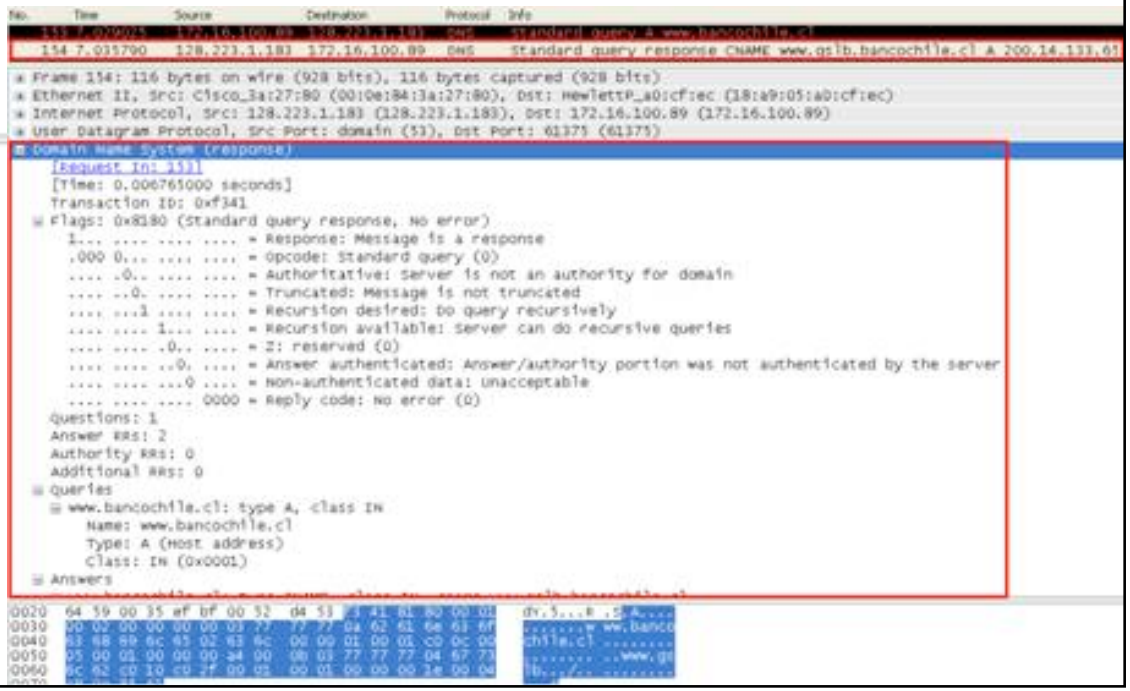
Ej: Establecimiento de conexión (handshake de tres vías)

Análisis del tráfico con Wireshark

http://bit.ly/1g0dkKj

petición al DNS

respuesta del DNS



Análisis del tráfico con Wireshark

Ejercicio:

Revisar los análisis de tráfico que se ofrecen en:
http://bit.ly/1g0dkKj

Instalar wireshark y observar cómo fluye el tráfico de red en uno de los servidores web mientras se le hacen peticiones HTTP.

Herramientas de análisis

Ejercicio:

Buscar información sobre características, disponibilidad para diversos SO, etc de herramientas para monitorizar las prestaciones de un servidor.

Para empezar, podemos comenzar utilizando las clásicas de Linux:

- top
- vmstat
- netstat

Índice

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 hay demasiadas conexiones por segundo.

