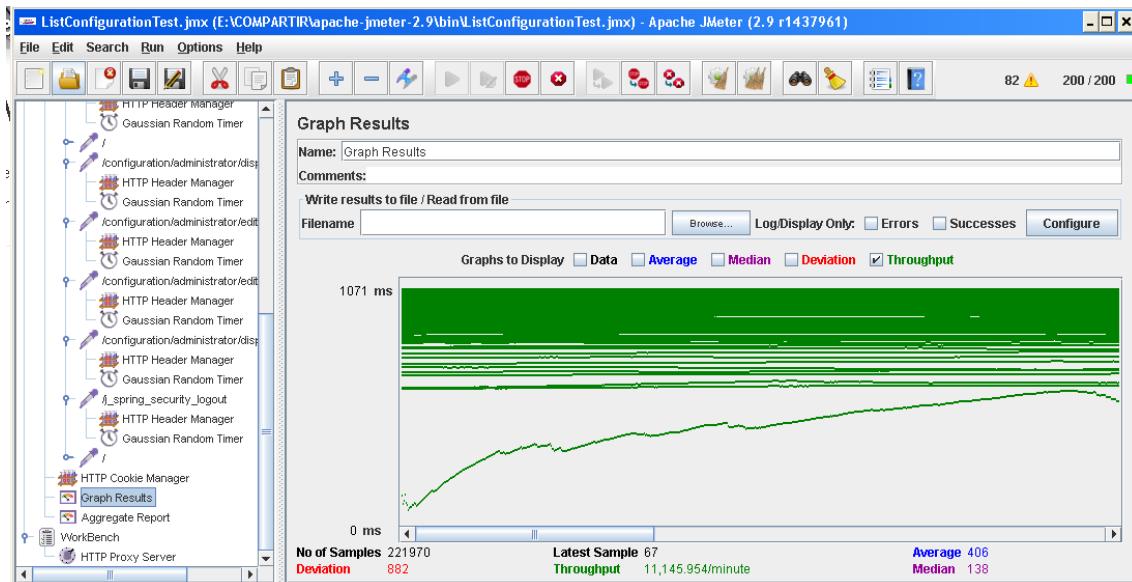


Caso de uso: editar taboo Word.

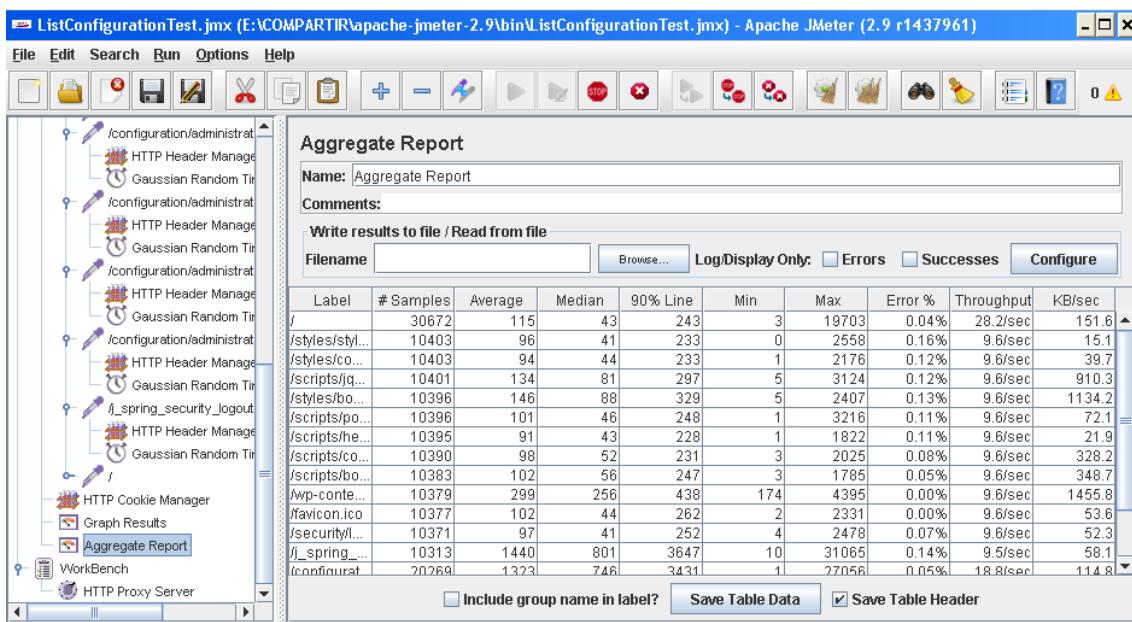
Nos autenticamos como administrador, accedemos a la configuración, añadimos una palabra y nos des autenticamos.

En este test, nos aparecen errores con 250 usuarios realizando 100 acciones cada uno, sin embargo, este error desaparece con 220 usuarios realizando 100 acciones cada uno. Todos los hilos habrán comenzado en un segundo.

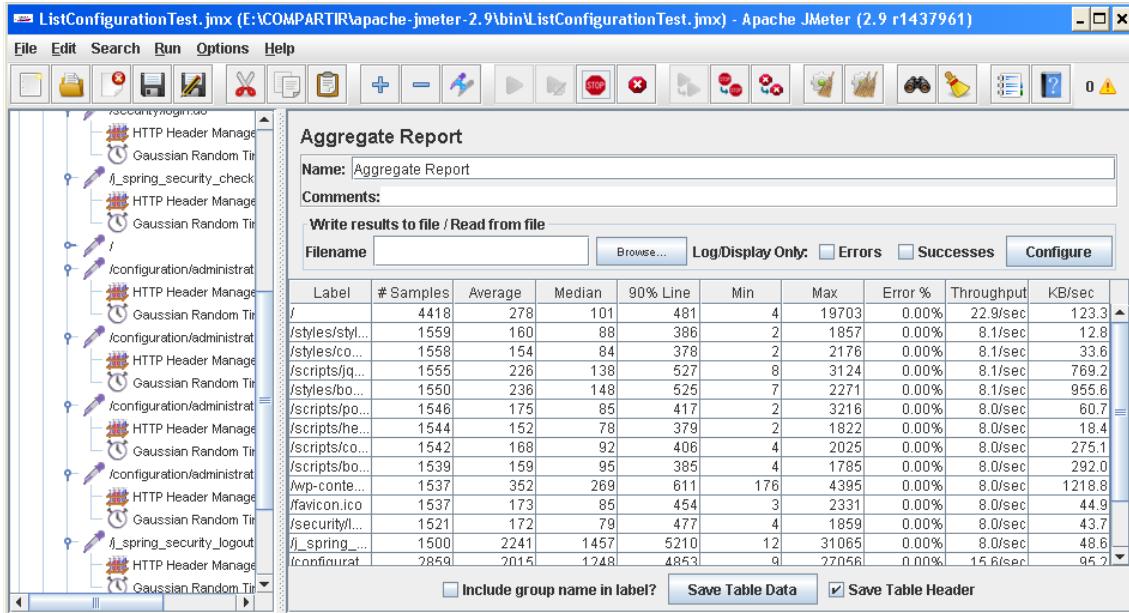
Captura del Aggregate Report con 250 usuarios:



Captura del Graph Results con 250 usuarios:

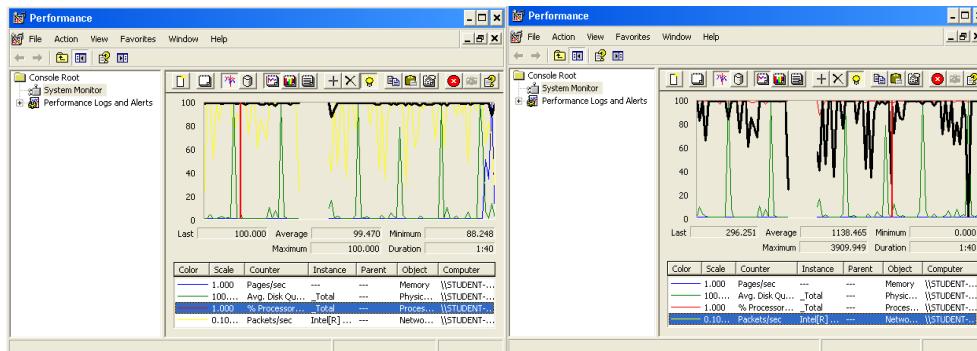


Captura del Graph Results con 220 usuarios:



En este caso de uso podemos destacar como cuello de botella, aquella que tiene el label: “/wp-content...” que es una imagen que debe obtener de internet. Vemos que las acciones que tienen mayor 90% Line, son aquellas en las que el usuario realiza alguna acción.

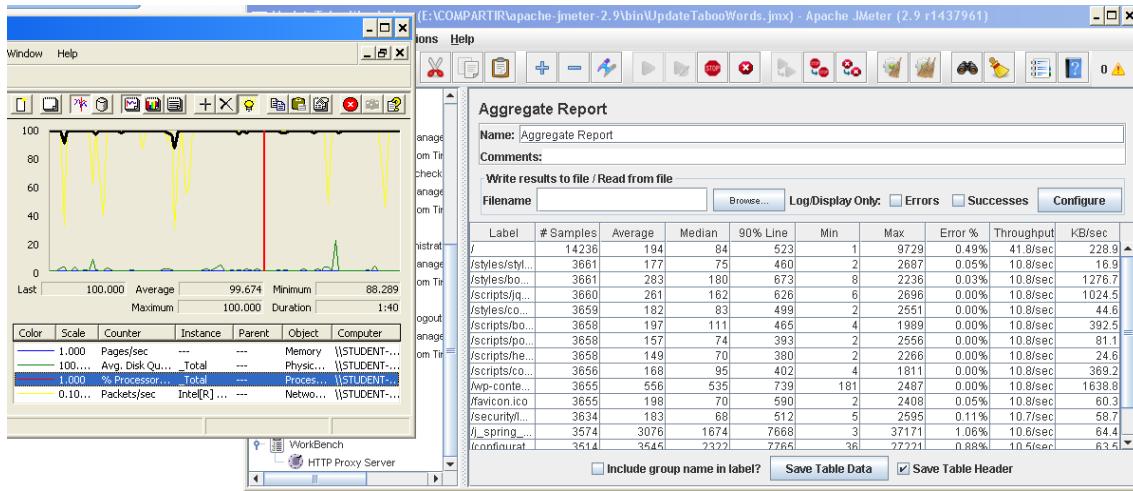
Al implementar ambos casos los límites máximos en rendimiento se producen en la CPU y la tarjeta de red:



Caso de uso: actualizar las palabras tabúes.

Nos autenticamos como administrador, accedemos a esta actualización y nos des autenticamos.

En este test, nos aparecen errores con 220 usuarios realizando 100 acciones cada uno, sin embargo, este error desaparece con 200 usuarios realizando 100 acciones cada uno. Todos los hilos habrán comenzado en un segundo.



En este caso es el procesador es el cuello de botella.

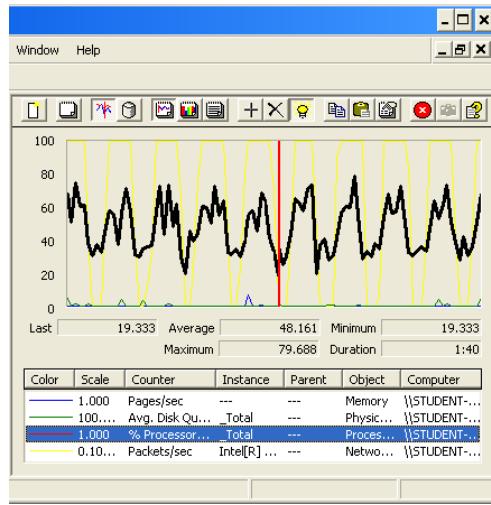
Como vemos, las peticiones que tardan más tiempo son las peticiones para actualizar las palabras taboo, y aquella que envía la información de login al usuario. Todo esto lo vemos en la columna Line 90%.

Caso de uso: borrar Follow-Up.

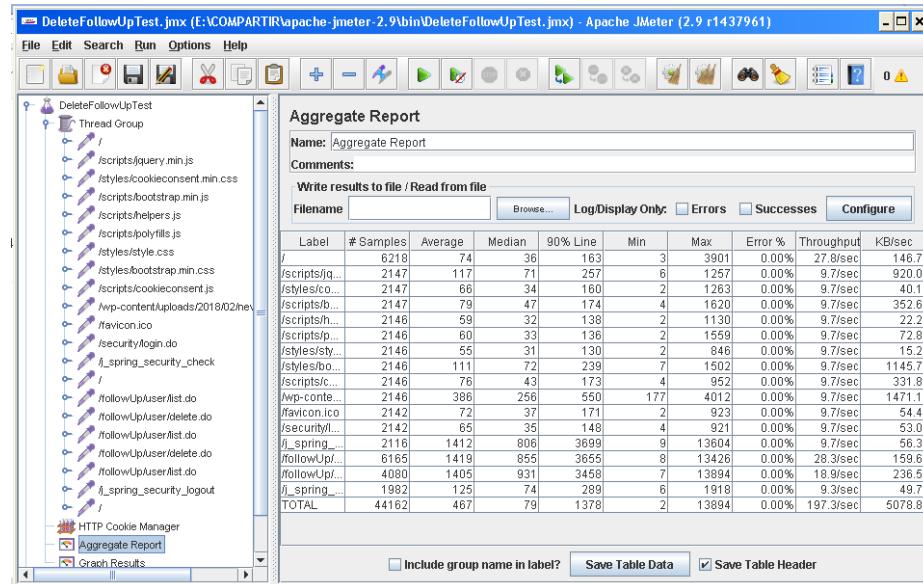
Nos autenticamos como usuario, accedemos al listado de los follow-ups de un usuario, borramos algunos.

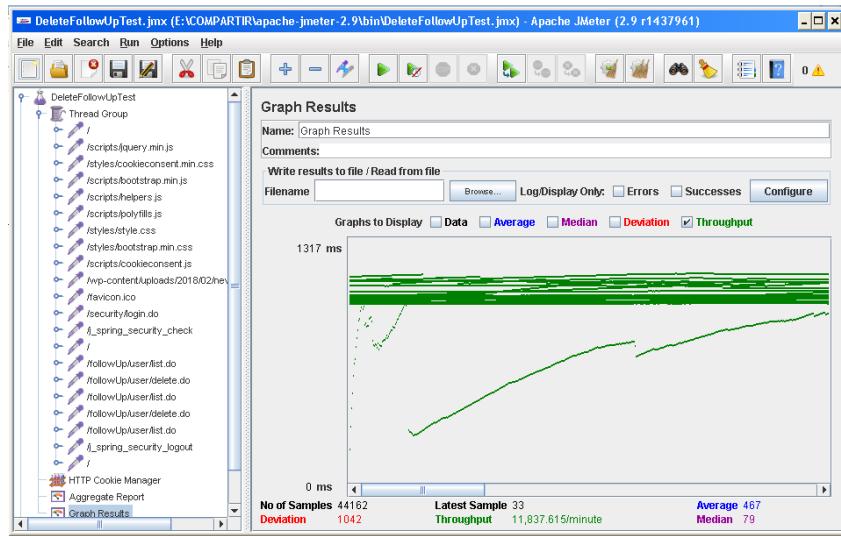
En este caso, obtenemos el máximo antes de producirse un error, con 200 usuarios conectados, realizando la acción 100 veces. Si, por el contrario, los usuarios conectados son 220, encontramos algunos errores derivados de la gran carga que tiene la CPU.

Al disminuir el número de usuarios que usan la aplicación en un momento dado a 100, la carga del procesador disminuye considerablemente como observamos en la figura:



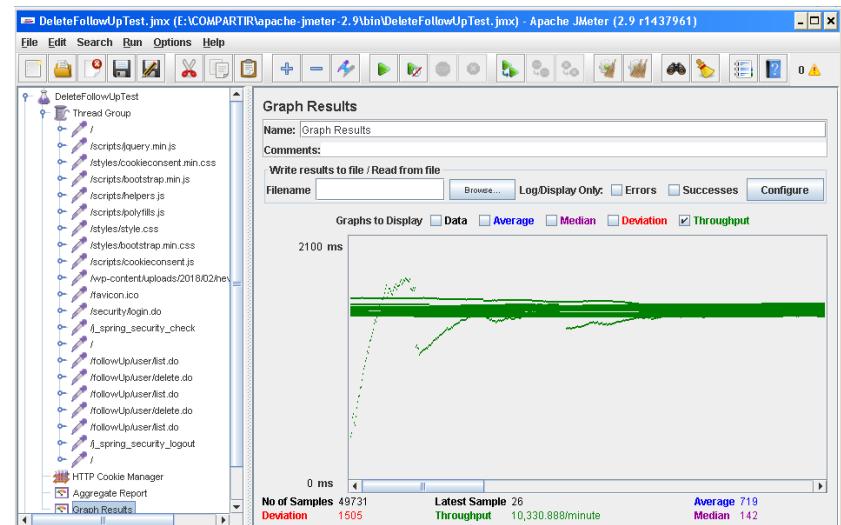
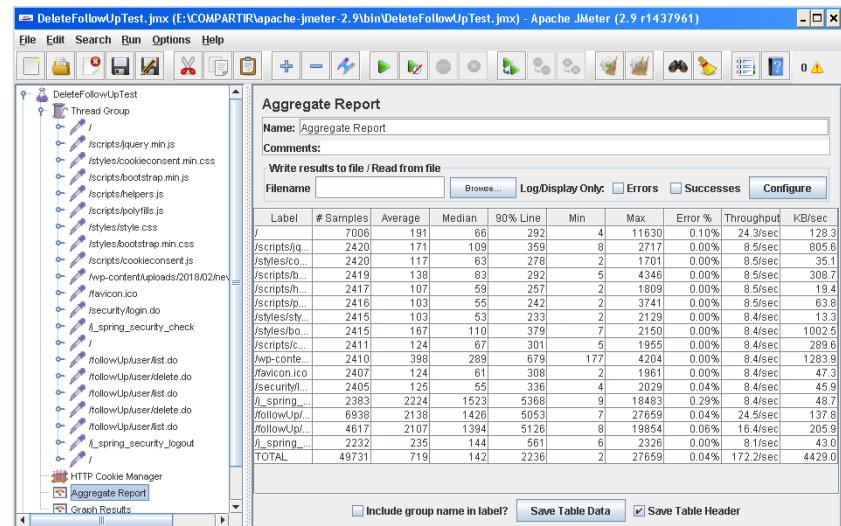
Al ser 200 usuarios los listeners quedan de esta forma:





Como vemos, las peticiones que tardan más tiempo son las peticiones para borrar el objeto, y aquella que envía la información de login al usuario. Todo esto lo vemos en la columna Line 90%.

Para el caso en el que tengamos 220 usuarios hemos obtenido los siguientes resultados:



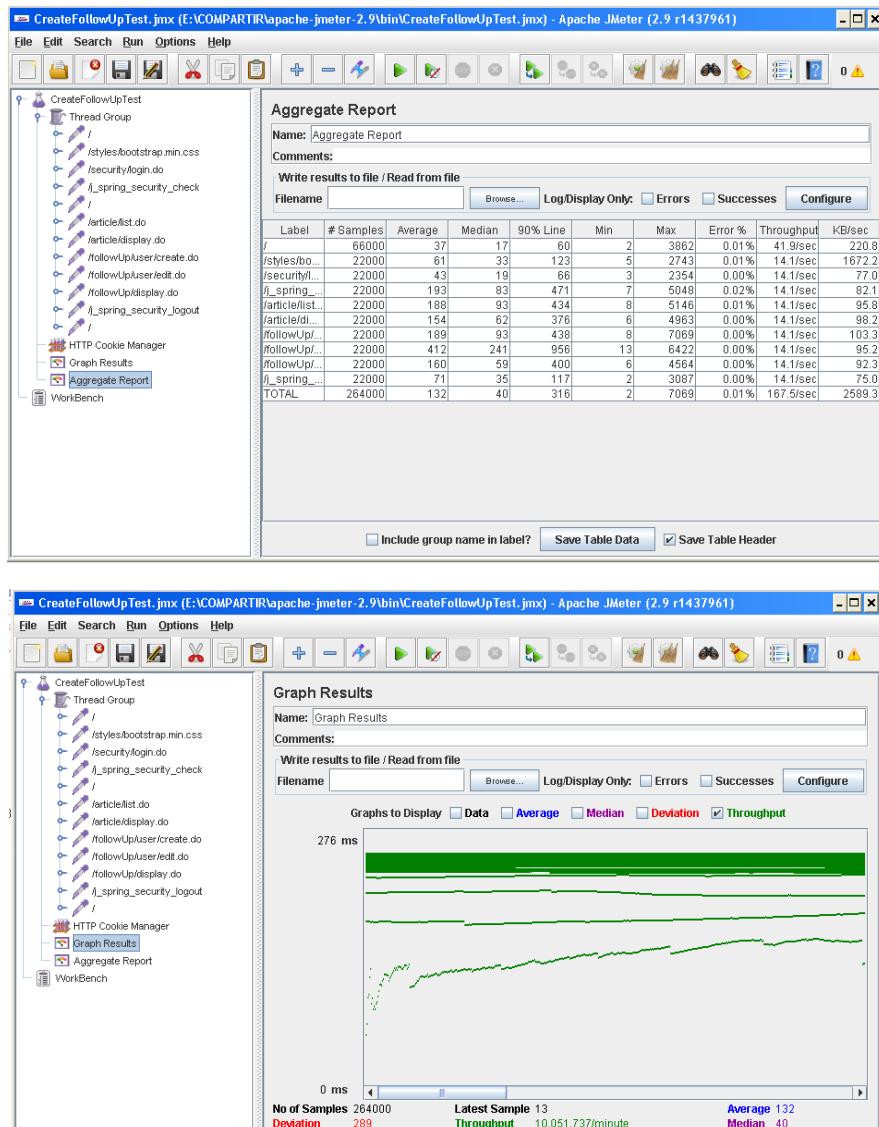
Caso de uso de crear un follow-up.

Se autentica un usuario que tenga un artículo publicado en un periódico, accede a sus artículos y escribe un follow-up, des autenticándose más tarde.

En este caso de uso, encontramos errores cuando coinciden 220 usuarios concurrentes realizando cada uno el caso de uso unas 100 veces. Estos errores como observamos en las imágenes son mínimos y vienen dados por el rendimiento de la máquina, en este caso la CPU.

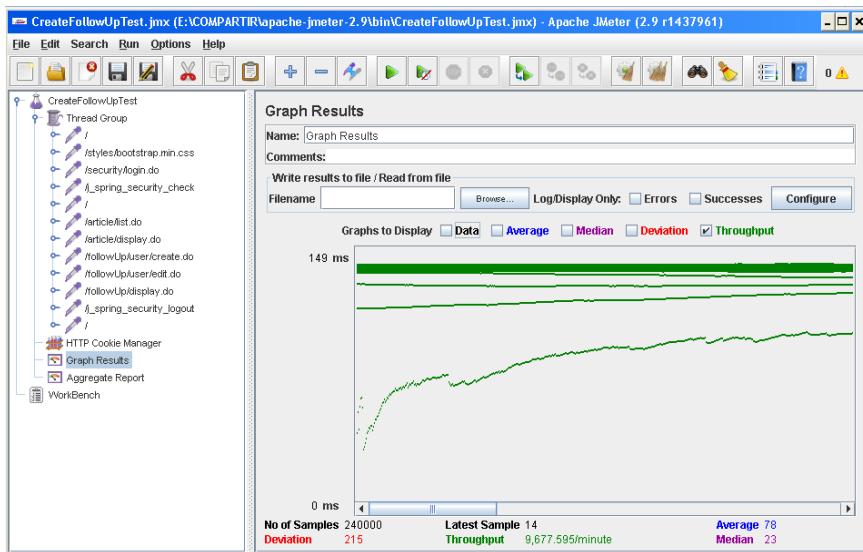
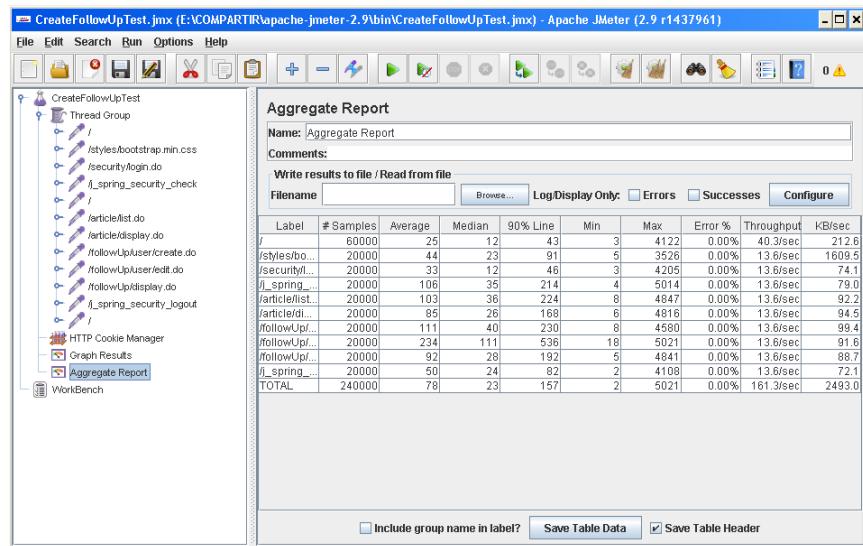
Como se observa en la figura la probabilidad de error es mínima así que podemos concretar que, para este caso de uso, llegamos al límite con dicho número de usuarios.

