Introducción

Los test de rendimiento evalúan el rendimiento de nuestra aplicación con el fin de poder transmitirle al cliente cual es la carga máxima que soporta su sistema. Puesto que no se puede probar directamente en el sistema, el lugar donde se debería probar es en la máquina de Pre-Producción ya que es el lugar más parecido al entorno real. Debido a que la máquina de Pre-Producción es una máquina virtual ejecutada desde distintos ordenadores, el rendimiento varía entre ellas y por esto hemos decidido recopilar los dos test que, tras ejecutarlo varias veces en distintas máquinas, obtienen un peor rendimiento (y destacable) entre los otros Test.

A continuación se van a ir detallando los citados test, ya que suponen el cuello de botella de la aplicación y se van a ir proponiendo opciones para mejorar el rendimiento de las acciones realizadas por los test consiguiendo así mejorar el rendimiento de la aplicación en general.

El primer test será denominado de aquí en adelante Test_A y hace referencia al caso de uso creado por el requisito funcional de nivel A 2.1 (Acme-Barter 2.0) el cual implica manejar las descripciones de los atributos de los objetos. El segundo test (de aquí en adelante Test_B) hace referencia al caso de uso creado por el requisito funcional de nivel C 12.2 (Acme-Barter 1.0) el cual implica manejar los términos legales.

Detalles del entorno

Antes de continuar debemos detallar en que máquina se han lanzado dichos tests y que configuraciones tenía tanto la máquina virtual como los propios test:

Máquina física

Portátil HP-Pavilion con I7, 8GB de RAM y disco HDD

Máquina virtual

2 Núcleos al 100% con las características PAE/NX habilitadas y con 1536 MB de RAM

Detalles JMeter

A ambos test se le han introducido un retraso de campana gaussiana de 1500 ms en cada acción que sería realizada por un usuario real (hacer click o rellenar un formulario) con el fin de que sea lo más real posible.

Respecto al número de usuarios, el “ramp-up period” y el número de iteraciones por cada usuario los datos en cada test han sido los siguientes:

- Test_A: 75 usuarios, en 1 segundo (ramp-up period) y cada usuario se ejecuta 50 veces.
- Test_B: 100 usuarios, en 1 segundo (ramp-up period) y cada usuario se ejecuta 100 veces.

Desarrollo de los test

Ambas pruebas tienen como cuello de botella el mismo error, realizar muchas peticiones a la base de datos de escritura/lectura del tipo save() o findAll() para comprobar los datos introducidos. Como podemos observar en ambas gráficas, esto es apreciable en las columna "90% Line" en las líneas listar y crear en el Test_A y en listar, crear y editar en el Test_B.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	11250	938	266	2499	4	55730	0.00%	8.1/sec	40.9
/styles/common.css	3750	42	11	123	1	1336	0.00%	2.7/sec	1.6
/scripts/jquery-ui.js	3750	178	62	458	4	44586	0.00%	2.7/sec	1237.3
/scripts/jquery.js	3750	109	32	316	3	1711	0.00%	2.7/sec	733.4
/styles/jmenu.css	3750	36	9	102	1	1313	0.00%	2.7/sec	5.1
/styles/displaytag.css	3750	34	8	97	1	1020	0.00%	2.7/sec	8.1
/scripts/jmenu.js	3750	33	8	98	1	1275	0.00%	2.7/sec	28.8
/images/logo.png	3750	155	44	451	3	7080	0.00%	2.7/sec	1249.3
/favicon.ico	3750	39	8	104	1	4996	0.00%	2.7/sec	4.4
/security/login.do	3750	67	18	176	3	7250	0.00%	2.7/sec	12.7
/j_spring_security_check	3750	1896	775	4997	6	54686	0.00%	2.7/sec	15.1
/item/user/list.do	3750	1069	401	2621	9	49740	0.00%	2.7/sec	20.3
/attribute-description/user/list.do	7500	1466	804	3505	12	47343	0.00%	5.4/sec	42.8
/attribute-description/user/create.do	3750	1357	649	3132	13	49327	0.00%	2.7/sec	17.4
/attribute-description/user/edit.do	3750	2949	1743	6630	30	58169	0.00%	2.7/sec	22.0
TOTAL	67500	762	85	2205	1	58169	0.00%	48.6/sec	3412.6

Tabla peticiones web Test_A

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	30000	1034	314	2787	4	30230	0.00%	11.1/sec	54.4
/styles/common.css	10000	29	11	58	1	2967	0.00%	3.7/sec	2.1
/styles/displaytag.css	10000	26	10	51	1	14007	0.00%	3.7/sec	10.5
/scripts/jmenu.js	10000	25	10	51	1	4172	0.00%	3.7/sec	38.1
/styles/jmenu.css	10000	22	9	46	1	2839	0.00%	3.7/sec	6.7
/scripts/jquery.js	10000	67	25	142	3	4189	0.00%	3.7/sec	966.5
/scripts/jquery-ui.js	10000	87	31	179	4	3865	0.00%	3.7/sec	1634.8
/images/logo.png	10000	64	26	145	4	4902	0.00%	3.7/sec	1705.7
/security/login.do	10000	25	7	55	3	4250	0.00%	3.7/sec	19.3
/j_spring_security_check	10000	1154	434	2911	9	47731	0.00%	3.7/sec	19.0
/folder/actor/list.do	20000	889	113	2736	5	21265	0.00%	7.4/sec	45.8
/images/arrow_off.png	10000	30	11	57	1	11561	0.00%	3.7/sec	1.6
/message/actor/create.do	20000	2719	1959	6537	14	40318	0.00%	7.4/sec	49.1
/images/arrow_down.png	10000	23	9	49	1	11572	0.00%	3.8/sec	1.6
TOTAL	180000	659	32	2298	1	47731	0.00%	66.5/sec	4530.0

Tabla peticiones web Test_B

Esto se debe (como podemos observar en las siguientes gráficas) al estado del procesador, la memoria RAM y el acceso al disco del sistema los cuales se encuentran continuamente al límite de sus posibilidades.