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

Índice

Límite de las prestaciones

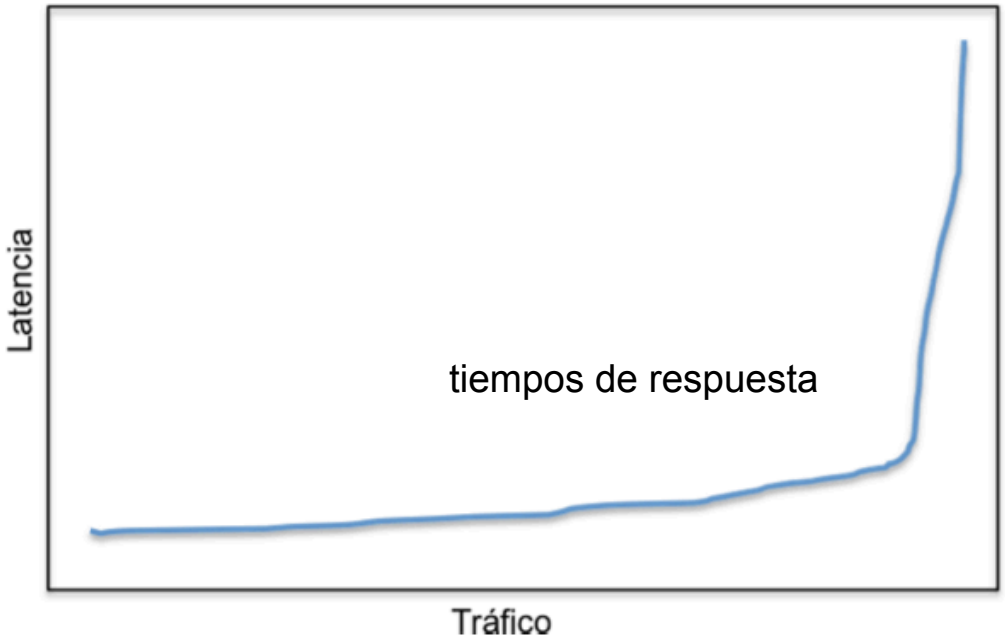
Existe un límite de tráfico de red suficientemente alto que produce una **degradación grave en las prestaciones**.

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.

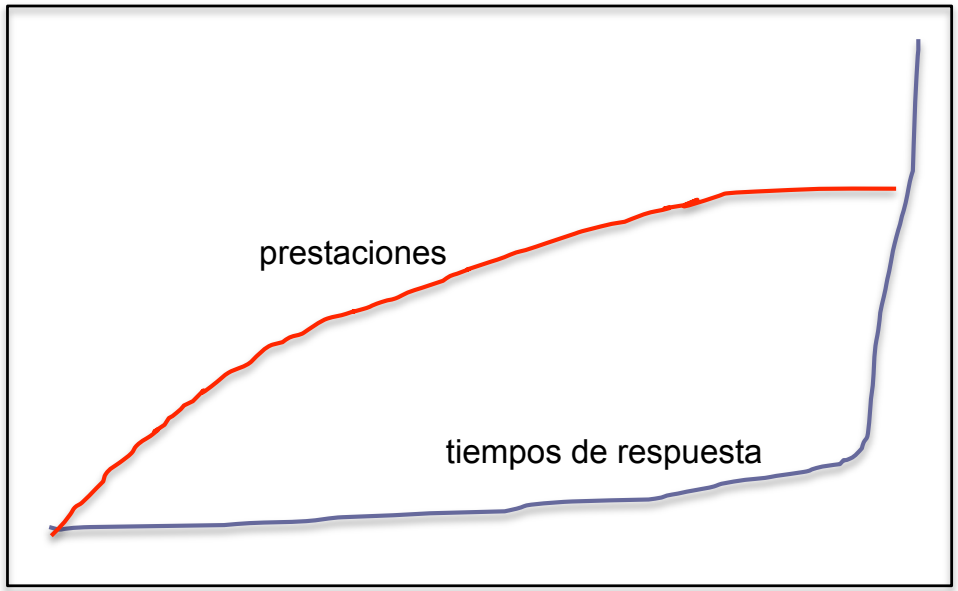
Hay que estudiar los datos de la monitorización durante las pruebas para determinar las carencias.

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.