Teniendo en cuenta la columna 90% line, podemos concretar que la acción que más tiempo lleva para ejecutarse en el sistema es la que se produce a la hora de guardar el follow-up.



Como caso positivo cercano al límite mostrado anteriormente, hemos evaluado 200 usuarios, realizando el caso 100 veces y todos ellos al mismo tiempo.

Analizando un poco las figuras siguientes podemos concluir que nuevamente la acción que más tiempo suele llevar es aquella en la que se guarda el follow-up.



Caso de uso dashboard.

Nos autenticamos como administrador, vemos el dashboard, navegamos por él, y nos des autenticamos.

En este caso, solamente vamos a mostrar un ejemplo de una cota bastante alta. En este ejemplo no se encuentran errores; sin embargo; los tiempos de ejecución no son admisibles. Para llegar a conseguir este ejemplo, hemos probado con 100 usuarios, los cuales realizan 100 veces el caso de uso a la misma vez.

En un caso real, no sería normal encontrar un número tan alto de administradores en la aplicación. No obstante, hemos considerado suficiente ver un valor alto para así tener una estimación bastante buena tanto de los errores como de los tiempos de ejecución. Aquí adjuntamos las imágenes correspondientes a la ejecución de los casos de uso.

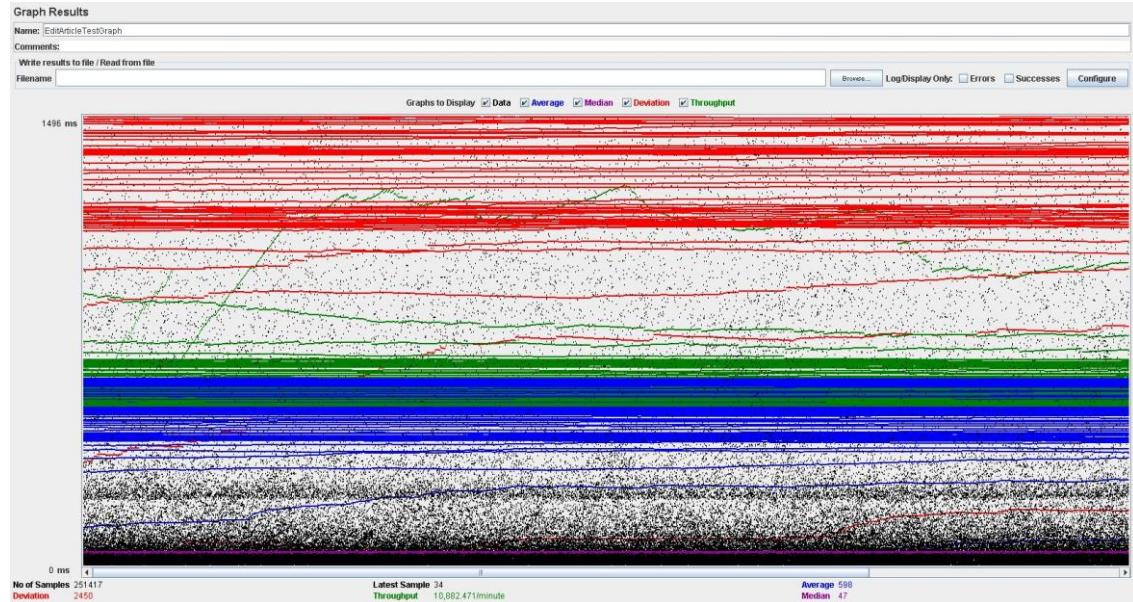
Caso de uso: editar un artículo.

En este caso de uso, accedemos a la página, nos logueamos como un usuario, accedemos a nuestros artículos, accedemos a la vista para editar, editamos algún parámetro, guardamos y nos deslogueamos.

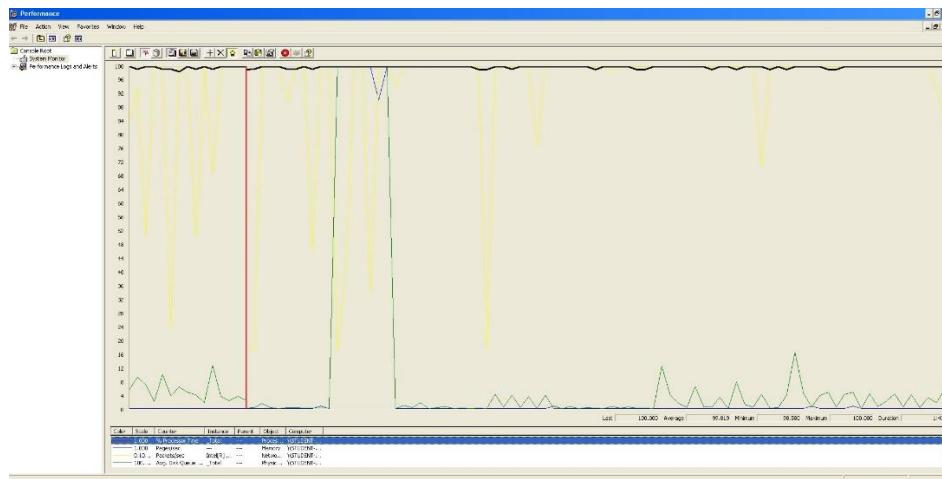
Hemos establecido el número de hilos máximo acorde con las prestaciones de nuestros ordenadores a 200 usuarios, 1 ramp-up y 100 loop count, para este caso de uso. Con esto, obtenemos las gráficas Aggregate Report y Graph Results:

Aggregate Report										
Name: EditArticleAggregate										
Comments:										
Write results to file / Read from file										
Filename	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec	
/	39819	51	28	93	2	6835	0.00%	28.0/sec	148.5	
/scripts/jquery.min.js	13272	87	51	183	1	5125	0.00%	9.6/sec	911.4	
/style/style.css	13270	51	25	95	1	6291	0.00%	9.6/sec	14.4	
/scripts/min.css	13269	68	55	183	1	4597	0.00%	9.6/sec	115.6	
/scripts/cookieconsent.js	13268	52	29	107	1	4461	0.00%	9.6/sec	328.3	
/scripts/polyfill.js	13268	47	25	93	1	1996	0.00%	9.6/sec	71.6	
/scripts/bootstrap.min.js	13265	53	32	107	0	2730	0.00%	9.6/sec	348.8	
/scripts/cookieconsentmanager.css	13265	45	24	92	1	3006	0.00%	9.6/sec	39.1	
/scripts/fontawesome-all.min.js	13264	44	23	91	0	1980	0.00%	9.6/sec	35.3	
/app-content/uploads/2018/01/favicon.ico	13264	2599	257	3443	19	208030	1.18%	9.6/sec	1441.5	
/security/login.do	13259	54	28	104	1	4519	0.01%	9.6/sec	53.0	
/spring_security_check	13232	60	30	107	3	4686	0.00%	9.6/sec	51.7	
/article/user/edit.do	13230	1931	1689	5119	7	31718	0.02%	9.6/sec	247.7	
/article/user/edit.do	26319	2007	1694	5144	8	32737	0.02%	19.1/sec	128.7	
/spring_security_logout	13101	89	52	191	4	1548	0.01%	9.6/sec	49.2	
TOTAL	251417	598	47	1353	0	208030	0.07%	181.4/sec	4859.8	

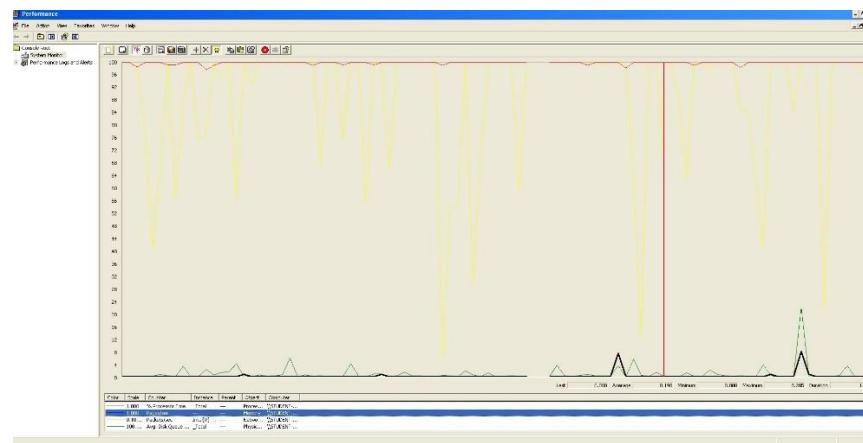
Como podemos observar fijándonos en 90% Line, las direcciones /article/list.do y /article/user/edit.do generan un tiempo de respuesta mayor a la media, 4983 ms y 5144 ms respectivamente.



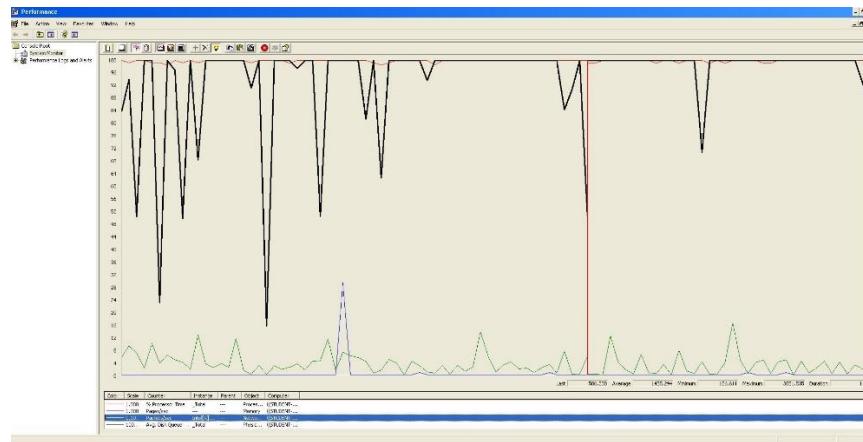
Al observar errores, revisamos performance.exe, obteniendo lo siguiente:



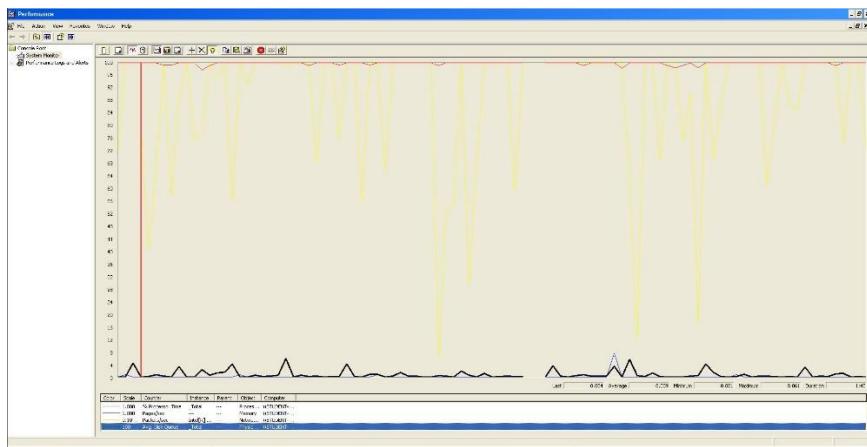
Observamos que podemos estar ante un problema de CPU, ya que se esta utilizando un alto porcentaje de la capacidad del procesador.



No hay indicios de que se estén produciendo fallos de memoria, los porcentajes no alcanzan el 10%.



Ante unos porcentajes tan altos, es posible que el problema sea de la tarjeta de red.



El disco no es el problema, puesto que los valores no alcanzan el 5%.

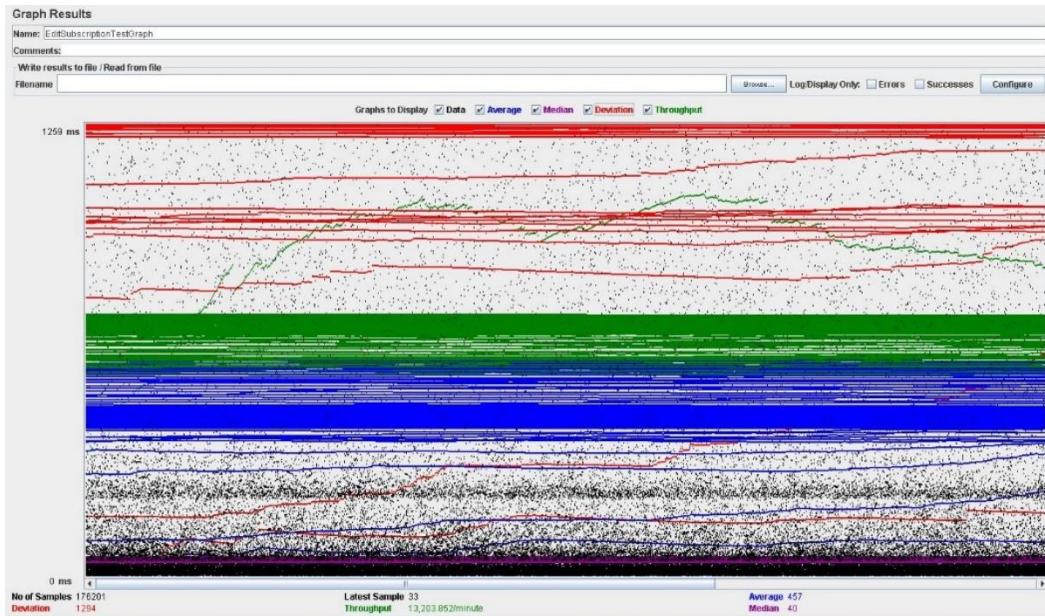
Caso de uso: editar suscripción.

En este caso de uso, accedemos a la página, nos logueamos como un cliente, accedemos a nuestras suscripciones, accedemos a la vista para editar, editamos algún parámetro, guardamos y nos deslogueamos.

Hemos establecido el número de hilos máximo acorde con las prestaciones de nuestros ordenadores a 200 usuarios, 1 ramp-up y 100 acciones cada uno, para este caso de uso. Con esto, obtenemos las gráficas Aggregate Report y Graph Results:

Aggregate Report										
Name:	EditSubscriptionAggregate									
Comments:										
Filename:	<input type="button" value="Browse..."/> Log Display Only <input type="checkbox"/> Errors <input type="checkbox"/> Successes <input type="checkbox"/> Configure									
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec	
/	26318	38	23	63	2	3784	0.00%	32.9/sec	170.6	
/scripts/jquery.min.js	8853	55	36	117	2	1000	0.00%	11.1/sec	1053.1	
/styles/bootstrap.min.css	8853	62	39	129	3	1095	0.00%	11.1/sec	1312.8	
/styles/style.css	8852	36	20	60	1	977	0.00%	11.1/sec	16.7	
/scripts/bootstrap.min.js	8852	40	24	78	1	990	0.00%	11.1/sec	403.3	
/scripts/cookiesconsent.js	8852	37	23	71	1	1130	0.00%	11.1/sec	379.4	
/scripts/helpers.js	8852	33	20	60	1	1020	0.00%	11.1/sec	35.5	
/scripts/helpers.cookieconsent.m...	8851	23	20	62	1	958	0.00%	11.1/sec	45.2	
/scripts/helpers.js	8851	32	19	63	1	1080	0.00%	11.1/sec	24.7	
/wp-content/uploads/20...	8851	264	238	318	176	9432	0.02%	11.1/sec	1686.3	
/favicon.ico	8846	41	22	81	1	1189	0.00%	11.1/sec	61.3	
/security/login.do	8845	36	22	64	3	1539	0.00%	11.1/sec	59.8	
/j_spring_security_check	8813	1372	627	3594	7	22096	0.01%	11.0/sec	62.9	
/subscription/customer/...	17505	1401	739	3521	9	23299	0.05%	22.0/sec	150.4	
/subscription/customer/...	17515	2081	1301	4989	7	28968	0.11%	22.0/sec	161.5	
/j_spring_security_logout	8852	73	48	137	4	1504	0.01%	11.0/sec	56.9	
TOTAL	176201	457	40	1330	1	28968	0.02%	220.1/sec	5717.9	

Como podemos observar fijándonos en 90% Line, las direcciones /subscription/customer/list.do y /subscription/customer/edit.do generan un tiempo de respuesta mayor a la media, 3521 ms y 4989 ms respectivamente.

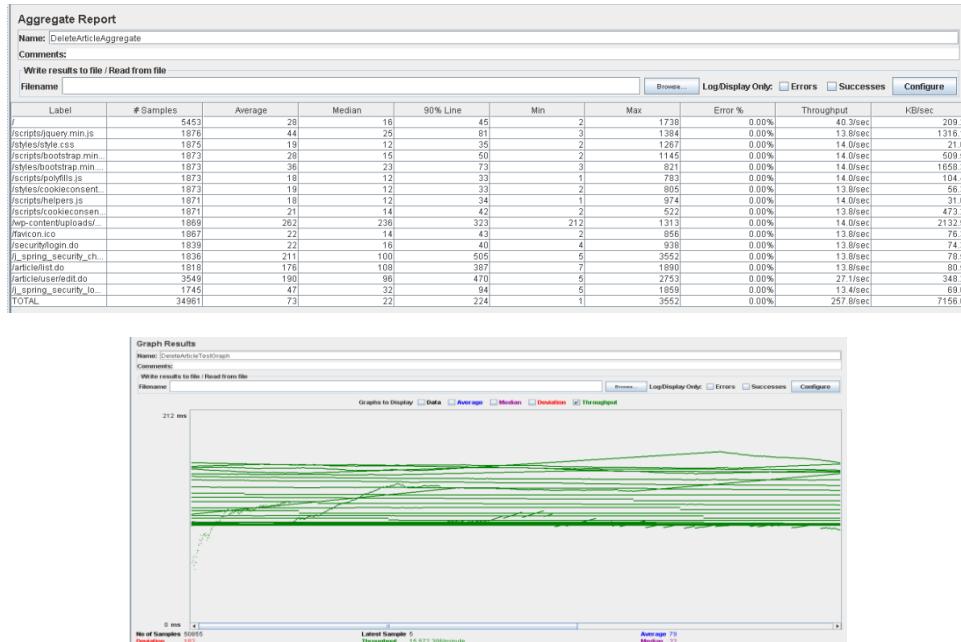


En esta ocasión, hemos realizado un caso de uso similar al anterior, con unos porcentajes de errores casi idénticos, con unas razones de estos también similares.

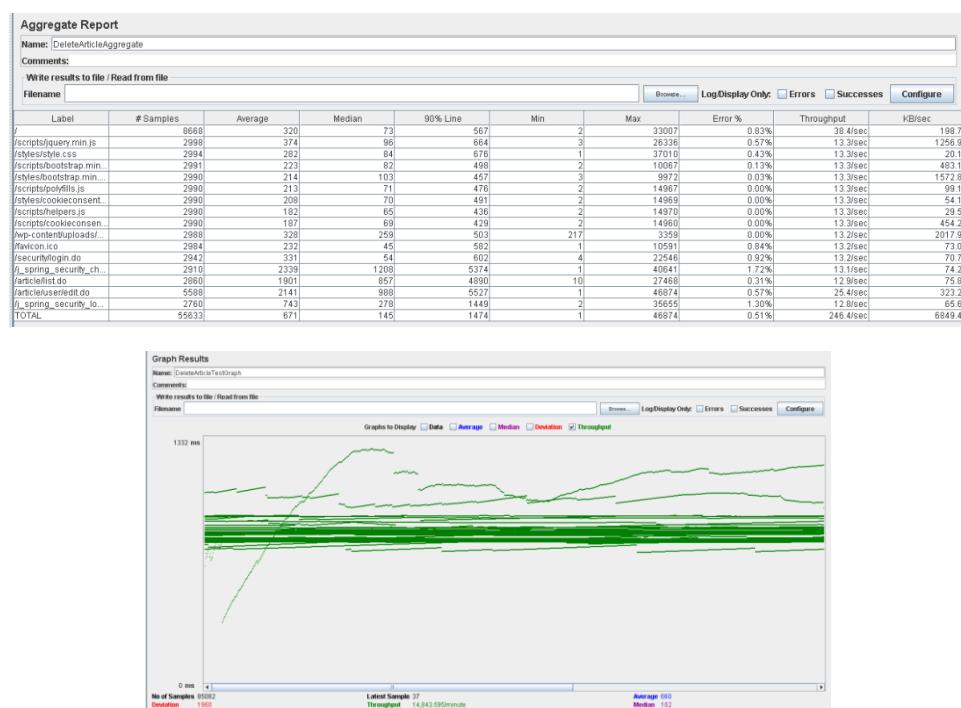
Caso de uso: borrar un artículo.

En este caso de uso, accedemos a la página, nos logueamos como un usuario, accedemos a nuestros artículos, accedemos a la vista para editar, borramos el artículo y nos deslogueamos.

La configuración en esta ocasión es de 150 usuarios, 1 ramp-up y 100 acciones cada uno, no produciéndose errores. Como observamos fijándonos en 90% Line, las direcciones /article/list.do y /article/user/edit.do generan un tiempo de respuesta mayor a la media, 387 ms y 470 ms respectivamente. Obtenemos las gráficas:



Ahora, la configuración es de 300 usuarios, 1 ramp-up y 200 acciones cada uno, produciéndose pequeños porcentajes de errores. El número de usuarios permitidos está entre 150 y 300.



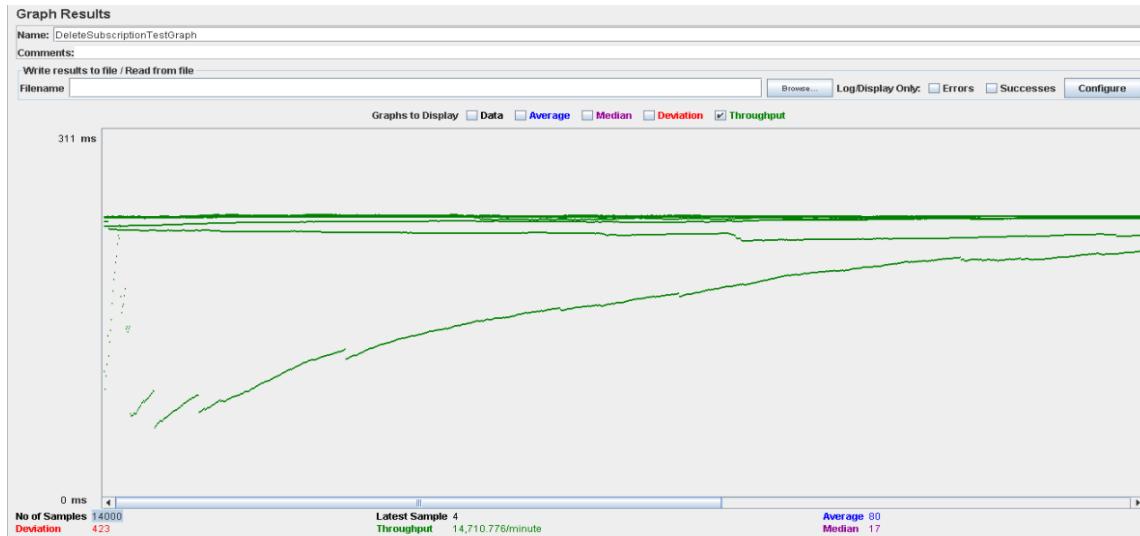
Caso de uso: borrar una suscripción.

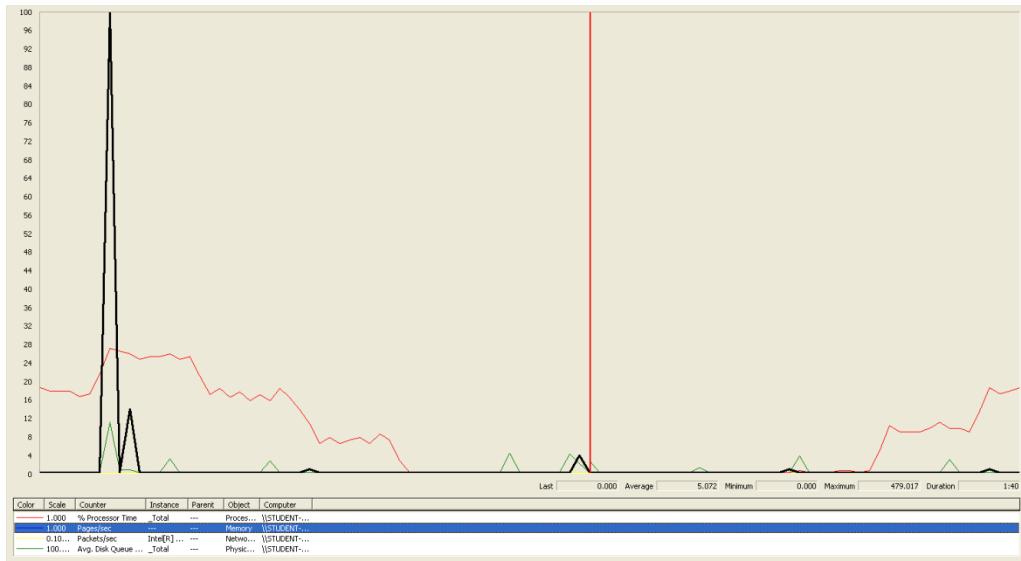
En este caso de uso, accedemos a la página, nos logueamos como un usuario, accedemos a nuestros artículos, accedemos a la vista para editar, borramos el artículo y nos deslogueamos.