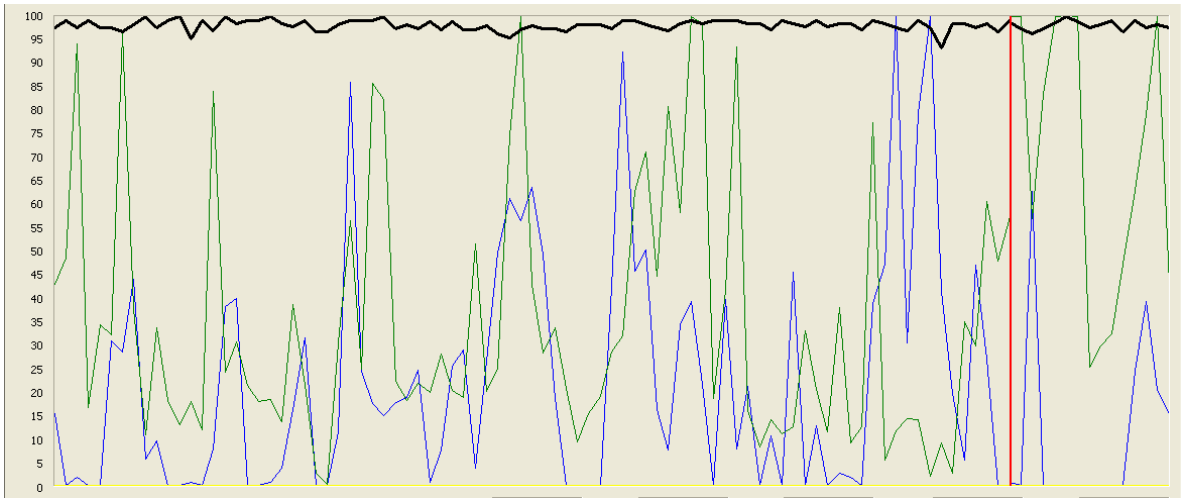

Ilustración carga sistema durante ejecución de Test_A

Nota: Los colores representan la memoria ram (azul), el disco duro (verde), el procesador (negro) y la tarjeta de red (amarillo).

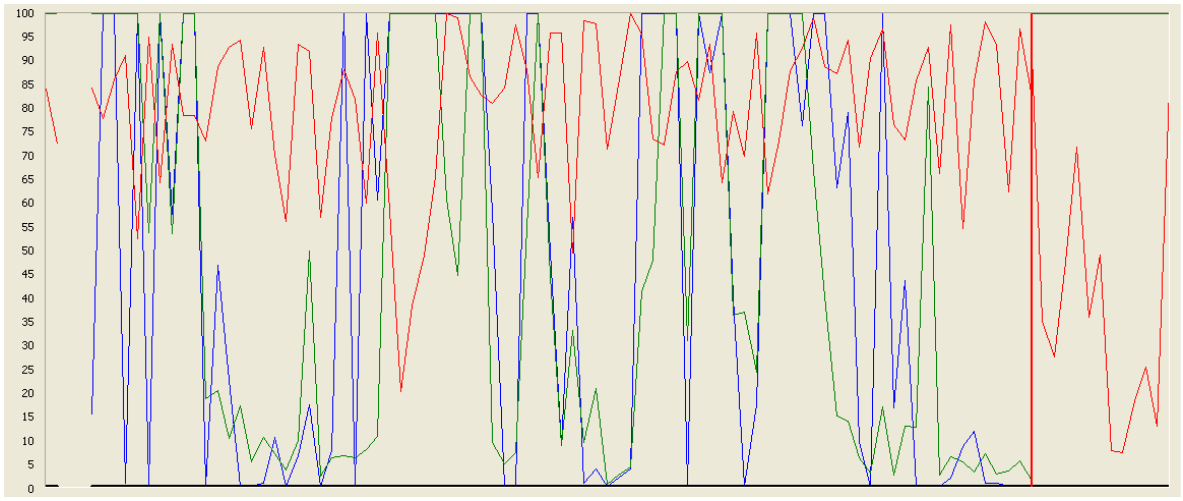

Ilustración carga sistema durante ejecución de Test_B

Nota: Los colores representan la memoria ram (azul), el disco duro (verde), el procesador (rojo) y la tarjeta de red (negro).

Para mejorar estos resultados, la opción por parte del cliente es comprar procesadores con más potencia, discos duros cuyo acceso de lectura escritura sea más rápido y memorias RAM más rápidas y con más capacidad (lo que podría evitar tener que mejorar el disco duro temporalmente) y por parte de los desarrolladores de la aplicación (nosotros) usar estas llamadas “lentas” el menor número de veces posible lo que conllevaría a programar de manera más eficiente en general.

Con todo esto podemos concluir que el límite de nuestro sistema serían unos 75 usuarios por segundo.