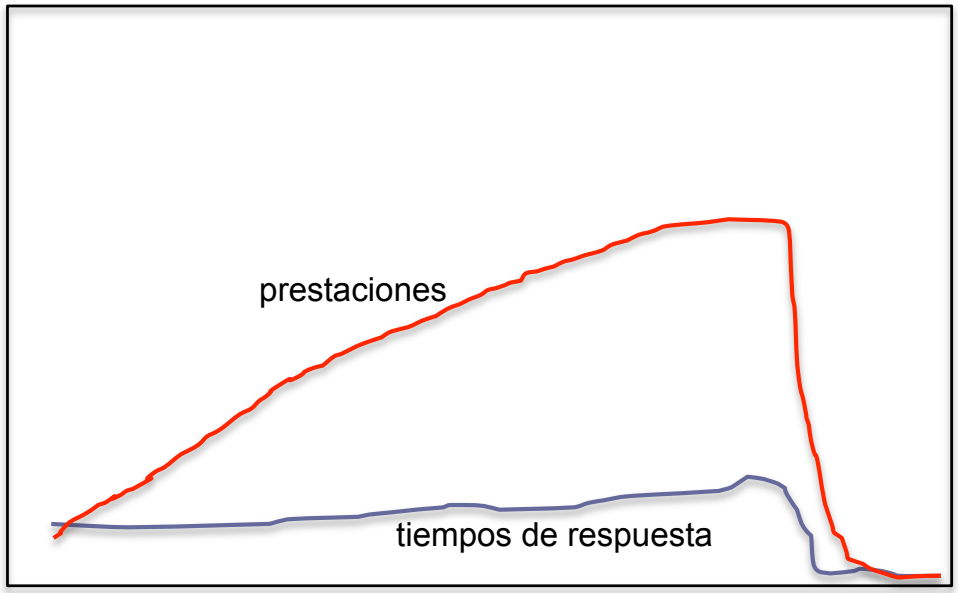


Límite de las prestaciones

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

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**. La idea es que 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**
- **httperf**
- **OpenWebLoad**
- The Grinder
- OpenSTA
- Jmeter
- siege
- Webstone (Mindcraft) http://mindcraft.com/webstone/

Las de línea de comandos sobrecargan menos las máquinas que estamos usando.

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

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.

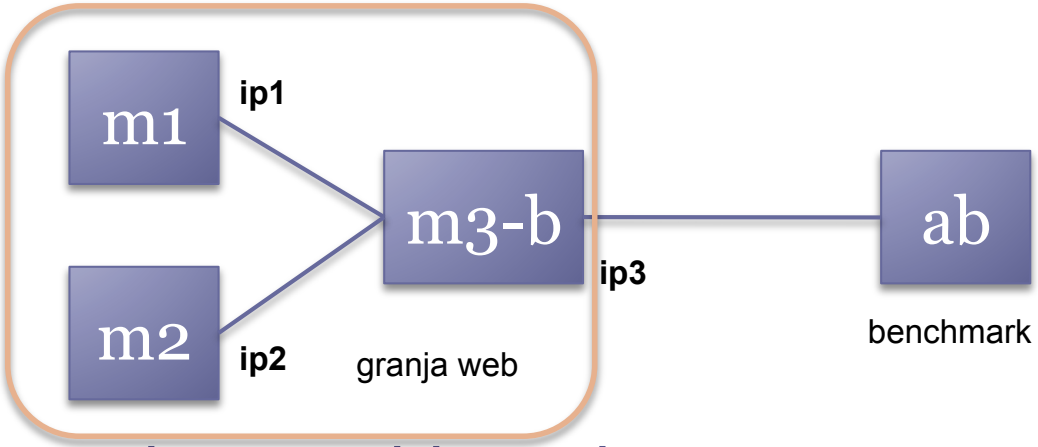
Va bien para **testear cómo se comporta el servidor** antes y después de modificar cierta configuración.

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.