Hemos establecido el número de hilos máximo acorde con las prestaciones de nuestros ordenadores a 1000 usuarios, 1 ramp-up y 350 loop count, para este caso de uso. Con esto, obtenemos las gráficas Aggregate Report y Graph Results:

Aggregate Report										
Name: DeleteSubscriptionAggregate										
Comments:										
Write results to file / Read from file										
Filename	Browse...	Log/Display Only:	<input type="checkbox"/> Errors	<input type="checkbox"/> Successes	<input checked="" type="checkbox"/> Configure					
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec	
/	2000	12	7	23	1	311	0.00%	36.7/sec	194.7	
/scripts/jquery.min.js	1000	17	10	34	2	368	0.00%	20.1/sec	1910.3	
/styles/bootstrap.min.css	1000	18	11	38	2	280	0.00%	20.1/sec	2380.5	
/scripts/bootstrap.min.js	1000	13	7	24	1	252	0.00%	20.1/sec	731.2	
/scripts/helpers.js	1000	11	6	19	1	309	0.00%	20.1/sec	44.8	
/scripts/polyfills.js	1000	9	6	19	1	249	0.00%	20.0/sec	149.9	
/scripts/cookieconsent.min.css	1000	8	6	16	1	145	0.00%	20.0/sec	81.9	
/styles/style.css	1000	9	6	16	1	267	0.00%	20.0/sec	30.3	
/scripts/cookieconsent.js	1000	10	7	17	1	157	0.00%	20.0/sec	606.6	
/wp-content/uploads/2018/02/ne...	1000	484	480	515	399	1541	0.00%	19.8/sec	3022.7	
/favicon.ico	1000	10	7	18	1	283	0.00%	20.0/sec	110.9	
/security/login.do	1000	13	10	23	2	106	0.00%	19.0/sec	102.6	
/j_spring_security_check	1000	42	31	75	5	2599	0.00%	19.0/sec	108.3	
/subscription/customer/list.do	1000	15	13	25	6	62	0.00%	108.4/sec	690.2	
TOTAL	15000	46	9	57	1	2599	0.00%	67.8/sec	2194.2	

Como observamos fijándonos en 90% Line, la dirección /wp-content/uploads/2018/02/news genera un tiempo de respuesta mayor a la media, 515 ms, debido a que se trata de una imagen.



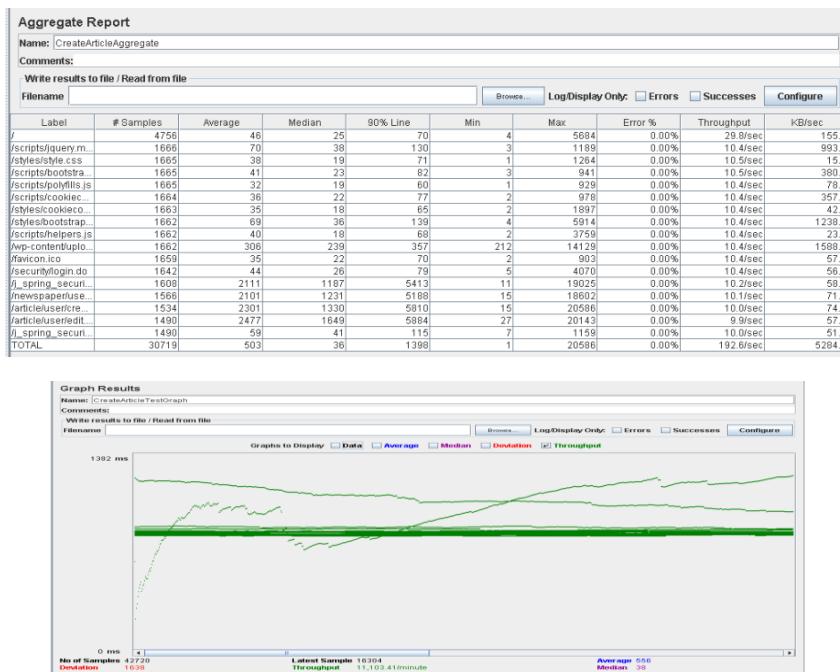


Ante la configuración de hilos escogida, abrimos performance.exe y vemos que la tarjeta de red puede fallar (por los picos encontrados).

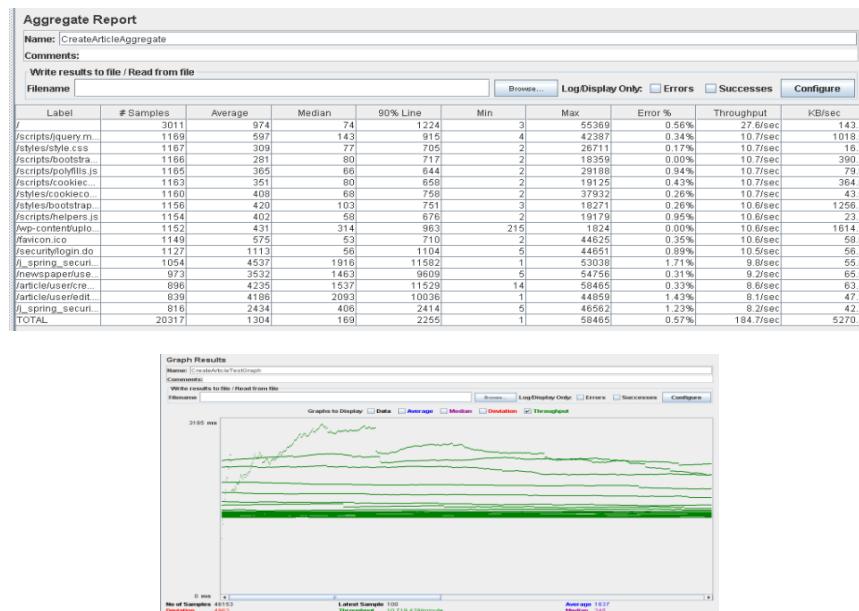
Caso de uso: crear artículo.

En este caso de uso, accedemos a la página, nos logueamos como un cliente, accedemos a la lista de periódicos, seleccionamos crear artículo, rellenamos el formulario, guardamos y nos deslogueamos.

La configuración en esta ocasión es de 200 usuarios, 1 ramp-up y 100 acciones cada uno, no produciéndose errores, que podemos ver en el Aggregate Report y en el Graph Results. Como observamos fijándonos en 90% Line, las direcciones /newspaper/user/list.do, /article/user/create.do y /article/user/edit.do, generan un tiempo de respuesta mayor a la media, 5188 ms, 5810 ms y 5884 ms respectivamente.



Ahora, la configuración es de 400 usuarios, 1 ramp-up y 200 acciones cada uno, produciéndose pequeños porcentajes de errores. El número de usuarios permitidos está entre 200 y 400.



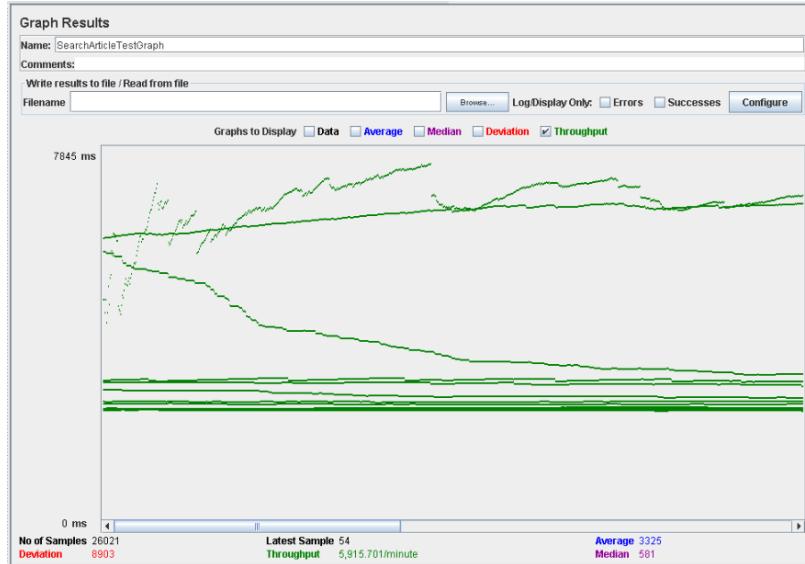
Caso de uso: buscar artículos.

En este caso de uso, accedemos a la página, nos logueamos como un usuario, accedemos a la lista de artículos, buscamos por un término clave, pasamos a la vista con los resultados y nos deslogueamos.

La configuración en esta ocasión es de 400 usuarios, 1 ramp-up y 200 acciones cada uno, produciéndose leves errores, que podemos ver en el Aggregate Report y en el Graph Results:

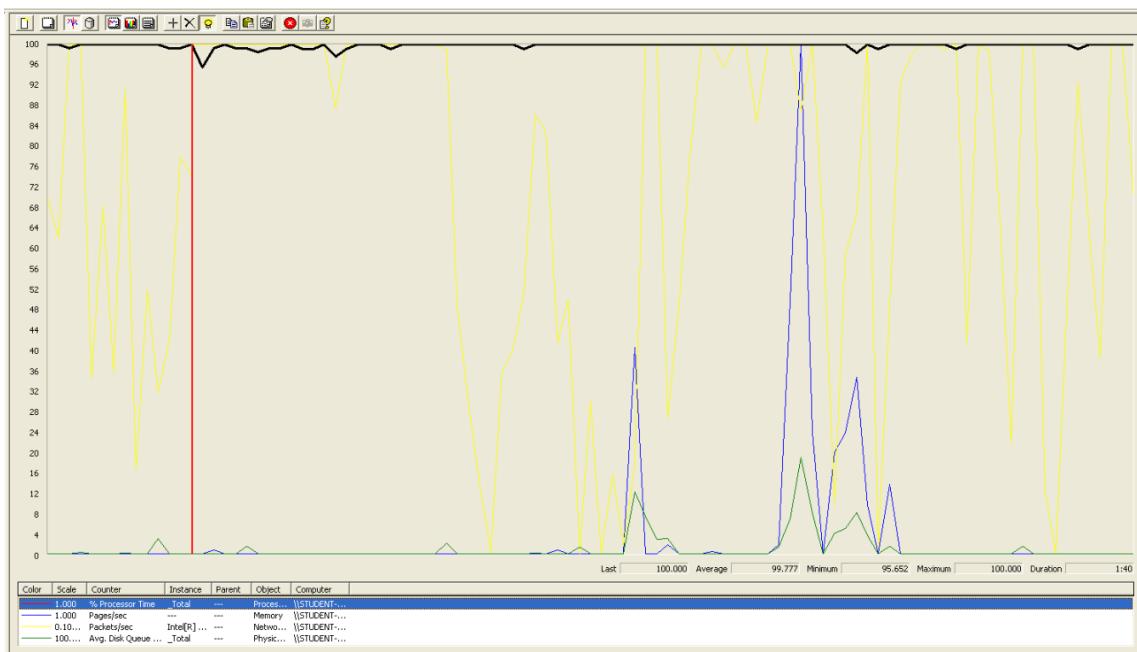
Aggregate Report										
Name: [SearchArticleAggregate]										
Comments:										
Write results to file / Read from file										
Filename	Browse...	Log Display Only	<input type="checkbox"/> Errors	<input type="checkbox"/> Successes	<input checked="" type="checkbox"/> Configure					
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec	
/	3779	1468	506	3253	3	30749	1.85%	16.0/sec	82.4	
/scripts/jquery...	1441	918	407	1858	2	27790	3.40%	6.1/sec	560.2	
/styles/bootstrap...	1428	1029	327	1677	2	31703	1.19%	6.0/sec	709.7	
/styles/style.css	1418	754	274	1587	1	25229	1.34%	6.0/sec	9.1	
/styles/cookie...	1402	608	190	1476	1	24480	0.57%	5.9/sec	24.2	
/scripts/cookies...	1392	535	146	1349	2	30301	0.43%	5.9/sec	201.0	
/scripts/helpers...	1384	664	141	1408	1	23603	0.51%	5.9/sec	13.1	
/scripts/bootstrap...	1375	570	142	1425	1	29419	0.51%	5.8/sec	210.9	
/scripts/sprintfills...	1367	558	141	1412	1	29132	0.29%	5.8/sec	43.2	
/wp-content/upl...	1300	16348	3021	44430	285	193649	1.47%	5.8/sec	867.5	
/index.php	1253	2003	384	5040	1	377541	0.30%	5.8/sec	31.4	
/search.php?in_d...	1344	1420	382	2651	4	27919	2.60%	5.8/sec	30.6	
/article/_secu...	1284	10111	4891	27722	2	98149	3.74%	5.5/sec	30.8	
/article/int.do	1194	9728	6062	23426	30	67206	1.84%	5.1/sec	3997.8	
/article/listBear...	1112	9563	5728	25283	4	75945	2.25%	4.8/sec	3789.7	
/article/_secu...	1093	2972	1719	7498	5	29621	3.39%	4.8/sec	24.2	
TOTAL	23726	322	574	9042	1	193649	1.72%	100.3/sec	10340.8	

Como podemos observar fijándonos en 90% Line, las direcciones /article/list.do y /article/listSearch.do generan un tiempo de respuesta mayor a la media, 23426 ms y 25283 ms respectivamente.

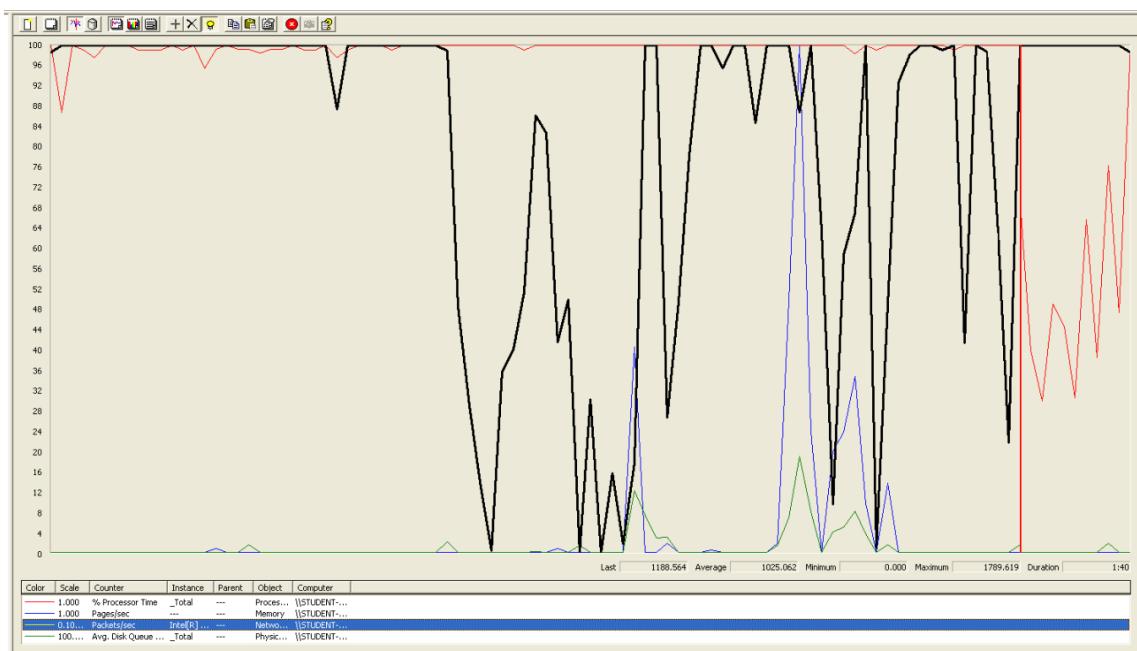


Ahora estudiaremos a que se deben los errores:

Abrimos performance.exe y vemos que tanto la tarjeta de red como la CPU tienen porcentajes muy altos, llegando al 100% en ocasiones, por lo que el problema se localizará al menos en una de las dos.



Performance con Processor Time.

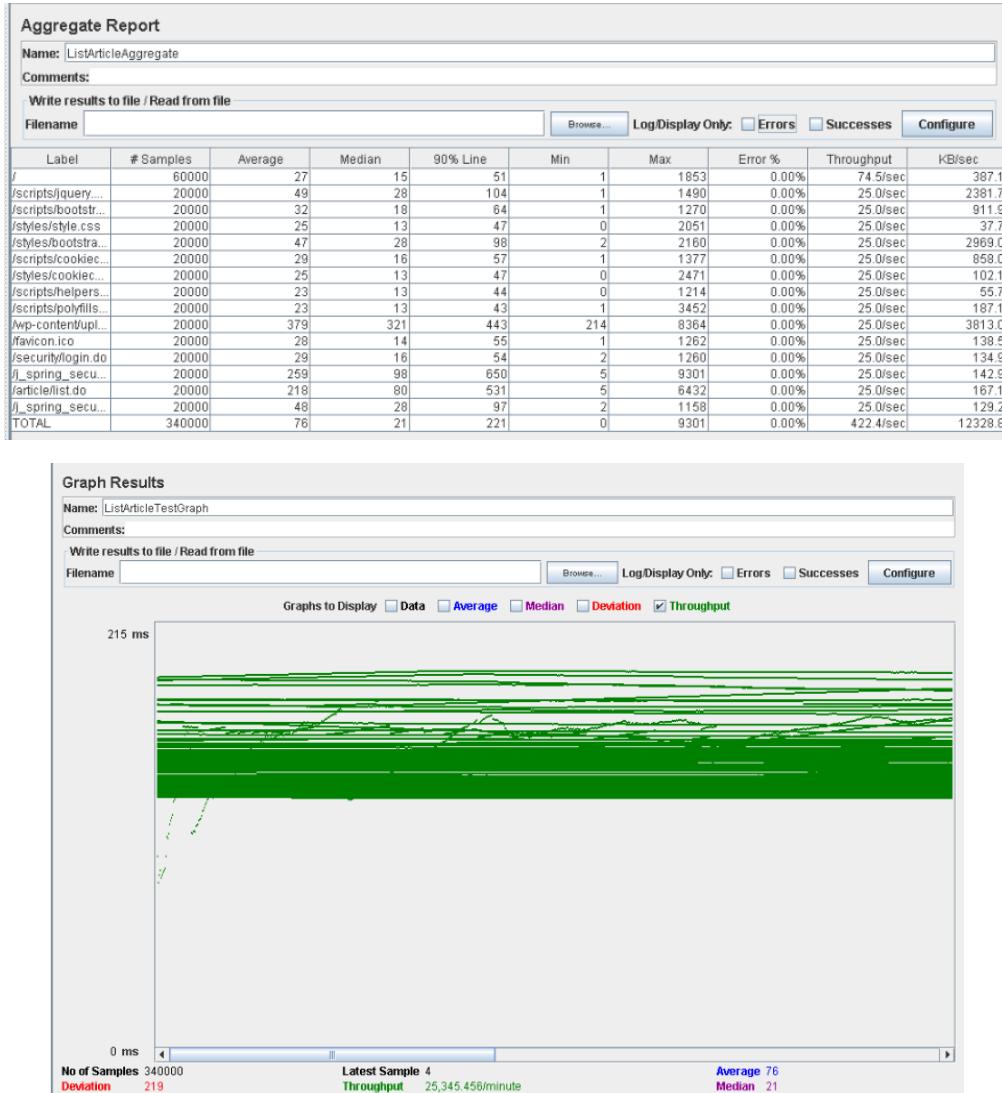


Performance con Packets/sec.

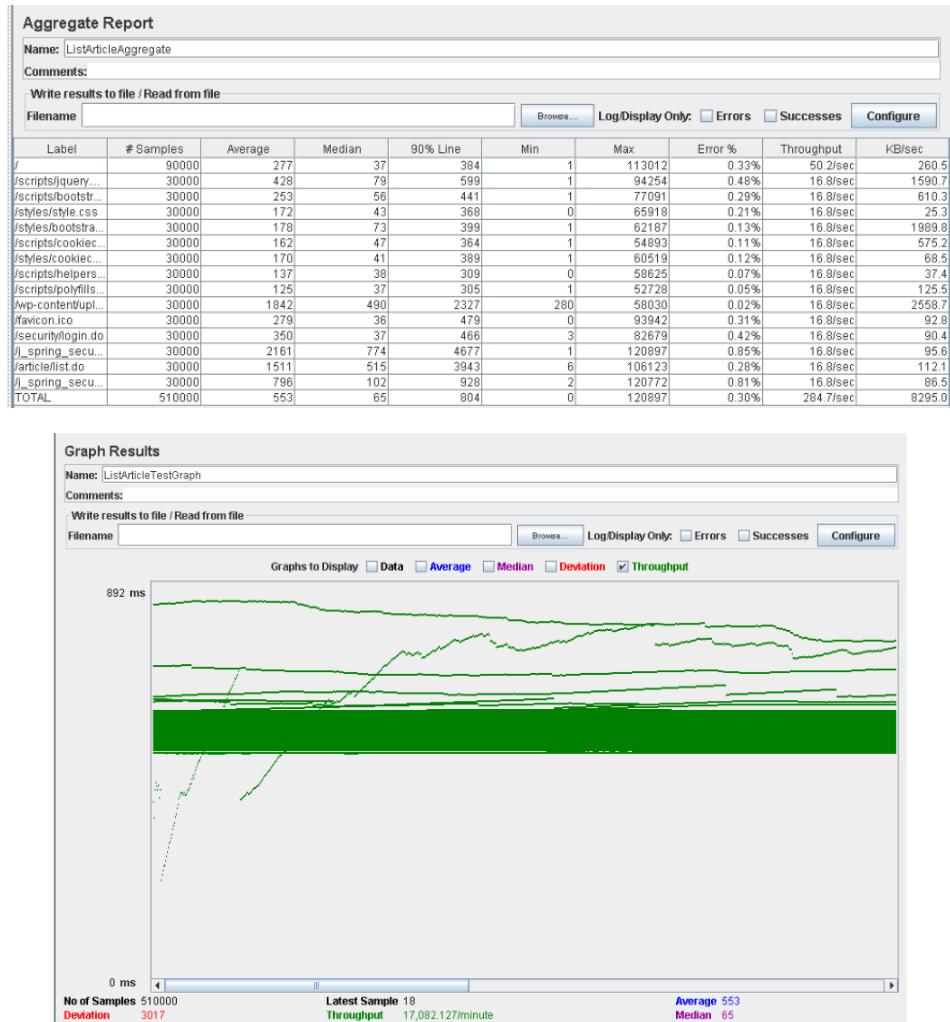
Caso de uso: listar artículos.

En este caso de uso, accedemos a la página, nos logueamos como un usuario, accedemos a la lista de artículos de un usuario y nos deslogueamos.

La configuración en esta ocasión es de 200 usuarios, 1 ramp-up y 100 acciones cada uno, no produciéndose errores, que podemos ver en el Aggregate Report y en el Graph Results:



Ahora probamos una configuración de 300 usuarios, 1 ramp-up y 100 acciones cada uno, obteniéndose ahora errores, por lo que entre 200 y 300 son los usuarios permitidos. Las gráficas correspondientes a este test son:

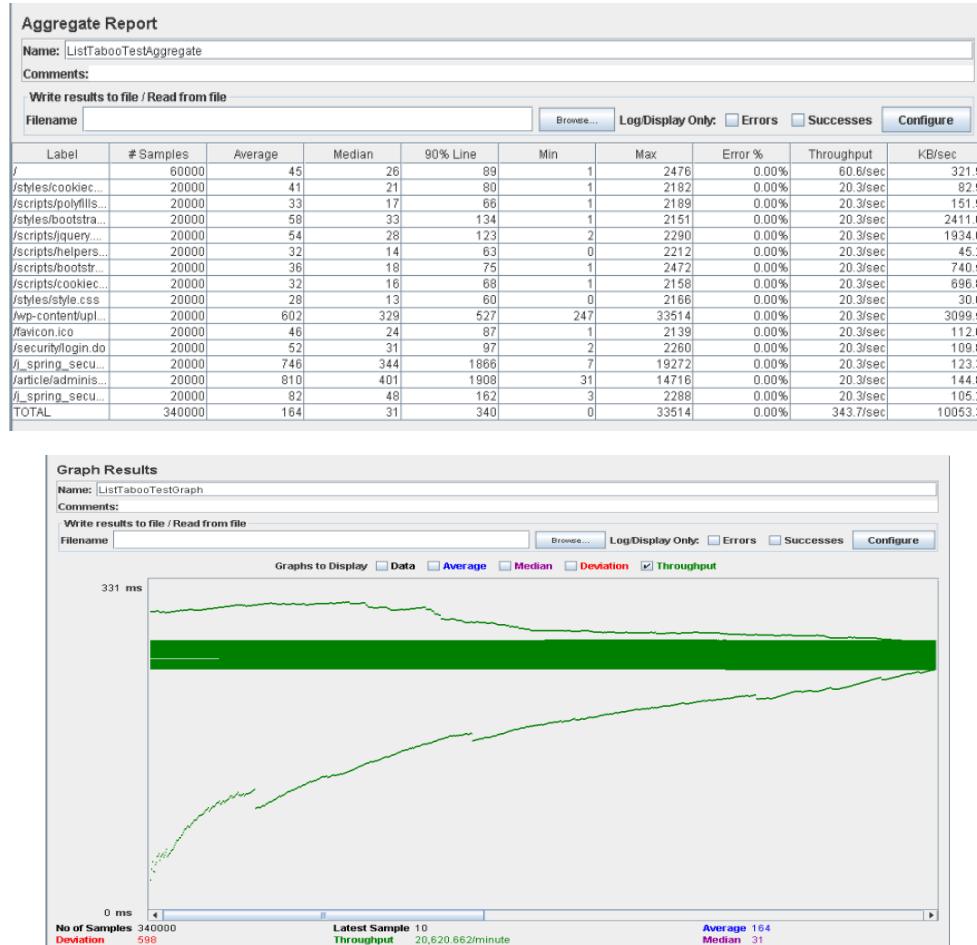


Como podemos observar fijándonos en 90% Line, la dirección /article/list.do genera un tiempo de respuesta mayor a la media, 3943 ms.

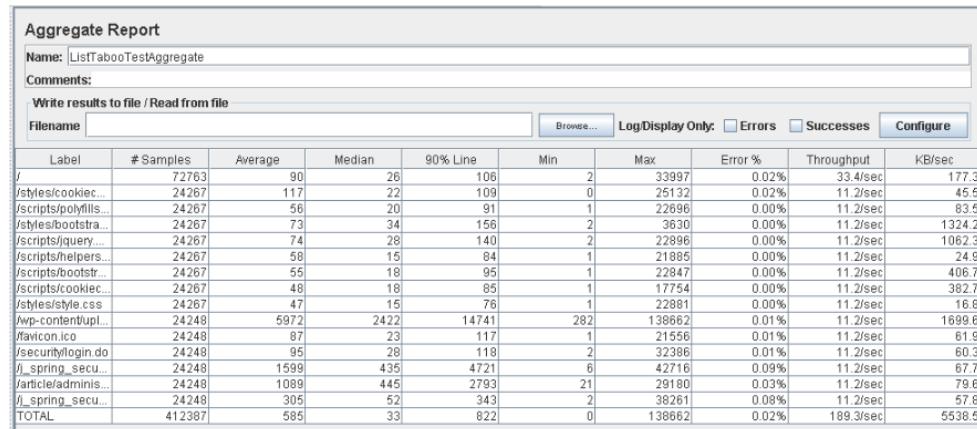
Caso de uso: listar artículos tabús.

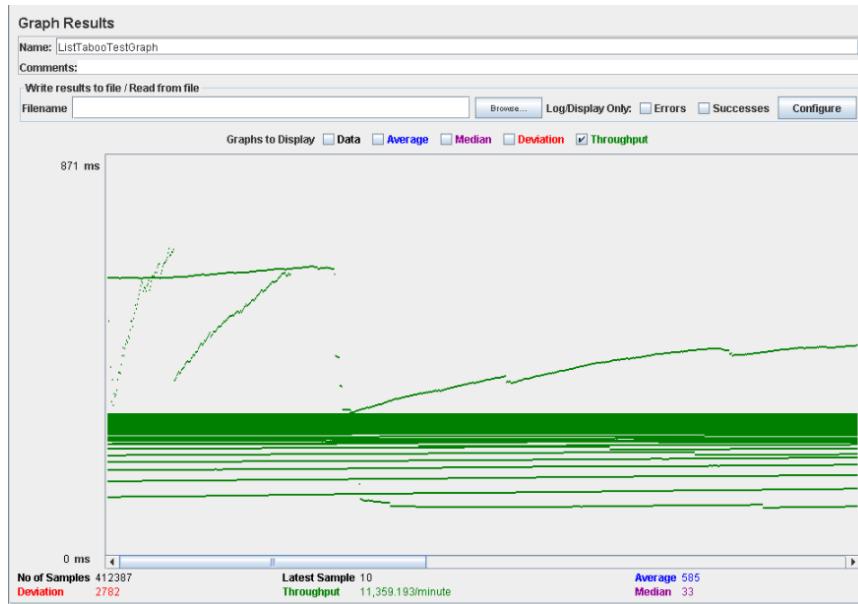
En este caso de uso, accedemos a la página, nos logueamos como un administrador, accedemos a la lista de artículos tabús y nos deslogueamos.

La configuración en esta ocasión es de 200 usuarios, 1 ramp-up y 100 acciones cada uno, no produciéndose errores, que podemos ver en el Aggregate Report y en el Graph Results:



Ahora probamos una configuración de 250 usuarios, 1 ramp-up y 100 acciones cada uno, obteniéndose ahora errores, por lo que entre 200 y 250 son los usuarios permitidos. Las gráficas correspondientes a este test son:



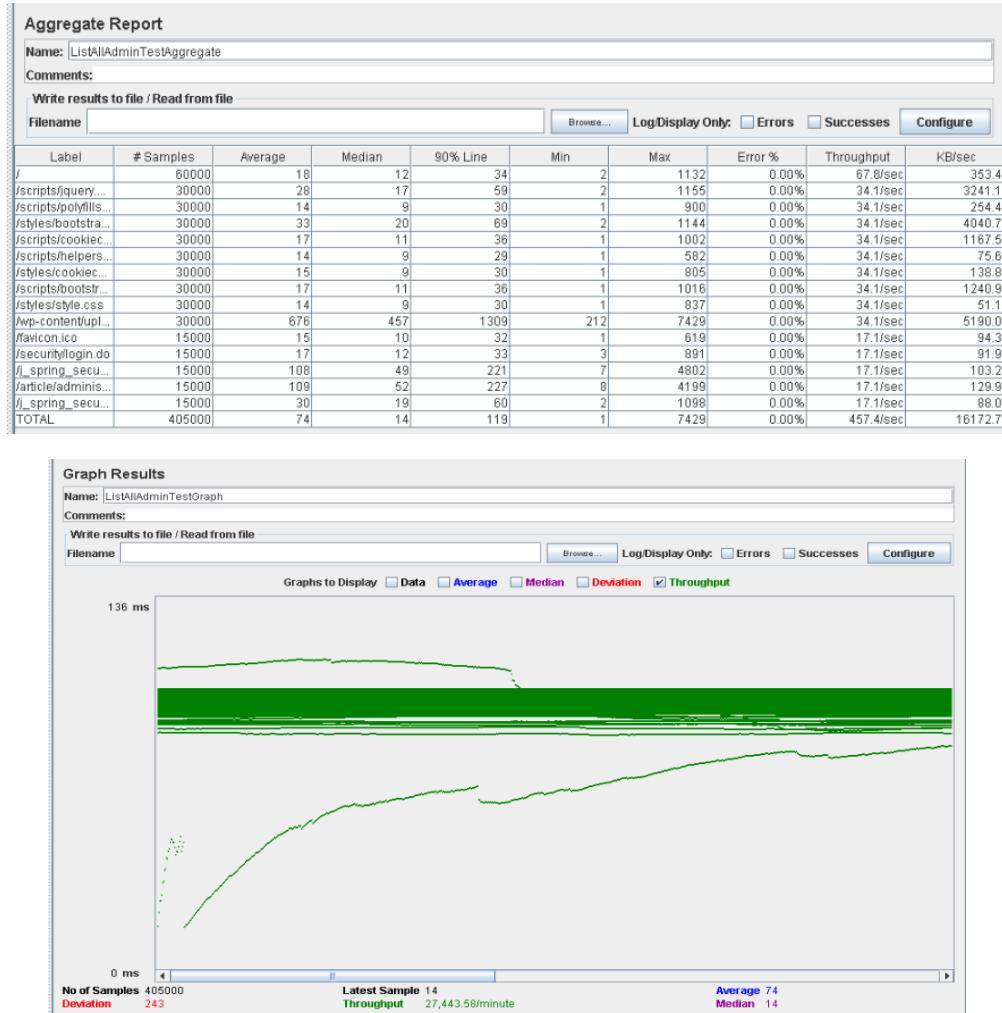


Como podemos observar fijándonos en 90% Line, la dirección [/article/administrator/listTaboo.do](#) genera un tiempo de respuesta mayor a la media, 2793 ms.

Caso de uso: listar todos los artículos administrador.

En este caso de uso, accedemos a la página, nos logueamos como un administrador, accedemos a la lista de artículos completa de un administrador y nos deslogueamos.

La configuración en esta ocasión es de 150 usuarios, 1 ramp-up y 100 acciones cada uno, no produciéndose errores, que podemos ver en el Aggregate Report y en el Graph Results:



Ahora probamos una configuración de 300 usuarios, 1 ramp-up y 100 acciones cada uno, obteniéndose ahora errores, por lo que entre 150 y 300 son los usuarios permitidos. Las gráficas correspondientes a este test son:

Aggregate Report										
Name: <input type="text" value="ListAllAdminTestAggregate"/> Comments: Write results to file / Read from file Filename <input type="text"/> Browse... Log/Display Only <input type="checkbox"/> Errors <input type="checkbox"/> Successes <input type="button" value="Configure"/>										
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec	
/	120000	242	35	412	2	81667	0.18%	67.6/sec	352.5	
/scripts/jquery....	60000	222	61	379	2	76533	0.13%	33.9/sec	3221.4	
/scripts/polyfills....	60000	183	35	383	1	81334	0.12%	33.9/sec	253.1	
/styles/bootstrap....	60000	253	67	414	2	78335	0.11%	33.9/sec	4017.0	
/scripts/cookiec....	60000	139	39	278	1	74724	0.08%	33.9/sec	1161.2	
/scripts/helpers....	60000	137	34	322	1	51758	0.06%	33.9/sec	75.4	
/styles/cookiec....	60000	186	35	377	1	58007	0.10%	33.9/sec	138.2	
/scripts/bootstrapstr....	60000	144	42	320	2	55751	0.08%	33.9/sec	1234.2	
/styles/style.css	60000	136	34	312	1	72825	0.08%	33.9/sec	51.0	
/wp-content/upl....	60000	587	445	805	222	22690	0.00%	33.9/sec	5165.5	
/favicon.ico	30000	351	35	484	1	81212	0.15%	17.0/sec	93.7	
/security/login.do	30000	281	30	457	2	81266	0.16%	17.0/sec	91.4	
/j_spring_secu...	30000	1181	475	1987	0	81789	0.56%	17.0/sec	102.5	
/article/administr...	30000	775	296	1686	3	75437	0.23%	17.0/sec	128.8	
/j_spring_secu...	30000	652	90	766	3	72240	0.33%	17.0/sec	87.5	
TOTAL	810000	303	52	555	0	81789	0.14%	456.4/sec	16126.2	



Como podemos observar fijándonos en 90% Line, la dirección /article/administrator/list.do genera un tiempo de respuesta mayor a la media, 1686 ms.

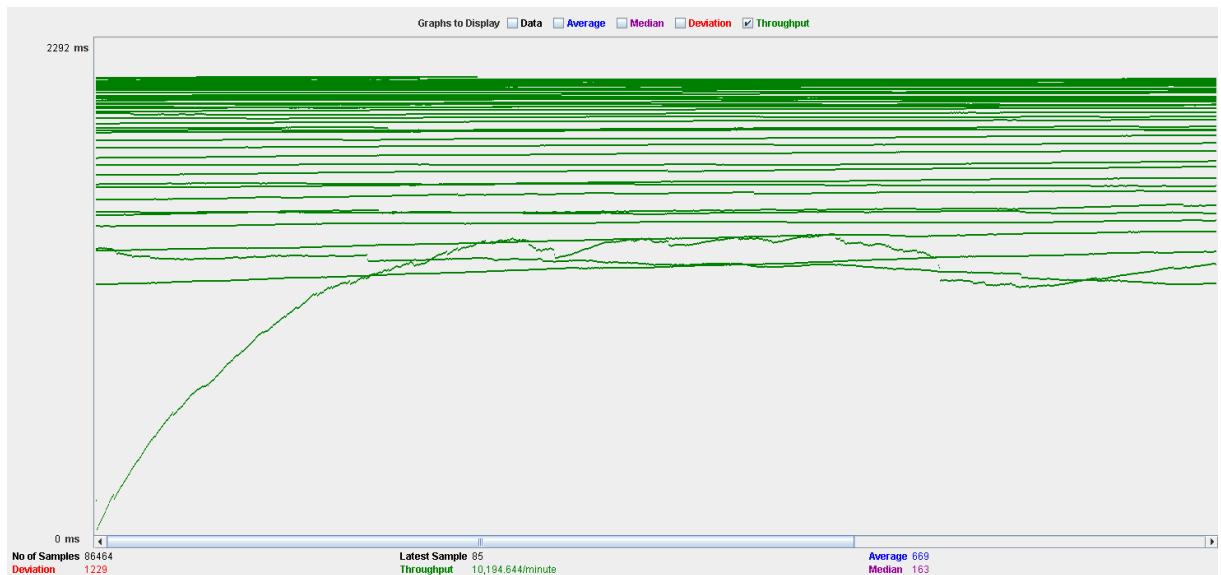
Caso de uso: crear Newspaper.

Realizamos una prueba estableciendo como número de usuarios simultáneo 200 personas con un retraso entre usuario y usuario de medio segundo. Con esta configuración tras ejecutar el siguiente script:



Nos logueamos como user, nos vamos a nuestros newspapers, creamos uno y nos deslogueamos.

Tras tener esta secuencia de pasos ejecutamos los tests y se nos generan las siguientes gráficas:



Y se nos genera el siguiente informe:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	12850	400	83	1276	1	10153	2.22%	25.3/sec	129.7
/scripts/jquery.min.js	4377	382	127	1093	1	10302	2.15%	8.6/sec	805.2
/styles/cookieconsent.mi...	4376	290	53	928	1	9962	1.90%	8.6/sec	35.3
/scripts/polyfills.js	4374	338	58	1092	1	10091	1.58%	8.6/sec	64.2
/scripts/bootstrap.min.js	4374	314	77	935	1	10199	1.46%	8.6/sec	310.7
/scripts/helpers.js	4374	295	50	956	1	10031	1.30%	8.6/sec	19.7
/scripts/cookieconsent.js	4374	317	72	1018	1	7795	1.23%	8.6/sec	293.2
/styles/style.css	4374	289	47	931	1	10080	1.21%	8.6/sec	13.5
/styles/bootstrap.min.css	4371	362	117	1008	1	10306	1.17%	8.6/sec	1013.6
/wp-content/uploads/201...	4371	278	240	389	61	3424	0.11%	8.6/sec	1315.1
/favicon.ico	4364	318	37	1091	1	9340	0.53%	8.6/sec	48.1
/security/login.do	4363	357	41	1230	1	9927	0.39%	8.7/sec	47.2
/j_spring_security_check	4328	1734	1285	3759	1	20267	2.08%	8.6/sec	49.6
/newspaper/user/list.do	8497	1219	721	2824	0	14171	2.00%	17.1/sec	1260.4
/newspaper/user/create....	4262	1057	511	2671	1	21015	1.22%	8.6/sec	63.5
/newspaper/user/edit.do	4235	2697	2223	5432	1	19271	3.31%	8.6/sec	636.4
/j_spring_security_logout	4200	824	184	2323	1	10208	2.50%	8.6/sec	45.1
TOTAL	86464	669	163	1998	0	21015	1.63%	169.9/sec	6075.9

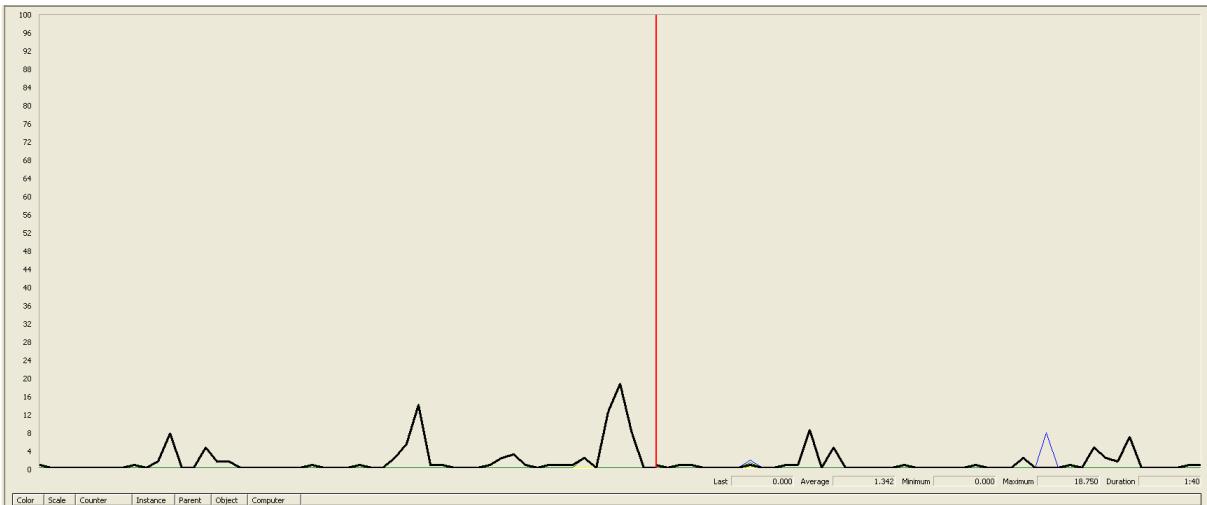
Como vemos se nos han generado errores pequeños con la configuración escogida por lo que el caso de uso de crear un newspaper es soportado de manera estable por un máximo de 200 personas. El causante del error que ha provocado esto ha sido el procesador de mi máquina. He llegado a esta conclusión ya que he mirado en los archivos de log y decían que el error era por un OutOfMemoryError.

```

tomcat7-stderr.2018-04-13.log - Notepad
File Edit Format View Help
Apr 13, 2018 1:47:25 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 40478 ms
Apr 13, 2018 4:29:52 AM org.apache.catalina.loader.WebappClassLoader clearReferencesJdbc
SEVERE: The web application [] registered the JDBC driver [com.mysql.jdbc.Driver] but failed to unregister it when t
Apr 13, 2018 4:29:53 AM org.apache.catalina.startup.HostConfig deleteRedeployResources
INFO: Undeploying context []
Apr 13, 2018 4:30:24 AM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\ROOT.war
java.lang.OutOfMemoryError: PermGen space
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:791)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:449)
at java.net.URLClassLoader.access$100(URLClassLoader.java:71)
at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
at java.lang.ClassLoader.loadClass(ClassLoader.java:423)
at java.lang.ClassLoader.loadClass(ClassLoader.java:356)
at org.eclipse.jdt.internal.compiler.parser.Parser.parse(Parser.java:9888)
at org.eclipse.jdt.internal.compiler.parser.Parser.parse(Parser.java:9868)
at org.eclipse.jdt.internal.compiler.parser.Parser.dietParse(Parser.java:8434)
at org.eclipse.jdt.internal.compiler.Compiler.internalBeginToCompile(Compiler.java:718)
at org.eclipse.jdt.internal.compiler.Compiler.beginToCompile(Compiler.java:383)
at org.eclipse.jdt.internal.compiler.Compiler.compile(Compiler.java:428)
at org.apache.jasper.compiler.JDTCompiler.generateClass(JDTCompiler.java:458)
at org.apache.jasper.compiler.Compiler.compile(Compiler.java:378)
at org.apache.jasper.compiler.Compiler.compile(Compiler.java:353)
at org.apache.jasper.compiler.Compiler.compile(Compiler.java:340)
at org.apache.jasper.JspCompilationContext.compile(JspCompilationContext.java:646)
at org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:357)
at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:390)
at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:334)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:728)

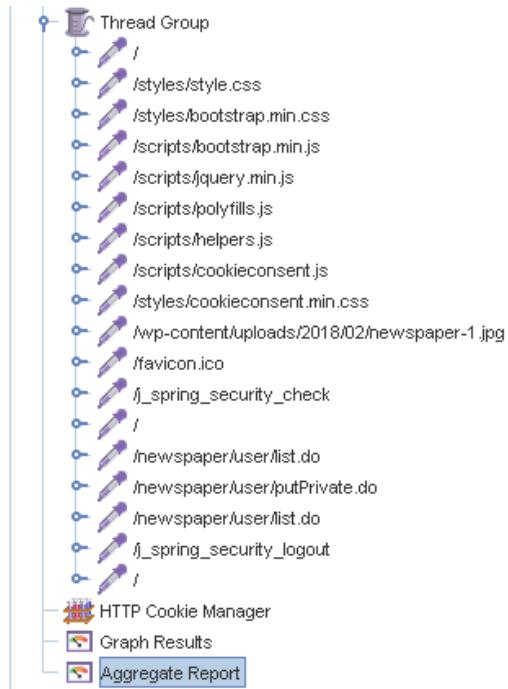
```

Así que lo que hacemos es observar los componentes de mi PC para ver que podría estar provocando un fallo, llegando a la conclusión de que se trata del procesador, ya que la gráfica asociada a esta tiene muchos picos y al ser menos cercana al suelo es menos buena.



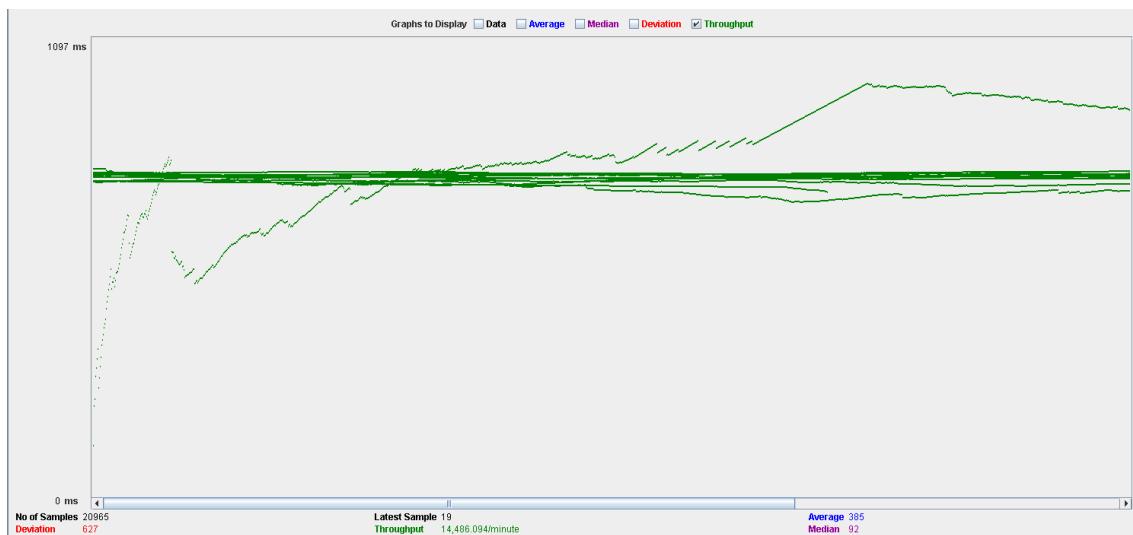
Caso de uso: establecer newspaper a privado.

Realizamos una prueba estableciendo como número de usuarios simultáneo 250 personas con un tipo de apertura entre persona y persona de 1/250. Con esta configuración tras ejecutar el siguiente script:



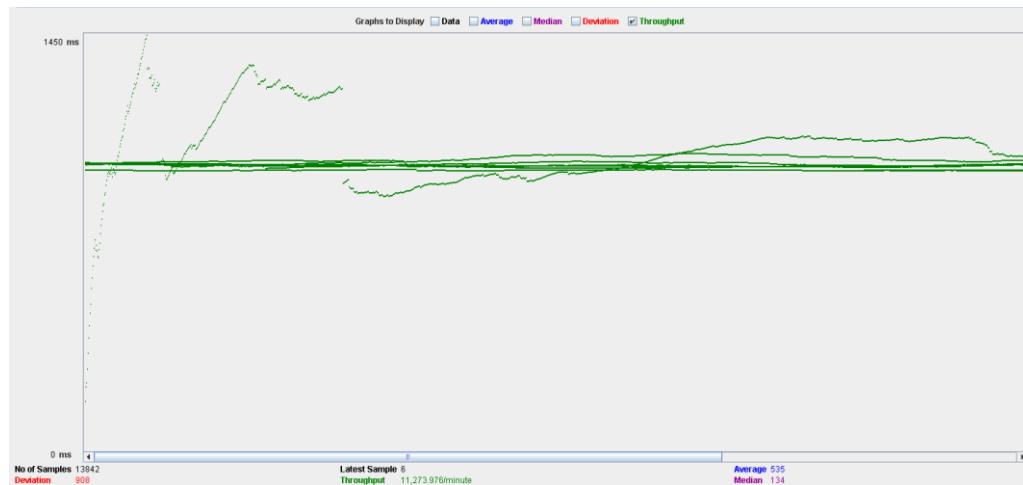
Nos logueamos como user, nos vamos a nuestros newspapers, ponemos uno público en privado y nos deslogueamos.