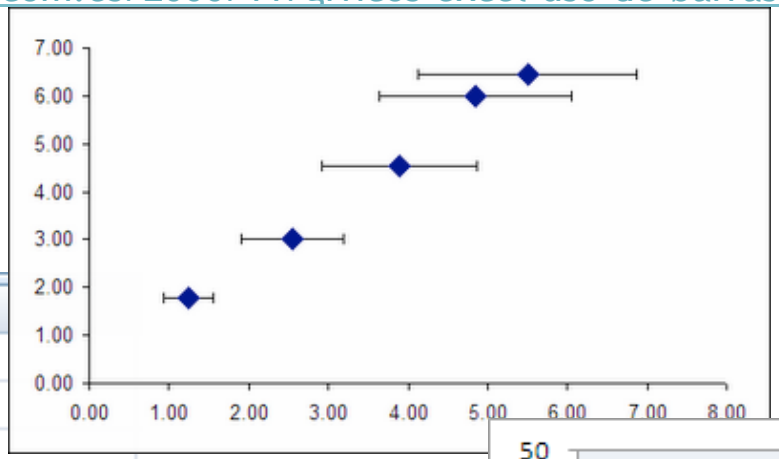
Esto es debido a que en el servidor hay diferente número de procesos en cada instante, y además la red puede encontrarse más sobrecargada en un momento que en otro.

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

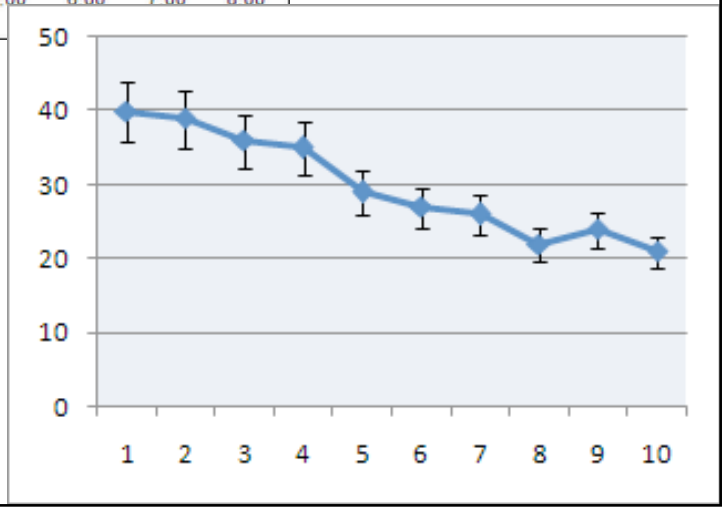
Apache Benchmark

¿Cómo mostrar los resultados?

- http://excelforo.blogspot.com.es/2011/05/grafico-en-excel-con-barras-de-error.html
- http://office.microsoft.com/es-es/excel-help/agregar-cambiar-o-quitar-barras-de-error-en-un-grafico-HP010342159.aspx
- http://jld-excel-grafico.blogspot.com.es/2006/11/grficos-excel-uso-de-barras-de-error.html



	A	B	C
1	**AÑO**	**Importe**	
2	2009	1.120,00	
3	2010	1.100,00	
4	2011	1.125,00	
5	2012	1.150,00	
6	2013	1.125,00	
7	2014	1.100,00	www.excelforo.blogspot.com
8			
9	media	1.120,00	=PROMEDIO(B2:B7)
10	desviación	**18,71**	=DESVEST(B2:B7)



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 (concurrencia)

```
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

httperf

httperf 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.

Httperf. Ejemplo

La **sintaxis** de ejecución es:

```
httperf --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.

Httperf. 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
```

Httperf. 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 httperf)
- 14 *fd-unavail*: httperf 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

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 muchas 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

Presentar resultados gráficamente

Tras realizar un experimento, debemos presentar los datos obtenidos en forma de tabla y también gráficamente de la forma más adecuada.

Tutoriales:

- http://www.slideshare.net/berumenll/representacion-grafica-de-datos
- http://www.profesorenlinea.cl/matematica/Graficos.html
- y los recursos que dejamos en PRADO2.