Tras tener esta secuencia de pasos ejecutamos los tests y se nos generan las siguientes gráficas:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	3348	313	53	1019	0	3595	7.77%	39.3/sec	194.3
/styles/style.css	1231	242	40	802	0	3391	7.15%	14.5/sec	22.6
/styles/bootstrap.min.css	1227	301	93	902	1	3857	7.01%	14.4/sec	1592.8
/scripts/bootstrap.min.js	1226	279	47	872	1	3642	8.33%	14.4/sec	484.1
/scripts/jquery.min.js	1222	312	64	962	0	3747	8.27%	14.4/sec	1258.1
/scripts/polyfills.js	1219	239	29	761	1	3703	9.19%	14.3/sec	100.1
/scripts/helpers.js	1217	214	28	733	0	3709	9.70%	14.3/sec	31.6
/scripts/cookieconsent.js	1217	222	39	706	1	3468	8.55%	14.3/sec	451.5
/styles/cookieconsent.mli	1217	194	29	687	0	3359	7.64%	14.3/sec	56.3
/wp-content/uploads/201...	1210	557	509	683	425	1574	0.00%	14.2/sec	2157.1
/favicon.ico	1192	298	30	1096	1	3736	5.29%	14.1/sec	76.0
/_spring_security_check	1166	710	294	2050	1	5604	8.32%	14.0/sec	81.8
/newspaper/user/list.do	2169	668	393	1614	1	5443	5.76%	26.5/sec	3451.1
/newspaper/user/putPriv...	1079	560	224	1599	1	5562	3.15%	13.4/sec	173.0
/_spring_security_logout	1026	656	160	1863	1	5786	7.50%	13.1/sec	65.8
TOTAL	20965	385	92	1120	0	5786	6.96%	241.4/sec	9847.9

Como vemos se han producido bastantes errores por lo tanto el valor de usuarios máximos simultáneo que puede soportar la aplicación es menor. Por lo tanto, buscamos con un número menor y probamos con 200 usuarios simultáneos. Las gráficas que nos salen al hacer esto son las siguientes:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	2201	451	101	1609	1	6722	0.41%	30.4/sec	158.3
/styles/style.css	818	264	83	1015	2	3594	0.24%	11.3/sec	17.7
/styles/bootstrap.min.css	814	333	139	884	1	3826	0.25%	11.3/sec	1333.8
/scripts/bootstrap.min.js	813	309	84	1167	1	4756	0.25%	11.2/sec	409.3
/scripts/jquery.min.js	812	398	126	1285	1	4666	0.25%	11.2/sec	1066.1
/scripts/polyfills.js	811	307	48	1108	2	4655	0.25%	11.2/sec	84.3
/scripts/helpers.js	810	283	52	1147	2	5005	0.25%	11.2/sec	25.5
/scripts/cookieconsent.js	808	273	69	744	1	4642	0.25%	11.2/sec	382.4
/styles/cookieconsent.mli	808	258	51	947	1	3950	0.25%	11.2/sec	46.1
/wp-content/uploads/201...	809	526	493	616	427	1358	0.00%	11.1/sec	1691.0
/favicon.ico	799	318	33	1370	1	5483	0.25%	11.2/sec	62.5
/_spring_security_check	775	1384	804	3527	10	9187	0.65%	11.1/sec	64.6
/newspaper/user/list.do	1403	1121	642	2913	1	6938	0.71%	20.5/sec	2930.2
/newspaper/user/putPriv...	698	678	375	1731	1	7168	0.29%	10.5/sec	142.3
/_spring_security_logout	664	979	198	3064	1	5493	0.30%	10.3/sec	54.4
TOTAL	13842	535	134	1659	1	9187	0.33%	187.9/sec	8143.7

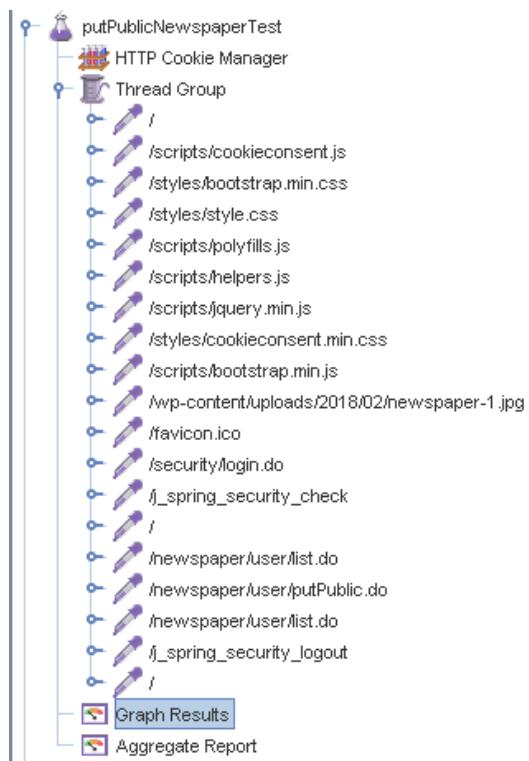
Como vemos el porcentaje del error es muy cercano a cero, por lo tanto 200 usuarios es un valor muy cercano al límite máximo que es capaz de soportar nuestra aplicación de forma simultánea para que se haga el caso de uso sin que haya ningún fallo. Cabe destacar por la columna de 90% Line que los pasos que más tiempo producen son aquellos realizados por el usuario donde se hace una petición Post.

Haciendo una comprobación con el performance vemos que se está produciendo un cuello de botella en la memoria del sistema debido al alto valor que produce esta al principio.



Caso de uso: establecer newspaper a público.

Realizamos una prueba estableciendo como número de usuarios simultáneo de 200 con un retraso de medio segundo entre usuario y usuario. Con esta configuración tras ejecutar el siguiente script:



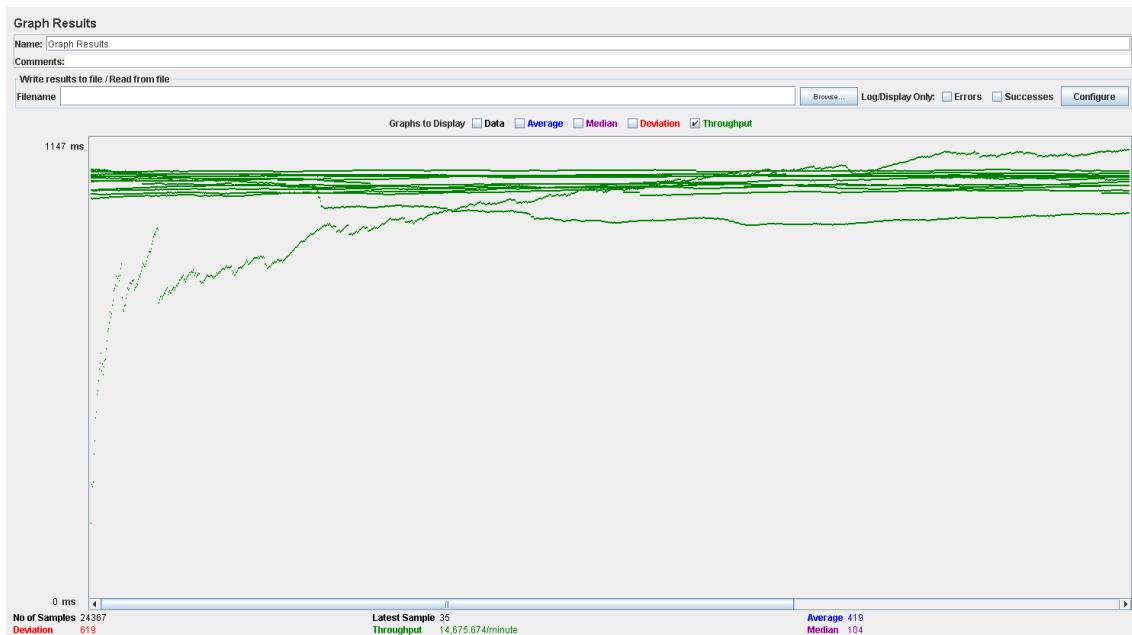
Nos logueamos como user, nos vamos a nuestros newspapers, ponemos uno privado en público y nos deslogueamos.

Tras tener esta secuencia de pasos ejecutamos los tests y se nos generan las siguientes gráficas:



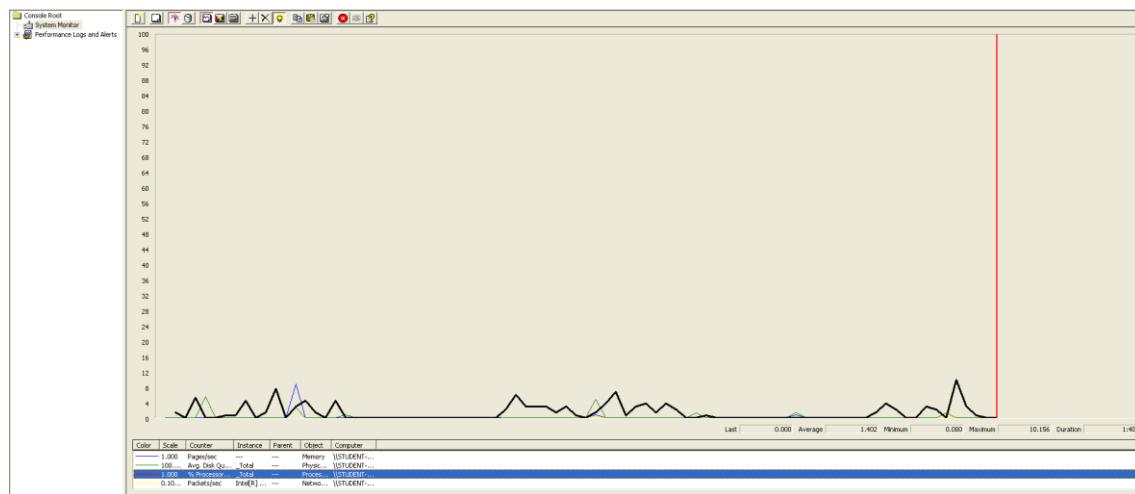
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	3415	139	47	370	3	3813	0.00%	38.2/sec	199.8
/scripts/cookieconsent.js	1228	143	56	382	3	1957	0.00%	13.7/sec	472.0
/styles/bootstrap.min.css	1225	184	91	446	7	2163	0.00%	13.7/sec	1627.1
/styles/style.css	1224	127	42	357	2	1843	0.00%	13.7/sec	21.6
/scripts/polyfills.js	1222	104	38	298	1	1416	0.00%	13.7/sec	103.2
/scripts/helpers.js	1219	92	35	265	2	1511	0.00%	13.7/sec	31.3
/scripts/jquery.min.js	1216	137	69	358	6	1405	0.00%	13.6/sec	1297.1
/styles/cookieconsent.mi...	1212	111	34	323	2	1353	0.00%	13.6/sec	56.3
/scripts/bootstrap.min.js	1208	142	58	404	3	1319	0.00%	13.5/sec	494.0
/wp-content/uploads/201...	1206	552	513	706	431	1568	0.00%	13.6/sec	2068.5
/favicon.ico	1202	137	34	405	2	1603	0.00%	13.6/sec	75.9
/security/login.do	1199	107	36	320	4	1364	0.00%	13.7/sec	75.0
/_spring_security_check	1189	1004	726	2262	12	7382	0.00%	13.8/sec	80.1
/newspaper/user/list.do	2226	1087	772	2519	31	10250	0.00%	26.2/sec	3764.1
/newspaper/user/getPub...	1110	954	602	2372	6	7030	0.00%	13.3/sec	180.5
/_spring_security_logout	1050	286	141	748	5	2104	0.00%	12.8/sec	68.4
TOTAL	22351	343	91	896	1	10250	0.00%	246.7/sec	10250.3

Como vemos no se producen errores con esta cantidad de usuarios, por lo que vamos a aumentar el número para ver si aparecen errores. Aumentamos el número a 230 usuarios simultáneos. Las gráficas que aparecen son los siguientes:



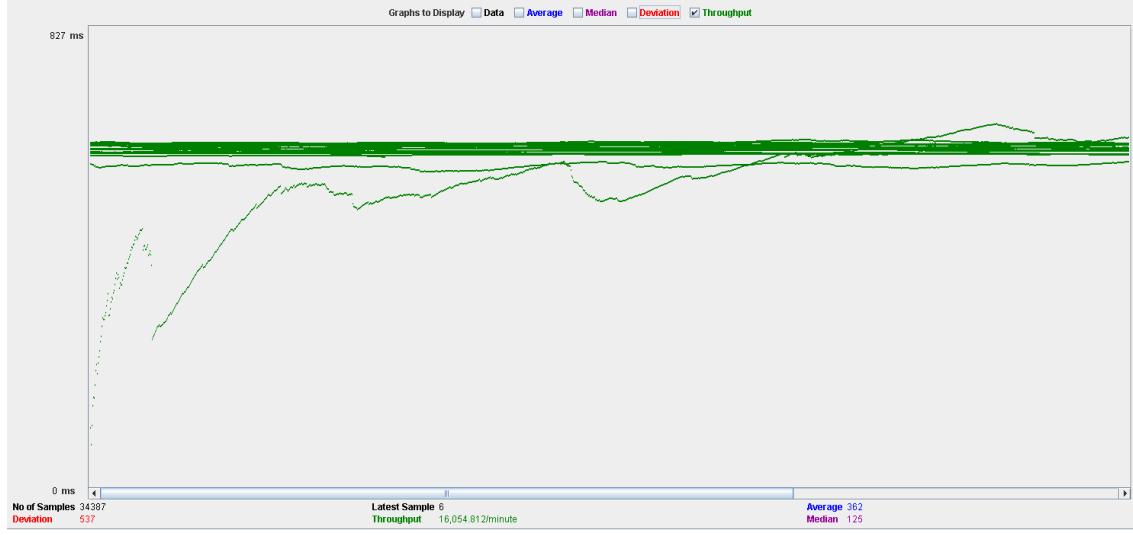
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	3703	278	56	795	0	3039	2.39%	37.0/sec	194.1
/scripts/cookieconsent.js	1337	242	54	752	0	2823	1.94%	13.7/sec	460.8
/styles/bootstrap.min.css	1336	273	89	815	0	3172	1.72%	13.7/sec	1593.5
/styles/style.css	1335	261	33	881	1	3051	1.35%	13.7/sec	21.4
/scripts/polyfills.js	1332	230	32	756	0	2831	1.28%	13.0/sec	101.6
/scripts/helpers.js	1332	199	29	678	0	2835	1.13%	13.0/sec	31.0
/scripts/jquery.min.js	1332	228	75	894	0	2849	0.83%	13.0/sec	1285.9
/styles/cookieconsent.mi...	1330	206	29	722	1	2832	0.60%	13.0/sec	56.1
/scripts/bootstrap.min.js	1330	250	50	799	1	2150	0.53%	13.0/sec	493.6
/wp-content/uploads/201...	1326	589	541	748	364	2525	0.00%	13.5/sec	2055.0
/favicon.ico	1311	241	30	817	1	3099	0.69%	13.5/sec	75.2
/security/login.do	1309	240	32	850	1	2747	1.22%	13.5/sec	73.1
/j_spring_security_check	1287	1112	853	2460	1	7035	2.64%	13.5/sec	78.4
/newspaper/user/list/do	2411	952	703	2164	0	6306	1.74%	25.0/sec	3560.6
/newspaper/user/publish...	1207	766	532	1857	1	5090	1.33%	13.0/sec	173.6
/j_spring_security_logout	1149	525	130	1411	1	3658	3.22%	12.7/sec	65.8
TOTAL	24367	419	104	1165	0	7035	1.51%	244.0/sec	10002.2

Con este valor empiezan a aparecer pequeños errores por lo que seguramente el valor máximo de usuarios concurrentes sea alrededor de 220. Viendo el performance podemos observar que lo que tiene una mayor carga es la memoria, por lo tanto, el cuello de botella se produciría ahí.



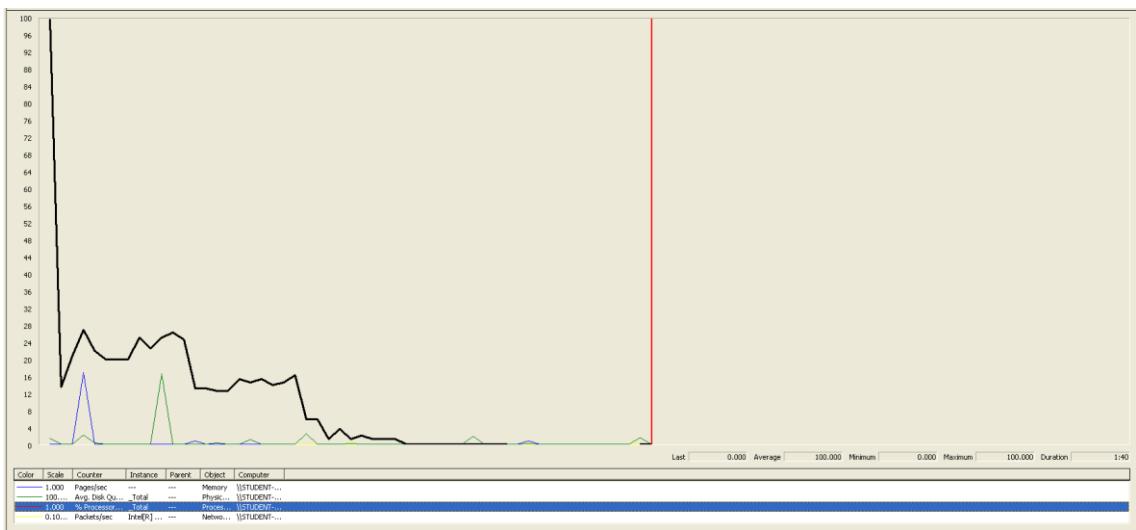
Caso de uso: publicar Newspaper.

Probamos inicialmente con 200 y 210 usuarios simultáneos, en ambos la probabilidad era cero, por lo tanto, aumentamos hasta 225 donde se empezó a producir errores, por lo tanto, este es aproximadamente el valor máximo de usuarios simultáneos. Las gráficas que se obtienen son:



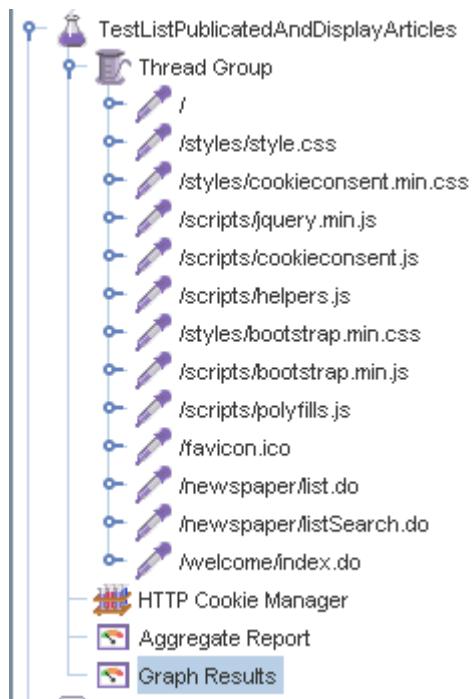
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	5305	229	58	658	0	6651	1.39%	41.8/sec	216.2
/styles/bootstrap.min.css	1875	245	94	685	1	1823	0.85%	14.8/sec	1745.8
/scripts/polyfills.js	1874	211	36	692	0	1936	0.69%	14.8/sec	111.1
/styles/chile.css	1870	203	38	685	1	1752	0.53%	14.8/sec	22.2
/scripts/jquery.min.js	1867	207	65	645	1	1915	0.48%	14.8/sec	1399.8
/styles/cookieconsent.mi...	1866	192	32	633	0	1716	0.49%	14.8/sec	61.0
/scripts/cookieconsent.js	1861	201	43	667	1	1944	0.43%	14.7/sec	503.5
/scripts/helpers.js	1651	163	28	593	1	1792	0.43%	14.6/sec	33.5
/scripts/bootstrap.min.js	1846	184	43	597	1	1809	0.43%	14.8/sec	530.8
/wp-content/uploads/201...	1843	434	397	589	295	1039	0.00%	14.5/sec	2215.9
/avocato_id	1834	178	25	630	2	1911	0.27%	14.6/sec	81.3
/secure/login.do	1832	186	28	651	2	1859	0.22%	14.5/sec	79.2
/j_spring_security_check	1786	977	783	2129	1	8906	2.13%	14.3/sec	82.4
/newspaper/user/list.do	3460	859	648	1836	1	7143	1.30%	28.1/sec	3951.1
/newspaper/user/publicis...	1724	714	488	1620	1	9038	0.52%	14.2/sec	191.3
/j_spring_security_logout	1693	485	361	1203	0	2451	1.30%	14.1/sec	74.5
TOTAL	34387	362	125	954	0	9038	0.81%	267.6/sec	11009.3

Se han producido los errores, mirando en el performance podemos observar que el cuello de botella se produce en el procesador debido a los altos valores que esta toma.

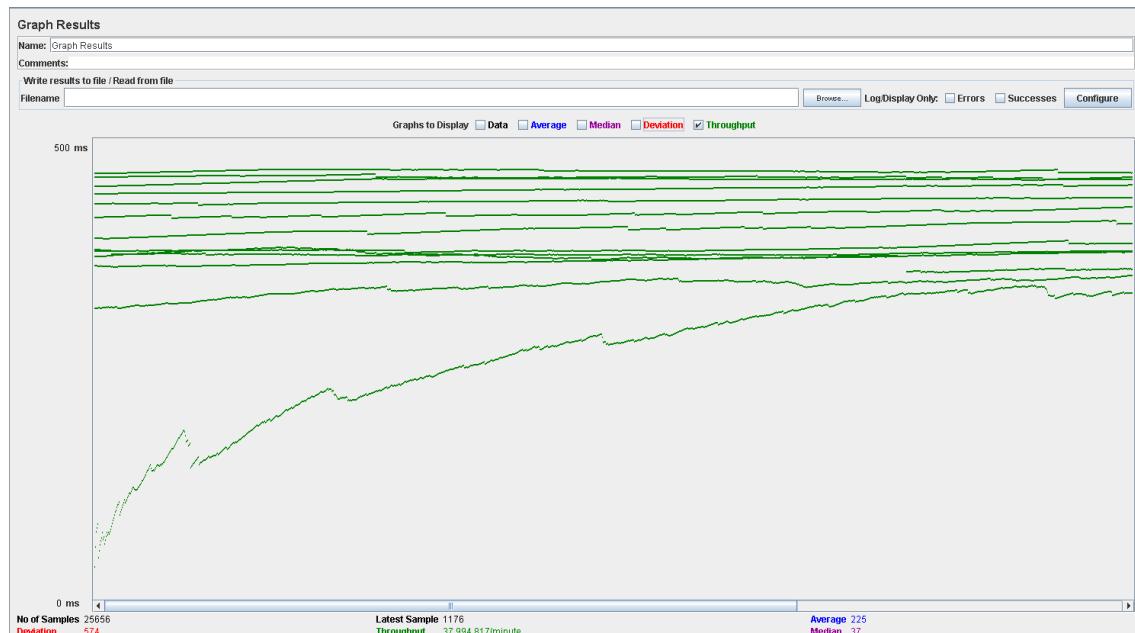


Caso de uso: buscador de newspaper.

Realizamos una prueba estableciendo como número de usuarios simultáneo 200 con un retraso de medio segundo entre persona y persona. Con esta configuración tras ejecutar el siguiente script:

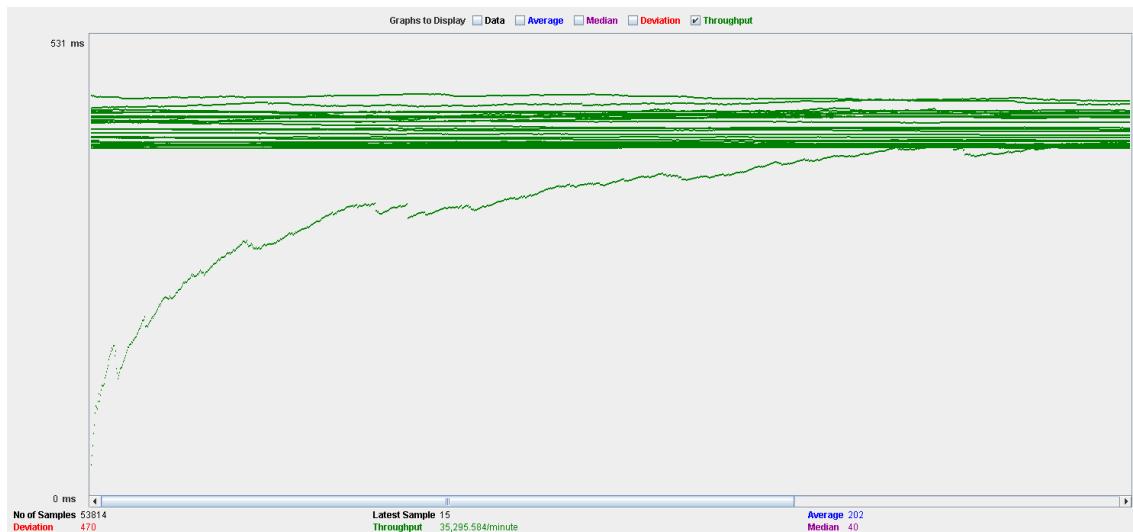


Al realizar el test nos salen las siguientes gráficas:



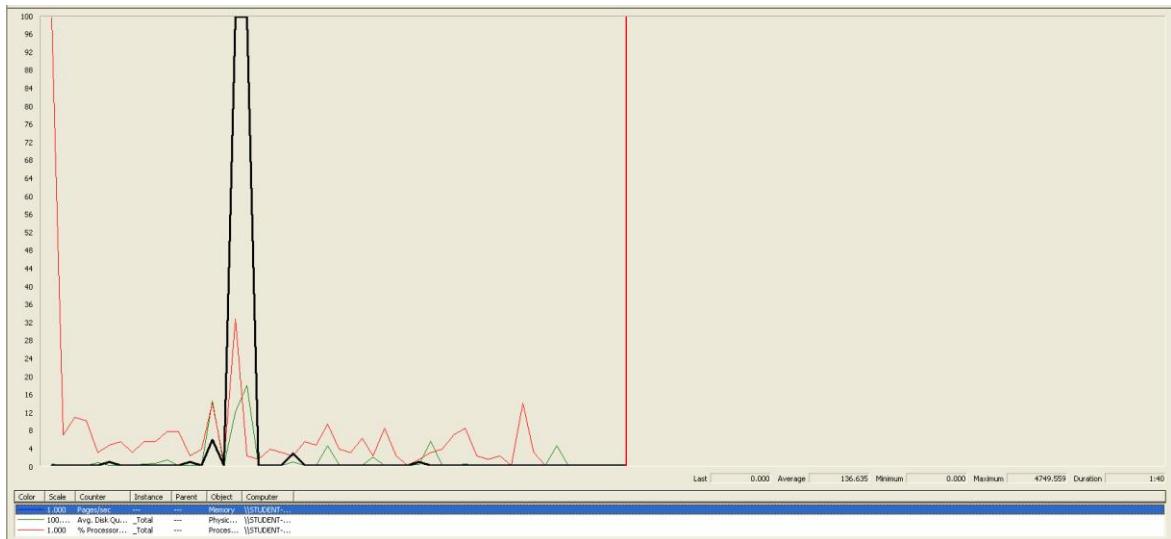
Aggregate Report											
Name: Aggregate Report Comments: Write results to file / Read from file Filename: <input type="text"/> Browse... Log Display Only Errors Successes Configure											
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec		
/	2031	200	31	461	0	5012	12.46%	51.5/sec	236.4		
/styles/style.css	2021	128	25	250	0	4984	11.97%	51.2/sec	79.4		
/styles/cookieconsent.mi...	2012	113	24	215	0	4986	11.93%	51.0/sec	194.6		
/scripts/jquery.min.js	2003	160	52	333	0	5036	11.58%	50.8/sec	4280.7		
/scripts/cookieconsent.js	1996	137	33	264	0	4994	11.62%	50.7/sec	1547.9		
/scripts/helpers.js	1989	128	26	260	0	5008	10.96%	50.5/sec	110.8		
/styles/bootstrap.min.css	1983	167	55	303	0	5047	10.39%	50.4/sec	5365.5		
/scripts/bootstrap.min.js	1978	140	33	273	0	5000	10.36%	50.4/sec	1657.1		
/scripts/polyfills.js	1974	127	24	233	0	4984	9.83%	50.3/sec	348.7		
/favicon.ico	1965	116	24	232	0	4978	9.41%	50.3/sec	261.6		
/newspaper/list/do	1956	655	309	1604	0	11054	9.25%	49.7/sec	291.4		
/newspaper/listSearch.do	1907	687	365	1688	0	12190	10.59%	47.5/sec	282.3		
/welcome/index.do	1841	196	27	475	0	4992	13.20%	47.2/sec	214.6		
TOTAL	25656	225	37	598	0	12190	11.04%	633.2/sec	14458.4		

Como vemos los valores del error son bastante elevados por lo que probamos a buscar el máximo de usuarios concurrentes disminuyendo el número de usuarios. Probamos con 160 y en ese caso las gráficas resultantes son:



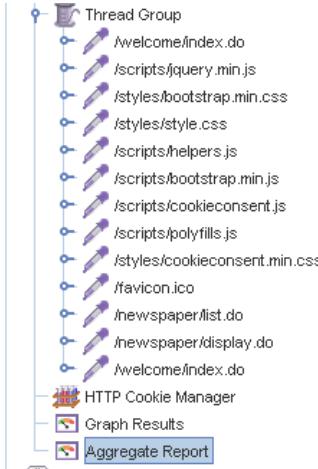
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	4189	88	27	246	1	3139	0.24%	45.8/sec	229.9
/styles/style.css	4184	61	25	121	1	2712	0.19%	45.8/sec	71.6
/styles/cookieconsent.mi...	4178	68	26	147	1	2473	0.10%	45.7/sec	189.0
/scripts/jquery.min.js	4172	104	58	215	6	2129	0.05%	45.7/sec	4344.0
/scripts/cookieconsent.js	4159	100	36	270	1	2761	0.07%	45.5/sec	1561.3
/scripts/helpers.js	4157	92	28	258	1	2708	0.07%	45.5/sec	103.9
/styles/bootstrap.min.css	4145	127	65	271	6	2743	0.07%	45.4/sec	5379.1
/scripts/bootstrap.min.js	4139	98	37	241	3	2723	0.07%	45.3/sec	1651.6
/scripts/polyfills.js	4131	88	27	254	1	2739	0.05%	45.2/sec	340.5
/favicon.ico	4127	73	27	177	1	2472	0.02%	45.2/sec	252.7
/newspaper/list/do	4120	761	437	1922	10	7163	0.02%	45.1/sec	291.0
/newspaper/listSearch.do	4076	857	587	2034	10	7495	0.27%	44.6/sec	288.5
/welcome/index.do	4037	121	27	380	1	2674	0.37%	44.3/sec	221.7
TOTAL	53814	202	40	538	1	7495	0.12%	588.3/sec	14907.4

Con este valor los errores son muy cercanos a cero por lo que podemos ver que 160 es un valor muy aproximado al número de usuarios máximo simultaneo que puede tener la aplicación para este caso de uso. Analizando el performance podemos ver que es en la memoria donde ocurre un cuello de botella.



Caso de uso: listar todos los newspapers y desplegar sus artículos.

Realizamos una prueba estableciendo como número de usuarios simultáneo con un retraso de medio segundo entre usuario y usuario. Con esta configuración tras ejecutar el siguiente script:

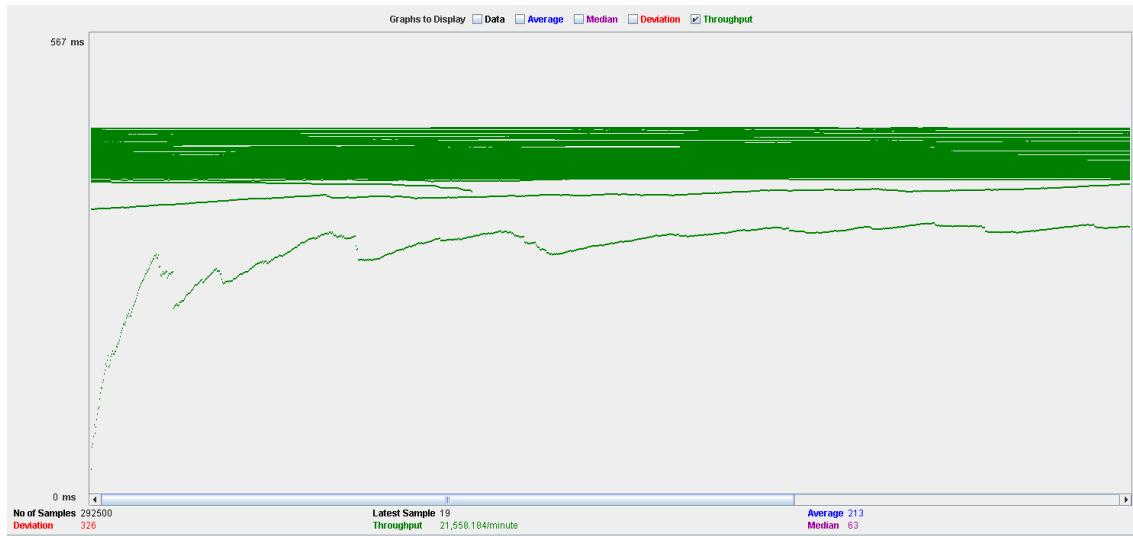


Al realizar el test aparecen las siguientes gráficas:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/welcome/index.do	40000	50	19	130	2	1453	0.00%	64.8/sec	325.6
/scripts/jquery.min.js	20000	74	44	174	3	1500	0.00%	32.7/sec	3108.6
/styles/bootstrap.min.css	20000	80	48	187	4	1553	0.00%	32.7/sec	3874.9
/styles/style.css	20000	51	20	143	1	1504	0.00%	32.7/sec	51.3
/scripts/helpers.js	20000	45	19	120	1	1437	0.00%	32.7/sec	74.8
/scripts/bootstrap.min.js	20000	50	26	121	2	1538	0.00%	32.7/sec	1191.7
/scripts/cookieconsent.js	20000	51	24	126	2	1442	0.00%	32.7/sec	1121.4
/scripts/polyfills.js	20000	44	18	120	1	1474	0.00%	32.7/sec	246.2
/styles/cookieconsent.mi...	20000	41	17	107	1	1443	0.00%	32.7/sec	135.3
/favicon.ico	20000	39	17	98	1	1412	0.00%	32.7/sec	182.9
/newspaper/list.do	20000	277	129	758	5	3909	0.00%	32.6/sec	210.7
/newspaper/display.do	20000	313	149	855	5	4992	0.00%	32.6/sec	214.7
TOTAL	260000	90	28	212	1	4992	0.00%	421.0/sec	10651.3

Como vemos el error es cero, así que vamos a aumentar el número de personas conectadas simultáneamente conectadas. Probamos con 225 y en ese caso las gráficas resultantes son:



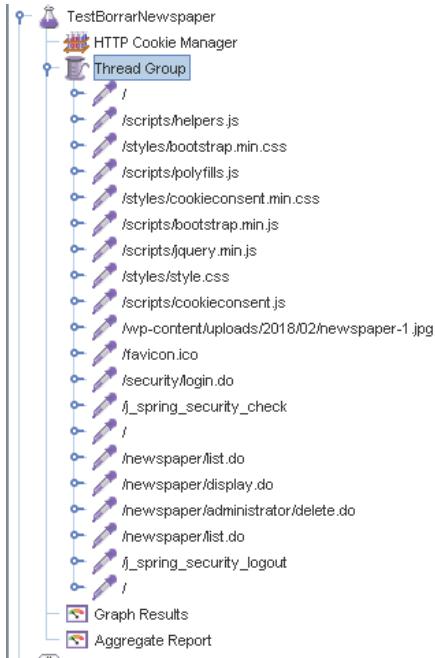
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/welcome/index.do	45000	177	37	565	0	2866	0.62%	55.3/sec	276.7
/scripts/jquery.min.js	22500	235	101	644	0	2963	0.69%	27.8/sec	2629.3
/scripts/bootstrap.min.css	22500	215	84	598	0	3054	0.69%	27.8/sec	3277.0
/scripts/style.css	22500	185	39	581	0	2918	0.78%	27.8/sec	43.6
/scripts/helpers.js	22500	162	38	506	0	2980	0.75%	27.8/sec	63.4
/scripts/bootstrap.min.js	22500	158	48	479	0	2945	0.69%	27.8/sec	1007.9
/scripts/cookieconsent.js	22500	162	45	492	0	3148	0.68%	27.8/sec	948.5
/scripts/polyfills.js	22500	159	36	498	0	2934	0.68%	27.8/sec	208.4
/scripts/cookieconsent.mi...	22500	152	35	470	0	2969	0.64%	27.8/sec	114.7
/favicon.ico	22500	145	35	460	0	2932	0.63%	27.8/sec	154.9
/newspaper/list/do	22500	404	221	994	0	5443	0.31%	27.8/sec	179.1
/newspaper/display/do	22500	443	256	1073	0	5156	0.41%	27.8/sec	182.4
TOTAL	292500	213	63	618	0	5443	0.63%	359.3/sec	9032.7

Vemos que los errores son muy parecidos a cero por lo tanto el número máximo de usuarios simultáneamente haciendo lo mismo es de 225. Analizando el performance podemos ver que el procesador es el que tiene mayor peso.



Caso de uso: borrar newspaper.

Realizamos una prueba estableciendo como número de usuarios simultáneo 200 con un retraso de medio segundo entre usuario y usuario. Con esta configuración tras ejecutar el siguiente script:

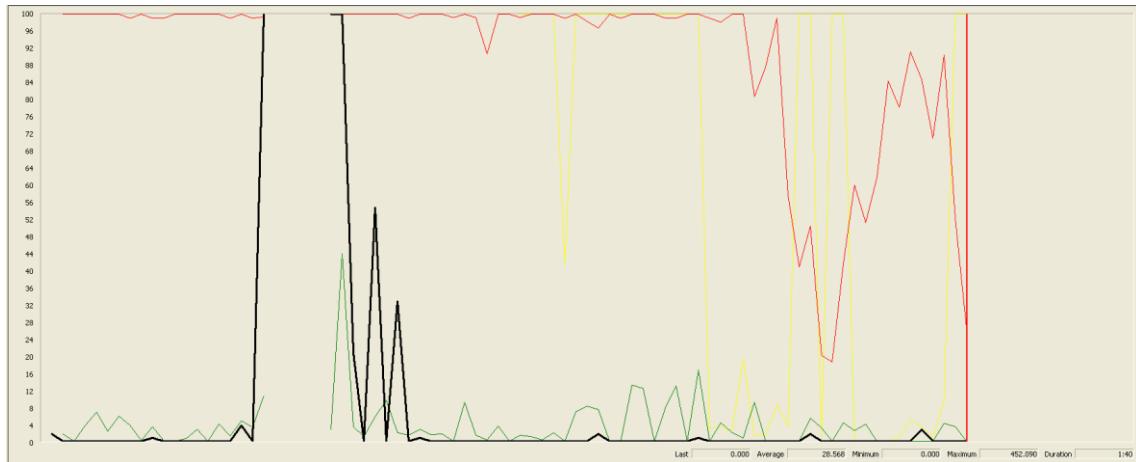


Al realizar el test aparecen las siguientes gráficas:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	60000	427	61	1119	0	44778	1.97%	23.6/sec	123.3
/scripts/helpers.js	20000	364	56	846	1	44923	1.68%	7.9/sec	17.9
/styles/bootstrap.min.css	20000	447	114	1002	1	44783	1.60%	7.9/sec	920.6
/scripts/polyfills.js	20000	377	52	836	1	46706	1.52%	7.9/sec	59.7
/scripts/cookieconsent.mi...	20000	421	53	1010	1	44766	1.44%	7.9/sec	32.3
/scripts/bootstrap.min.js	20000	372	68	795	1	44783	1.32%	7.9/sec	284.0
/scripts/jquery.min.js	20000	404	95	909	1	44772	1.29%	7.9/sec	74.5
/scripts/style.css	20000	373	49	870	0	44722	1.26%	7.9/sec	12.3
/scripts/cookieconsent.js	20000	410	64	936	1	44770	1.23%	7.9/sec	267.5
/wp-content/uploads/201...	20000	1366	519	2256	197	27094	0.00%	7.9/sec	1201.3
/favicon.ico	20000	382	37	898	1	46701	0.96%	7.9/sec	43.8
/securitylogin.do	20000	408	38	844	1	45464	1.16%	7.9/sec	42.8
/j_spring_security_check	20000	1684	675	4306	1	48809	2.23%	7.9/sec	47.3
/newspaper/list.do	40000	1183	592	2920	1	460369	1.74%	15.8/sec	2134.2
/newspaper/display.do	20000	936	325	2466	1	45232	1.26%	7.9/sec	103.5
/newspaper/administrat...	20000	1020	301	2961	0	46379	1.60%	7.9/sec	99.2
/j_spring_security_logout	20000	1010	142	3068	1	45947	2.20%	7.9/sec	41.3
TOTAL	400000	686	118	1829	0	48809	1.51%	157.1/sec	6155.9

Vemos que los errores son muy parecidos a cero por lo tanto el número máximo de usuarios simultáneamente haciendo lo mismo es de 200. Analizando el performance podemos ver que el procesador, tarjeta de red y memoria son los principales causantes de que haya un cuello de botella.



Caso de uso: listar los newspapers con palabras taboo.

Realizamos una prueba estableciendo como número de usuarios simultáneo 200 con un retraso de medio segundo entre usuario y usuario. Con esta configuración tras ejecutar el siguiente script:



Al realizar el test aparecen las siguientes gráficas:



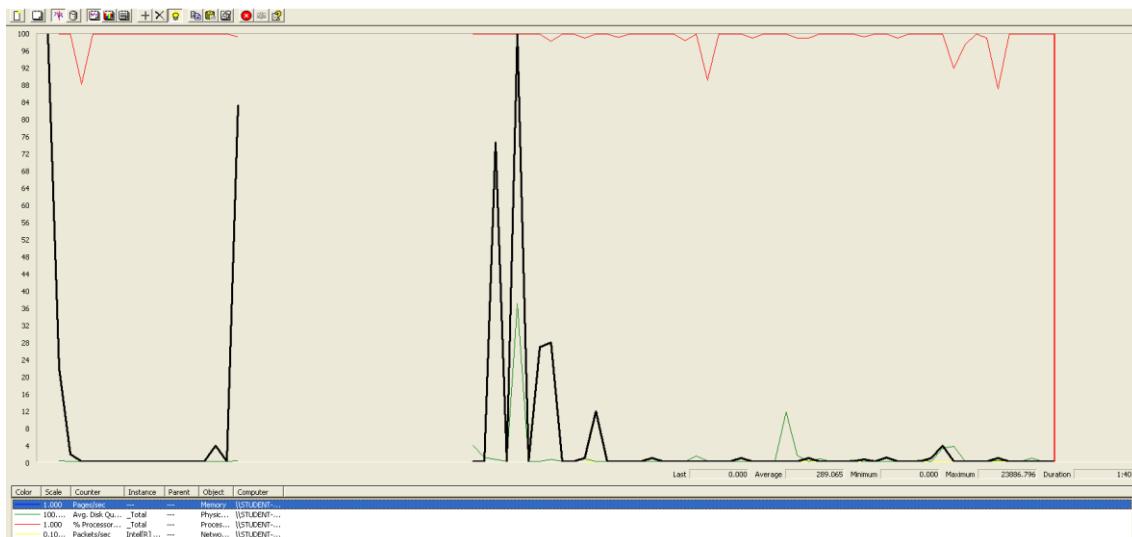
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	60000	36	18	78	1	1415	0.00%	72.1/sec	382.5
/styles/bootstrap.min.css	20000	66	42	142	3	1578	0.00%	24.2/sec	2872.9
/scripts/jquery.min.js	20000	64	37	144	3	1052	0.00%	24.2/sec	2304.8
/styles/style.css	20000	42	17	106	1	931	0.00%	24.2/sec	38.1
/scripts/bootstrap.min.js	20000	47	24	103	2	1447	0.00%	24.2/sec	883.5
/styles/cookieconsent.mi...	20000	37	16	86	1	931	0.00%	24.2/sec	100.3
/scripts/polyfills.js	20000	39	17	90	1	876	0.00%	24.2/sec	182.5
/scripts/helpers.js	20000	37	16	81	1	959	0.00%	24.2/sec	55.5
/scripts/cookieconsent.js	20000	43	22	95	2	1554	0.00%	24.2/sec	831.4
/favicon.ico	20000	37	16	85	1	922	0.00%	24.2/sec	135.6
/security/login.do	20000	49	20	121	2	1451	0.00%	24.2/sec	132.2
/_spring_security_check	20000	373	205	887	5	8741	0.00%	24.2/sec	147.6
/newspaper/administrat...	40000	344	168	830	8	8067	0.00%	48.3/sec	321.4
/_spring_security_logout	20000	59	43	229	2	1502	0.00%	24.2/sec	128.6
TOTAL	340000	101	28	240	1	8741	0.00%	408.3/sec	8453.2

Como vemos el error es cero, así que vamos a aumentar el número de personas conectadas simultáneamente. Probamos con 225 y en ese caso las gráficas resultantes son:



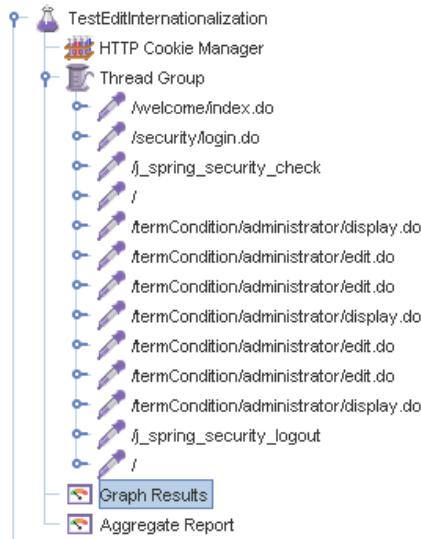
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	67500	117	27	367	0	3894	0.37%	69.0/sec	365.0
/styles/bootstrap.min.css	22500	142	55	395	0	3573	0.29%	23.1/sec	273.7
/scripts/quen.min.js	22600	147	49	429	0	3867	0.20%	23.1/sec	2196.0
/styles/style.css	22500	125	24	403	0	2173	0.29%	23.1/sec	36.3
/scripts/bootstrap.min.js	22500	128	36	385	0	3702	0.28%	23.1/sec	842.0
/styles/cookieconsent.mi...	22500	113	24	363	0	3942	0.28%	23.1/sec	95.7
/scripts/polyfills.js	22500	117	26	364	0	3511	0.27%	23.1/sec	174.0
/scripts/helpers.js	22500	113	25	357	0	3622	0.27%	23.1/sec	52.9
/scripts/cookieconsent.js	22500	121	34	367	0	3700	0.28%	23.1/sec	792.3
/favicon.ico	22500	116	25	363	0	3524	0.27%	23.1/sec	129.3
/security/login.do	22500	137	26	436	0	2700	0.20%	23.1/sec	126.3
/j_spring_security_check	22500	455	291	1057	0	7031	0.52%	23.1/sec	140.6
/newspaper/administrato...	45000	303	154	717	0	6716	0.23%	46.2/sec	306.7
/j_spring_security_logout	22500	284	122	769	0	4707	0.40%	23.1/sec	122.6
TOTAL	382500	174	48	497	0	8716	0.31%	390.8/sec	8067.7

Vemos que los errores son muy parecidos a cero por lo tanto el número máximo de usuarios simultáneamente haciendo lo mismo es de 225. Analizando el performance podemos ver que la memoria es el principal causante de que haya un cuello de botella.

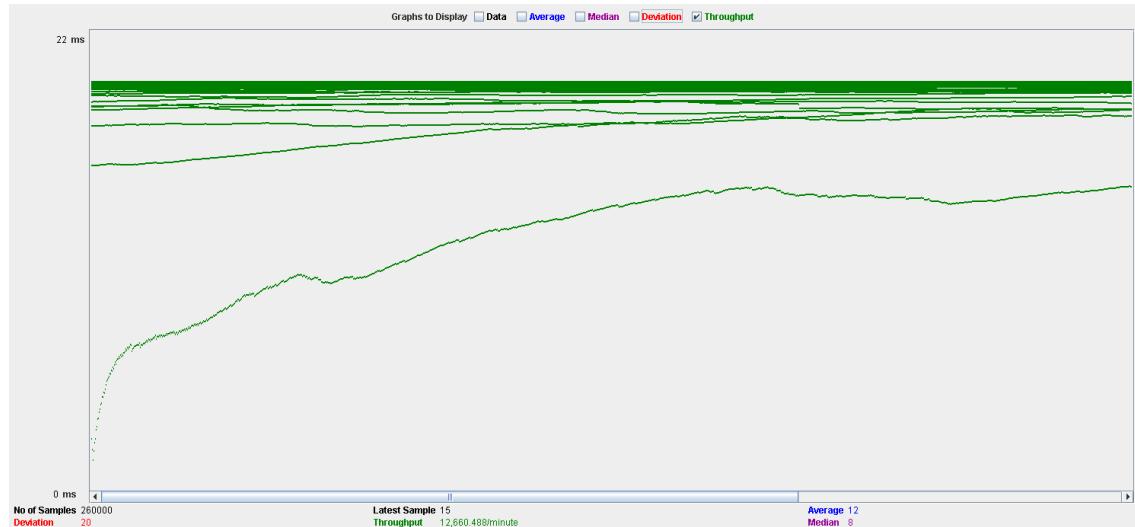


Caso de uso: editar los términos y condiciones.

Realizamos una prueba estableciendo como número de usuarios simultáneo 200 con un retraso de medio segundo entre usuario y usuario. Con esta configuración tras ejecutar el siguiente script:



Al realizar el test aparecen las siguientes gráficas:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/welcome/index.do	20000	5	4	8	1	320	0.00%	16.4/sec	82.5
/security/login.do	20000	5	5	8	2	323	0.00%	16.4/sec	89.6
/j_spring_security_check	20000	12	10	20	3	668	0.00%	16.4/sec	100.1
/	40000	4	4	7	1	394	0.00%	32.5/sec	177.3
/termCondition/administr...	60000	13	9	23	3	913	0.00%	48.9/sec	469.0
/termCondition/administr...	80000	18	12	33	3	1038	0.00%	65.3/sec	592.2
/j_spring_security_logout	20000	9	7	14	3	386	0.00%	16.4/sec	87.2
TOTAL	260000	12	8	22	1	1038	0.00%	211.0/sec	1587.9

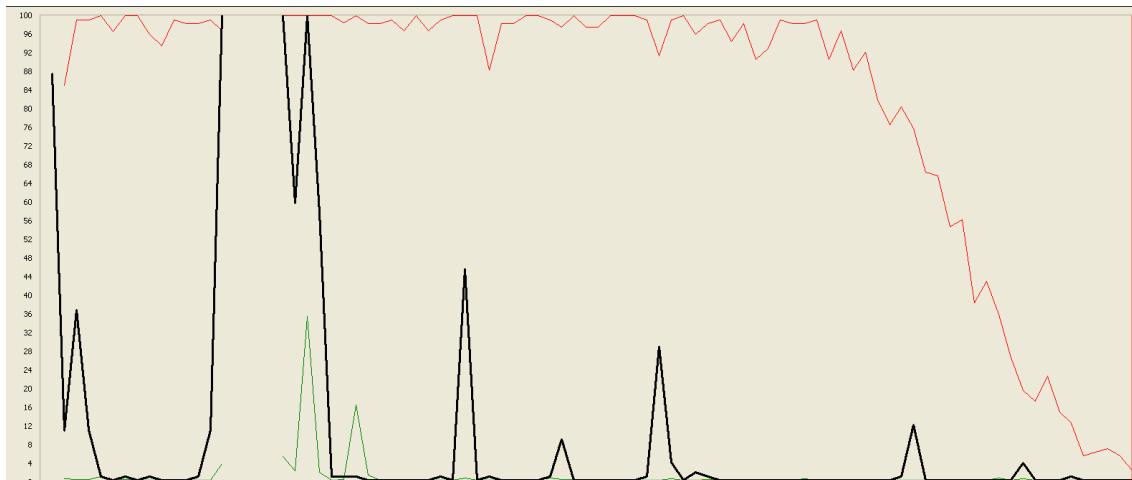
Como vemos el error es cero, así que vamos a aumentar el número de personas conectadas simultáneamente conectadas. Probamos con 350 y en ese caso las gráficas resultantes son:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
Welcome/index.do	36000	63	19	194	0	1117	0.34%	24.1/sec	121.2
/security/login.do	35000	67	19	206	0	1490	0.26%	24.1/sec	131.8
jl_Spring_security_check	35000	209	139	470	0	3415	0.44%	24.1/sec	147.0
/	70000	89	21	205	0	1413	0.41%	48.0/sec	260.8
TermCondition/administr...	105000	168	114	370	0	3125	0.69%	72.1/sec	688.9
TermCondition/administr...	140000	250	172	567	0	4058	0.48%	96.2/sec	868.7
jl_Spring_security_logout	35000	141	57	386	0	2176	0.43%	24.1/sec	128.2
TOTAL	455000	163	91	404	0	4058	0.48%	311.0/sec	2332.7

Vemos que los errores son muy parecidos a cero por lo tanto el número máximo de usuarios simultáneamente haciendo lo mismo es de aproximadamente 350 para este caso de uso.

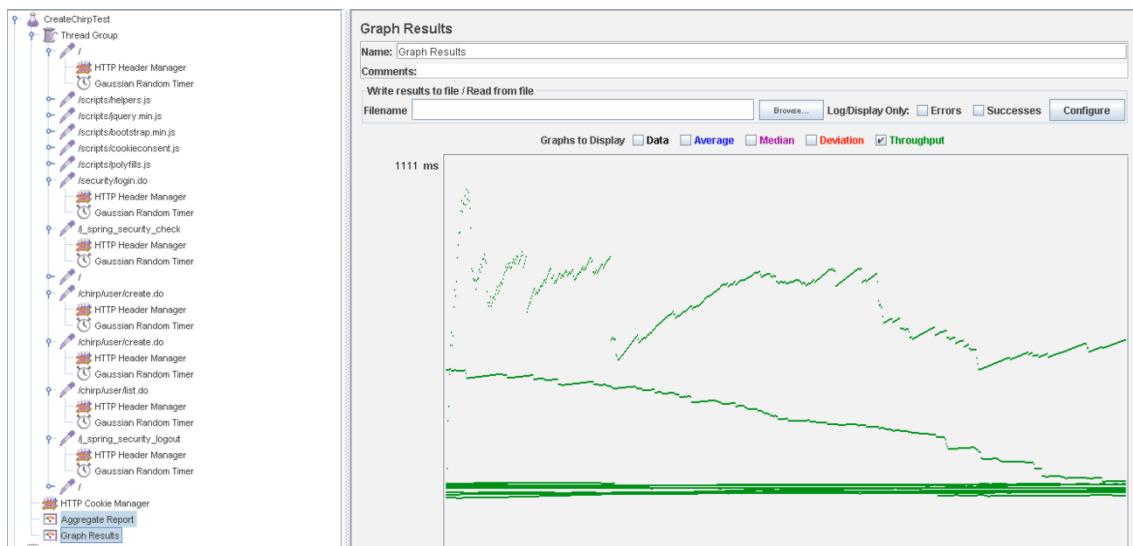
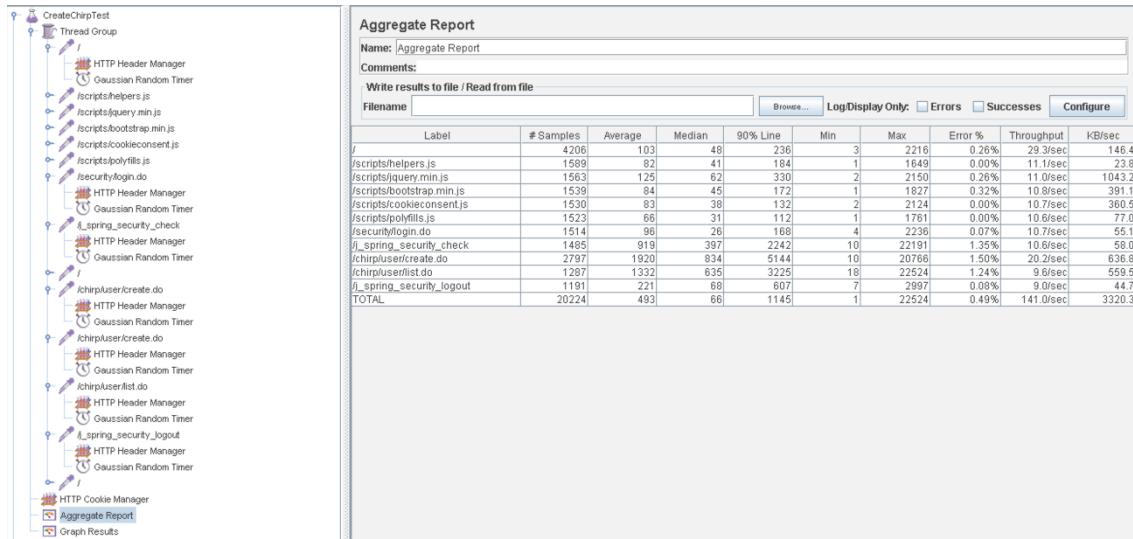
Analizando el performance podemos ver que la memoria es el principal causante de que haya un cuello de botella.



Caso de uso: crear Chirp.

Realizamos un test en el que iniciamos sesión como usuario, creamos un chirp y nos des autenticamos.

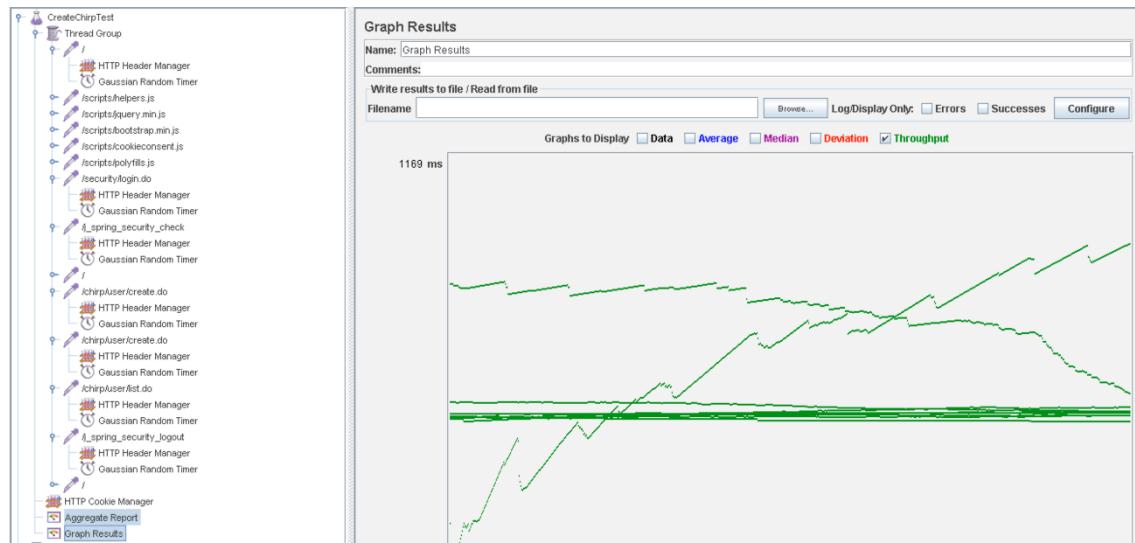
En las pruebas, se producen errores cuando hay 285 usuarios realizando 100 veces el caso de uso cada uno, como podemos ver en las gráficas:



Cuando son 270 los usuarios que realizan 100 veces el caso de uso cada uno, deja de haber errores, lo comprobamos:

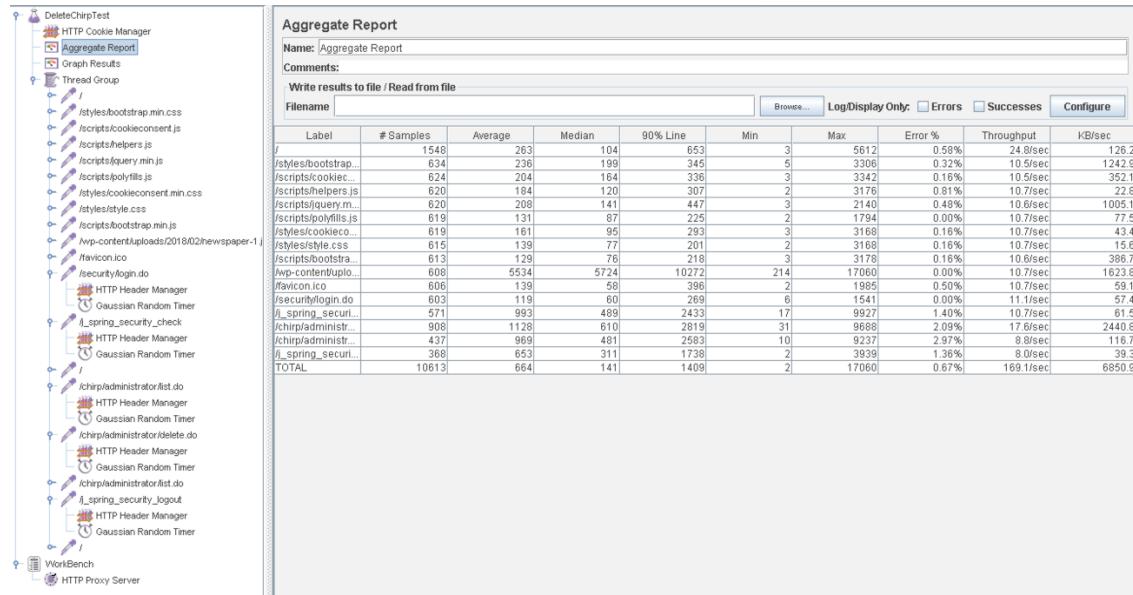
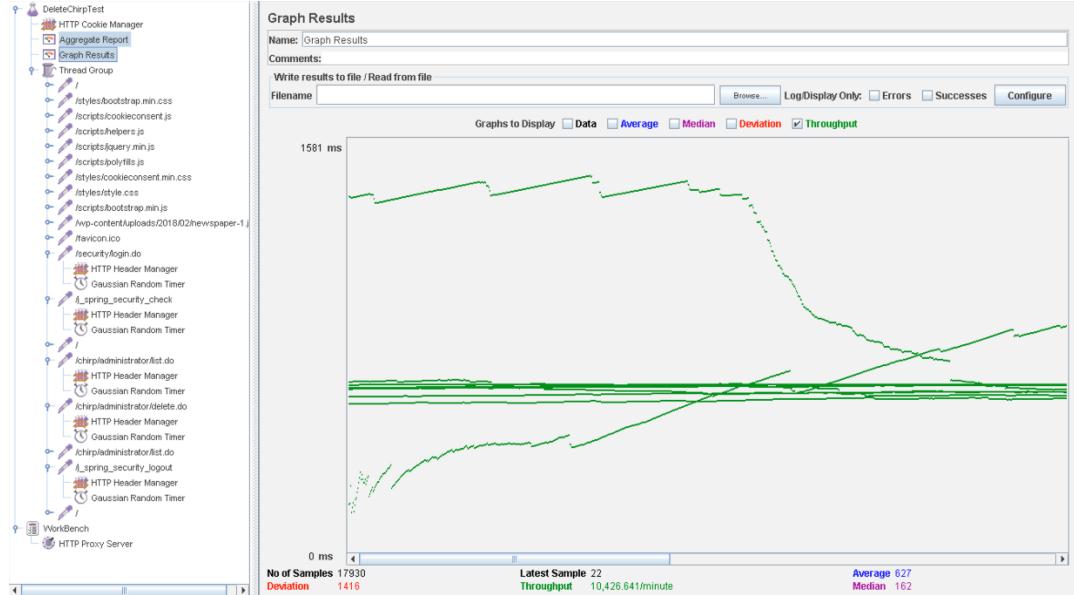
Aggregate Report

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/scripts/helpers.js	3571	63	43	142	2	611	0.00%	30.3/sec	191.1
/scripts/jquery.min.js	1301	59	39	123	2	564	0.00%	14.0/sec	30.0
/scripts/bootstrap.min.js	1297	82	61	156	2	677	0.00%	14.0/sec	1336.5
/scripts/cookieconsent.js	1289	62	48	122	2	438	0.00%	14.0/sec	509.2
/scripts/polyfills.js	1287	65	40	137	3	591	0.00%	14.0/sec	472.8
/scripts/login.do	1287	56	38	119	2	653	0.00%	14.1/sec	101.6
/j_spring_security_check	1274	55	25	148	3	559	0.00%	14.2/sec	73.2
/chirpuser/create.do	1229	732	293	1935	11	9396	0.00%	13.9/sec	76.9
/chirpuser/list.do	2274	1801	928	4808	12	13954	0.00%	26.3/sec	1512.3
/chirpuser/logout	1066	1332	830	3108	26	15596	0.00%	12.9/sec	1445.3
/j_spring_security_logout	1049	107	63	264	7	824	0.00%	13.0/sec	64.8
TOTAL	16923	428	62	1106	2	15596	0.00%	181.2/sec	5491.4

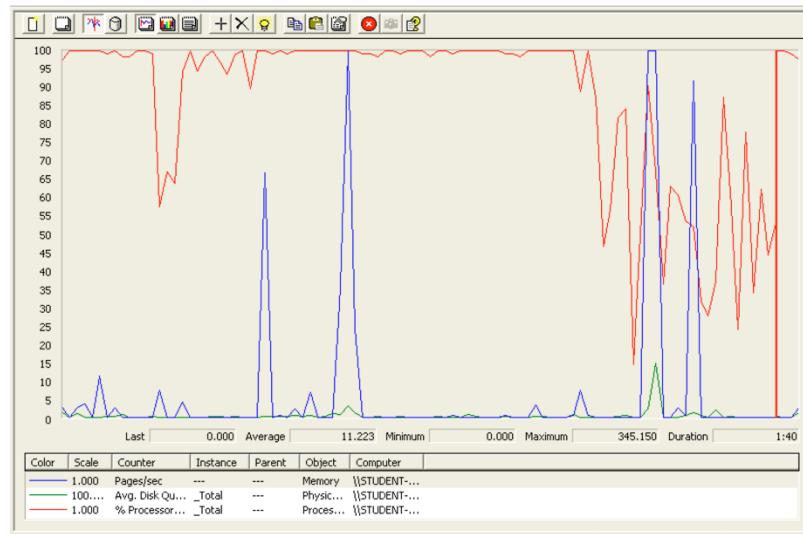


Caso de uso: borrar Chirp.

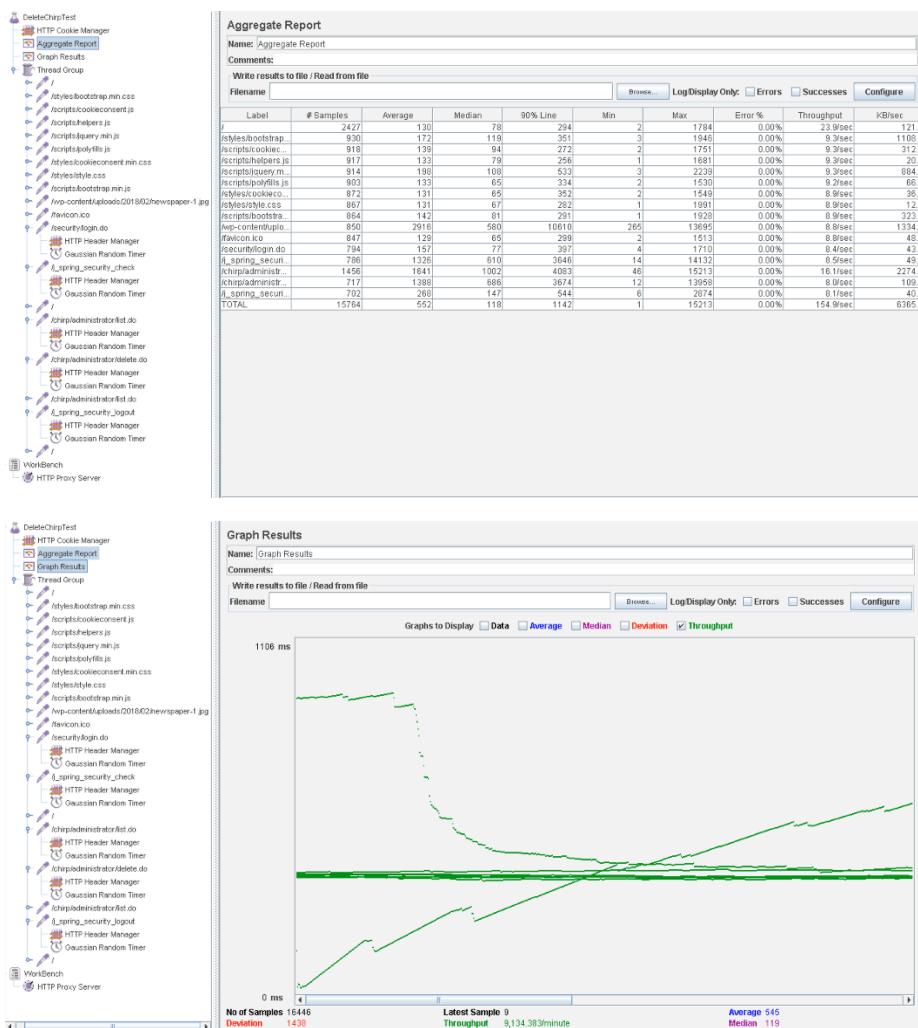
En este test nos identificaremos como administrador y navegaremos hacia la vista en la que se listan los chirps y procederemos a borrarlo, tras eso cerramos sesión. Realizando la primera prueba con 270 usuarios realizando 100 veces el caso de uso cada uno a la vez, esto nos dará errores.



Como se puede apreciar en la siguiente figura, el uso del CPU llega a estar cerca del 100% durante buena parte del proceso:

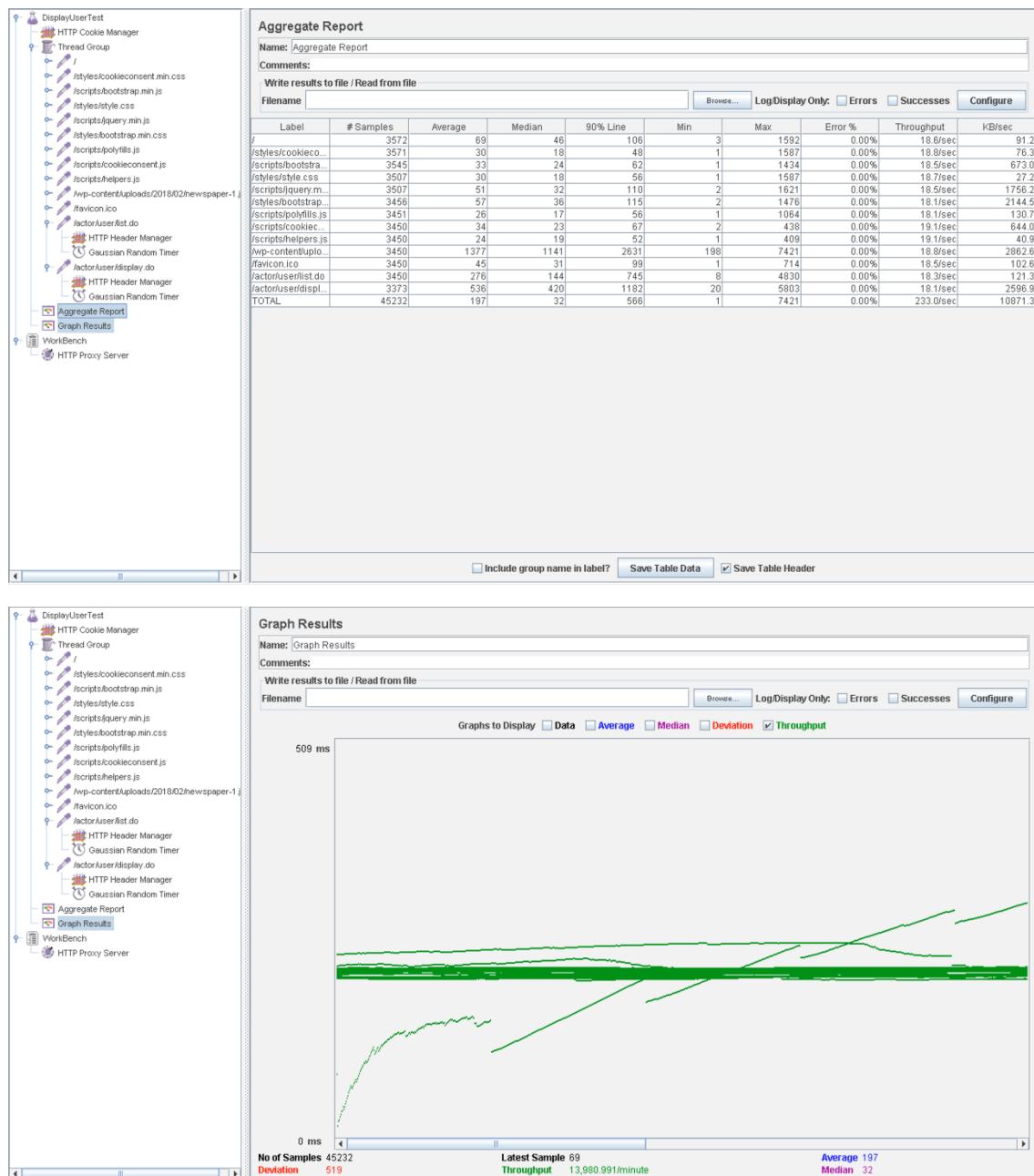


Disminuyendo el número de usuarios hasta los 240, no se producen errores:



Caso de uso: desplegar User.

En este caso, un usuario no autenticado navega a la lista de usuarios y de ahí muestra el perfil de un usuario donde se muestran, además de sus datos, sus artículos publicados y chirps. En una primera prueba de rendimiento se realiza con 200 usuarios realizando 200 veces las acciones y en principio no se producen errores:



Pero en cuanto se aumenta un poco más, el número de usuarios empiezan a verse fallos y al subirlo un número elevado (300) los fallos son muy comunes llegado cierto punto del test:

DisplayUserTest

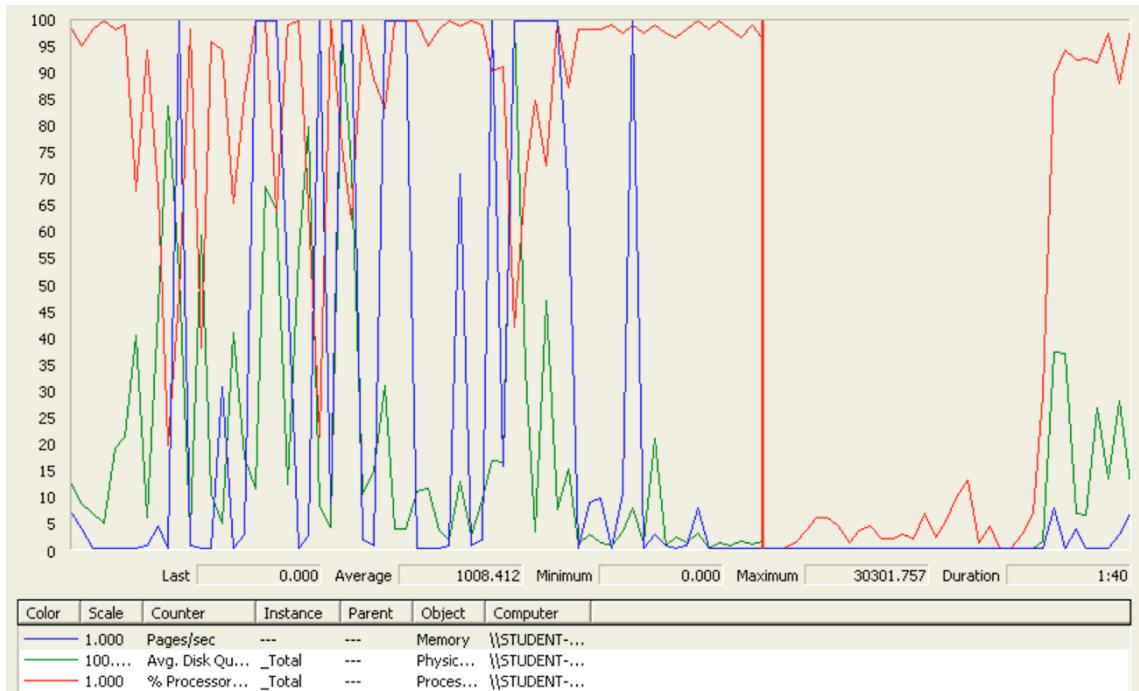
- HTTP Cookie Manager
- Thread Group
 - /
 - /styles/cookieconsent.min.css
 - /scripts/bootstrap.min.js
 - /styles/style.css
 - /scripts/jquery.min.js
 - /scripts/bootstrap.min.css
 - /scripts/polyfills.js
 - /scripts/cookieconsent.js
 - /scripts/helpers.js
 - /wp-content/uploads/2018/02/newspaper-1.jpg
 - /favicon.ico
 - /actorUserList.do
 - HTTP Header Manager
 - Gaussian Random Timer
 - /actorUserDisplay.do
 - HTTP Header Manager
 - Gaussian Random Timer
- Aggregate Report
- Graph Results
- WorkBench
- HTTP Proxy Server

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file
Filename: Browse... Log Display Only Errors Successes Configure

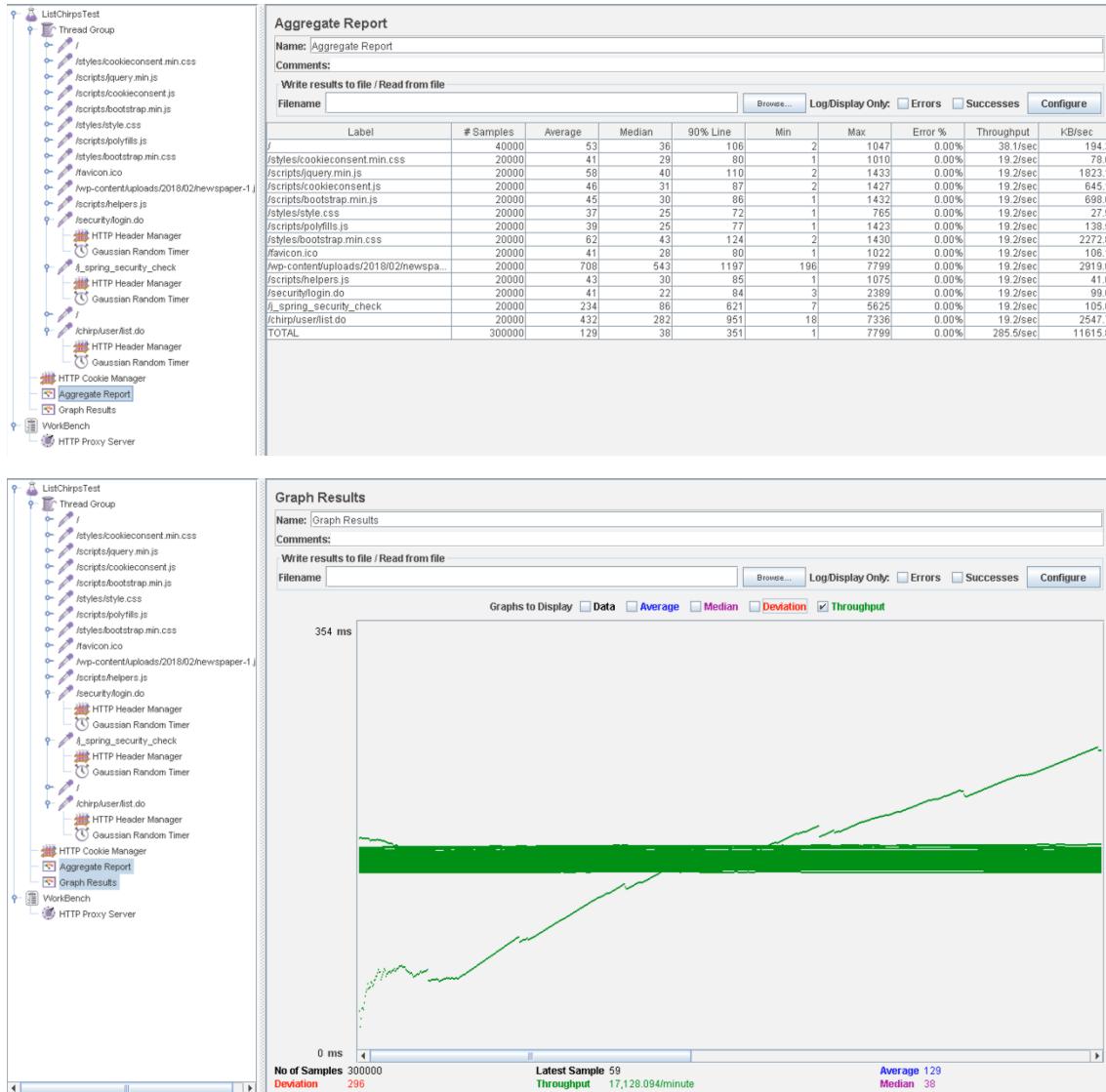
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/styles/style.css	911	236	66	793	3	2711	3.29%	24.8/sec	36.4
/favicon.ico	929	189	65	560	2	2104	5.81%	23.3/sec	124.3
/styles/bootstrap...	918	296	131	730	4	2319	4.79%	25.2/sec	2846.9
/scripts/cookiec...	973	237	85	684	1	2328	2.67%	24.4/sec	801.8
/scripts/polyfills.js	957	233	82	633	1	2320	4.81%	24.0/sec	167.7
/scripts/helpers.j...	901	251	107	763	3	2306	4.11%	24.6/sec	2241.8
/wp-content/uplo...	931	1697	664	5085	240	7094	0.00%	23.5/sec	3573.7
/scripts/helpers.j...	963	212	65	629	2	2308	2.49%	24.1/sec	51.6
/actorUserList.do	854	1001	538	2532	4	8546	1.87%	21.6/sec	141.5
/actorUser/displ...	812	1373	888	3320	2	8804	4.31%	21.7/sec	2986.5
/	805	346	131	943	6	2290	8.45%	21.6/sec	100.4
/styles/cookieco...	786	263	75	808	2	2328	3.56%	21.1/sec	84.5
/scripts/bootstrap...	779	230	68	751	3	2149	2.57%	21.0/sec	747.2
TOTAL	11519	500	125	1311	1	8604	3.72%	288.8/sec	13205.8

En este caso, la vista de display muestra distintos datos (artículos, chirps y datos del usuario) por lo que el uso del disco es mucho mayor, así se refleja en la gráfica del Performance.



Caso de uso: listar Chirps.

Para este caso de uso, un usuario inicia sesión y entra a la lista de chirps donde verá los chirps creados por la gente a la que sigue y por él mismo. Comenzamos este test de rendimiento con 200 usuarios realizando 100 acciones a la vez, sin que se muestre ningún error:



Cuando aumentamos los usuarios hasta 275 y disminuimos el número de acciones simultaneas a 75, empezamos a ver pequeñas cantidades de fallos:

Aggregate Report

Name: Aggregate Report
Comments:
Write results to file / Read from file
Filename: LogDisplay Only, Errors, Successes,

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	27500	165	59	362	2	9159	1.23%	33.5/sec	169.6
/styles/cookieconsent.min.css	13750	147	48	309	1	9131	1.59%	16.9/sec	68.3
/scripts/jquery.min.js	13750	147	68	327	2	9133	0.97%	16.9/sec	1592.5
/scripts/cookieconsent.js	13750	122	52	265	1	9173	0.96%	16.9/sec	503.9
/scripts/bootstrap.min.js	13750	124	52	265	1	9116	0.81%	16.9/sec	611.1
/styles/style.css	13750	113	46	233	1	9124	0.83%	16.9/sec	24.7
/scripts/polyfills.js	13750	116	44	238	1	9106	0.76%	16.9/sec	121.7
/styles/bootstrap.min.css	13750	146	71	314	2	9107	0.93%	16.9/sec	1988.7
/favicon.ico	13750	128	47	271	1	9131	0.95%	16.9/sec	93.1
/wp-content/uploads/2018/02/newspaper-1.j	13750	2321	646	5091	200	54292	0.00%	16.9/sec	2578.7
/scripts/helpers.js	13750	141	49	277	1	9178	0.90%	16.9/sec	36.2
/security/login.do	13750	131	33	278	1	9102	0.51%	16.9/sec	87.3
/j_spring_security_check	13750	528	202	1336	3	21225	2.04%	16.9/sec	92.4
/chirpuser/list.do	13750	804	489	1855	3	17163	0.58%	16.9/sec	2224.6
TOTAL	206250	353	67	683	1	54292	0.95%	251.1/sec	10142.2

Include group name in label? Save Table Header

Graph Results

Name: Graph Results
Comments:
Write results to file / Read from file
Filename: LogDisplay Only, Errors, Successes,

Graphs to Display: Data, Average, Median, Deviation, Throughput

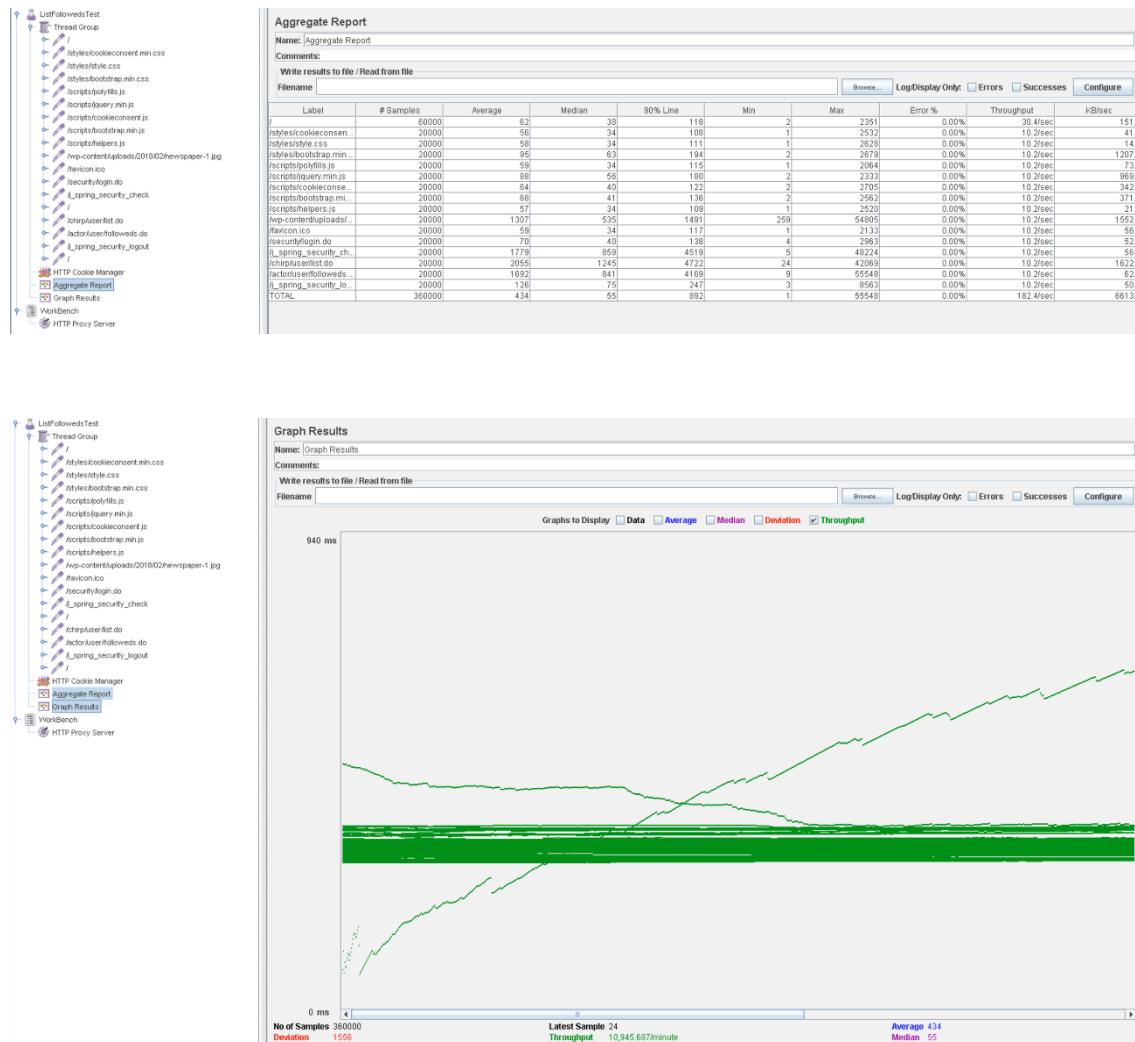
686 ms

0 ms

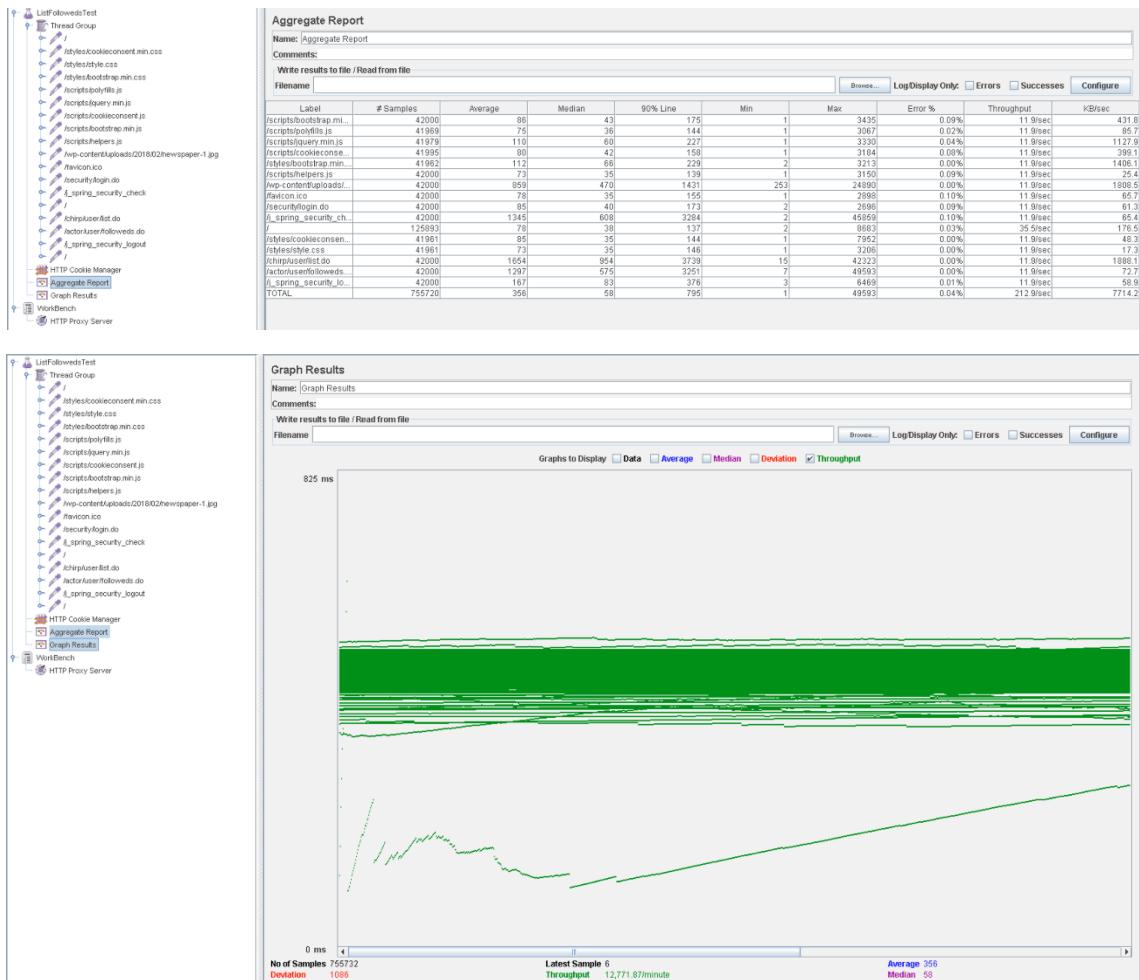
No of Samples: 206250 | Latest Sample: 59 | Throughput: 15,063.834/minute | Average: 353 | Median: 67 | Deviation: 1711

Caso de uso: listar Followeds.

Realizamos el caso de uso que comienza cuando el usuario inicia sesión, entra en la lista de chirps y acaba navegando hacia la lista de los usuarios a los que sigue. Este test de rendimiento fue realizado con 200 usuarios que realizan 100 veces el caso de uso y como vemos no tenemos errores ningunos.

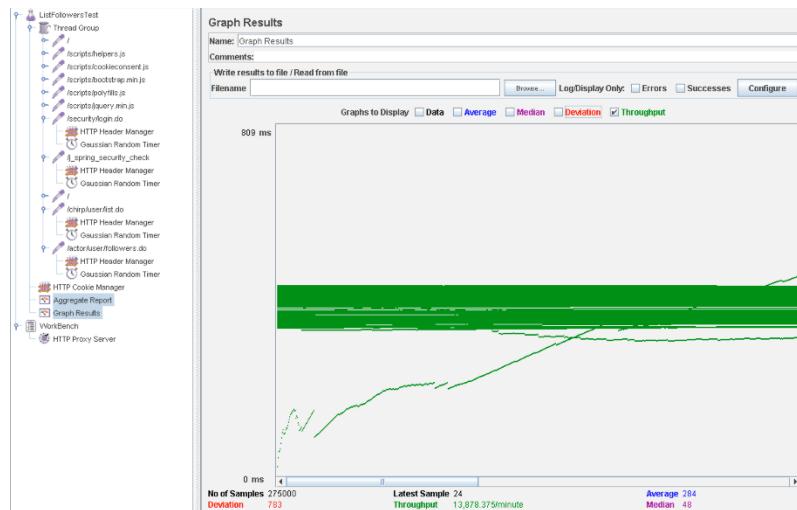
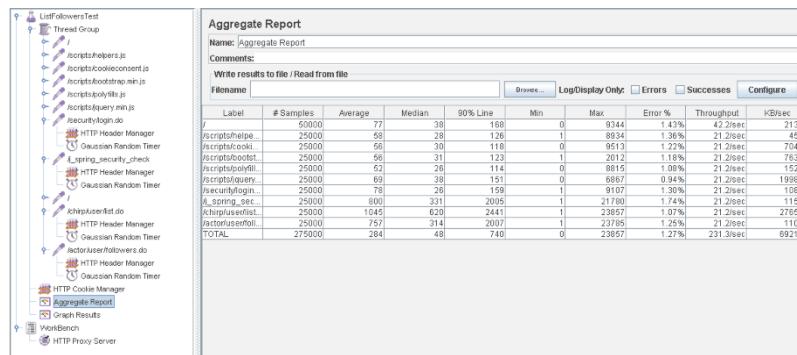


Aumentando los valores de usuarios a 210 y el número de veces que realiza cada uno el caso de uso será de 200, el porcentaje de errores es muy pequeño:

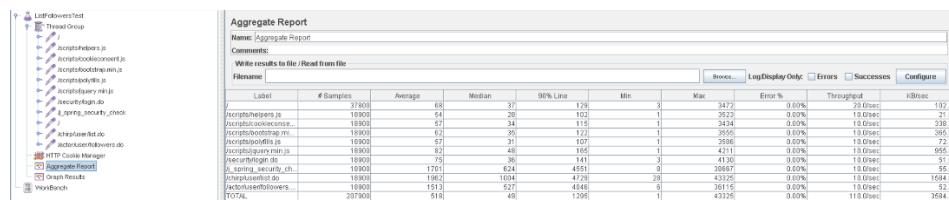


Caso de uso: listar Followers.

Realizamos el caso de uso que cubre la secuencia de pasos en la que el usuario inicia sesión, entra en la lista de chirps y navega hacia la lista de sus seguidores. Este test de rendimiento fue realizado con 250 usuarios con 100 acciones a la vez cada uno. Estos valores nos darán como resultado errores debido a un cuello de botella.



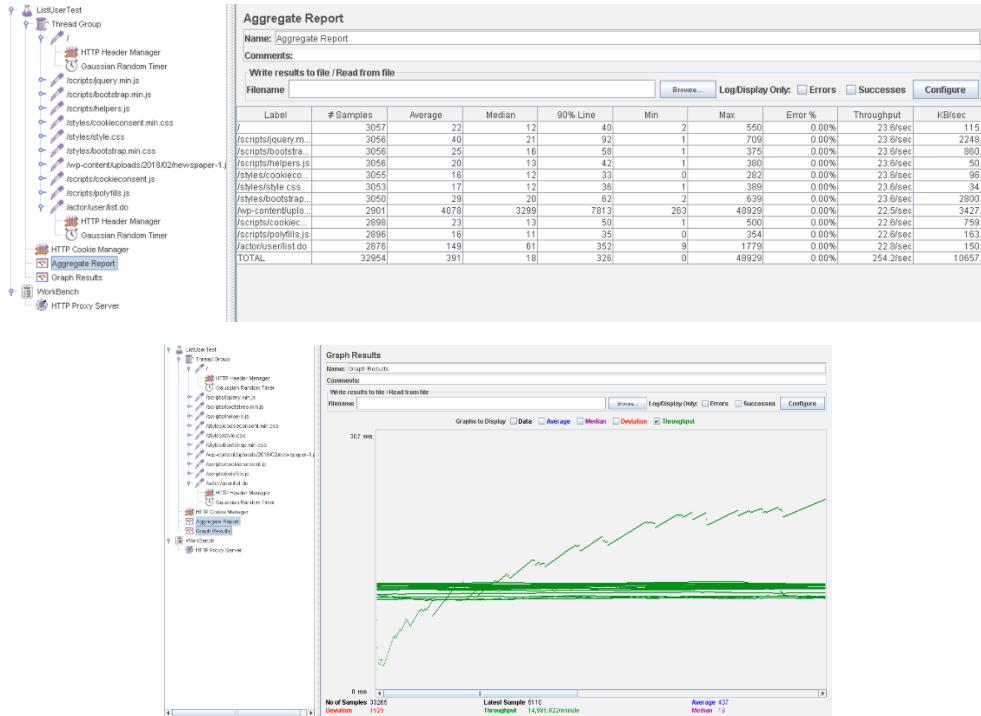
Aligerando los valores de usuarios a 210 y el número de acciones que se realizan a la vez a 90, no tenemos errores:



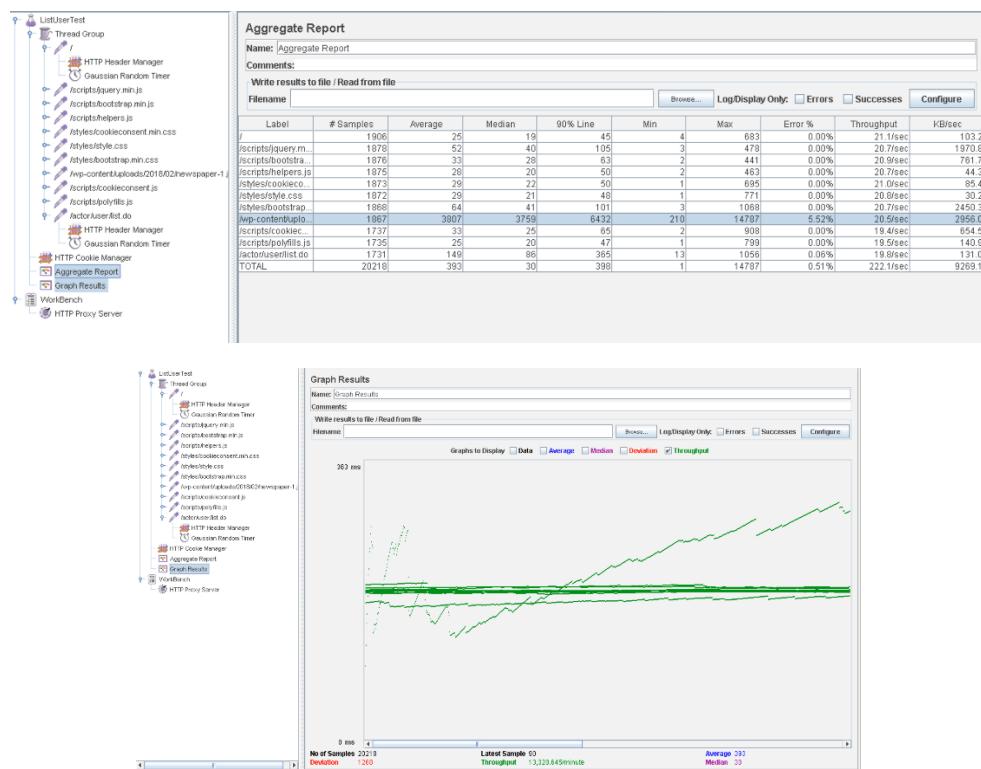


Caso de uso: listar User.

En este test, realizaremos el caso de uso en el que un usuario sin iniciar sesión accede a la lista de usuarios. En este caso hemos aumentado el número de veces que se realiza la acción hasta 400 y dejado en número de usuarios en 200:



Buscando la cantidad del *Loop Count* que empieza a dar fallos hemos encontrado el 600, donde la máquina empieza a resentirse y a mostrar sus debilidades:



Conclusiones:

Como conclusión, obtenemos que nuestro sistema es capaz de soportar 160 usuarios, del caso de uso Buscador de Newspaper, ya que el buscador de Article y Newspaper utiliza *like*, un operador de comparación por cadenas de caracteres, que es poco eficiente. El mínimo número usuarios concurrentes lo tenemos en el Dashboard por el motivo indicado anteriormente. Además, queremos destacar que el rendimiento del sistema ha mejorado bastante al usar índices en las queries, así como una paginación sobre los métodos de cada repositorio utilizando Pageable.

Además, destacamos que el principal cuello de botella ha sido la CPU. Para ello, hemos usado la herramienta performance, para monitorizar la interfaz de red, la memoria, el disco duro y la CPU.

