Ítem 5 – Performance Tests

Diseño y Pruebas
Grado de Ingeniería del Software
Curso 3

Armando Garrido Castro
César García Pascual
Manuel Enrique Pérez Carmona
Jorge Puente Zaro
Pablo Tabares García
Rafael Trujillo González

Fecha: 13 de abril de 2018

Contenido

Introducción **3**

Tests de rendimiento del nivel C **4**

2.1. UC1- Registrarse como usuario 4

2.2. UC2- Listar un newspaper y navegar hacia sus artículos 5

2.3. UC3- Listar los usuarios del sistema y navegar hacia sus perfiles 6

2.4. UC4- Listar periódicos, buscar periódicos por una palabra clave y que estos se listen 7

2.5. UC5- Listar artículos, buscar artículos por una palabra clave y que estos se listen 8

2.6. UC6-Un usuario crea un periódico y tiene la opción de que este sea público o dejarlo privado 9

2.7. UC7- Crear un artículo para ese periódico y ponerlo en final mode para que se publique 10

2.8. UC8- Un admin marca como inapropiado un artículo, chirp o newspaper 11

2.9. UC09-Un admin visualiza su dashboard 12

Tests de rendimiento del nivel B **13**

3.1. UC10- Un usuario publica un chirp 13

3.2. UC11- Un usuario sigue a otro 14

3.3. UC12- Un usuario lista los usuarios que sigue y que le siguen 15

3.4. UC13- Visualizar los chirps de todos los usuarios que sigue 16

3.5. UC14- Un admin crea, edita y borra palabras tabú 17

3.6. UC15-Un admin lista los periódicos con palabras tabú 18

3.7. UC18-Un admin lista los artículos con palabras tabú 19

3.8. UC18-Un admin lista los chirp con palabras tabú 20

Tests de rendimiento del nivel A **21**

4.1. UC18 - Registrarse como customer 21

4.2. UC19 - Un customer se suscribe a periódicos privados 22

Conclusión **23**

1. Introducción

Detrás de todo proyecto no pueden faltar sus correspondientes pruebas de aceptación y calidad. Entre ellas encontramos los *tests* de rendimiento. Este tipo de pruebas nos permite conocer en todo momento los límites de nuestro servidor. En un entorno de desarrollo, es común que naveguemos a través de nuestro sistema de información web sin ningún tipo de problema y a una velocidad más que decente.

Sin embargo, es un hecho que, una vez que nuestro proyecto sea desplegado en una red tan amplia como es Internet, nuestro sistema de información será utilizado de forma concurrente por cientos e incluso miles de usuarios. La cuestión es: ¿cuánta carga de trabajo concurrente podrá soportar nuestro servidor? Surgen para ello las pruebas de rendimiento, las cuales nos permitirán prevenir el máximo de nuestro sistema según sus capacidades y podremos explorar, analizar y detectar qué casos de uso son llevados a cabo de forma rápida y cuáles no han sido diseñados de forma eficaz.

2. Tests de rendimiento del nivel C

2.1. UC1- Registrarse como usuario

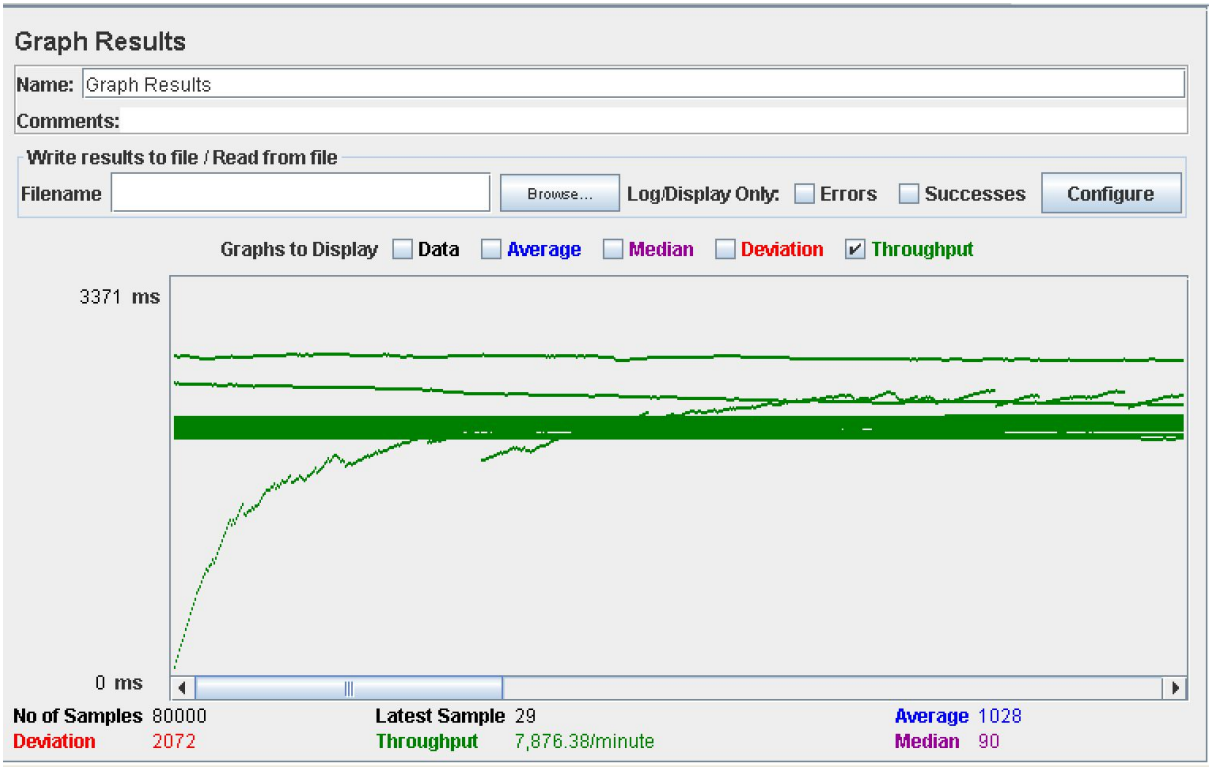
Parámetros:

Número de usuarios: 50

Número de bucles: 120

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput
/	20000	65	23	115	1	3817	0.20%	32.9/sec
/register/use...	40000	995	66	3125	1	38952	50.10%	65.7/sec
/user-displa...	20000	2054	1235	4894	2	38359	0.18%	32.9/sec
TOTAL	80000	1028	90	3211	1	38952	25.14%	131.3/sec



Conclusiones:

En esta ocasión utilizaremos una carga total de 50 usuarios concurrentes realizando 120 veces la misma acción. Como podemos ver, en el percentil 90 se tarda en torno a 3 segundos para la carga del formulario de registro y casi 5 segundos en el registro y redirección al perfil del usuario registrado. En total, dicho proceso tarda 8134 ms en realizarse, un número bastante aceptable y positivo para la operación realizada.

2.2. UC2- Listar un newspaper y navegar hacia sus artículos

Parámetros:

Número de usuarios: 20

Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/styles/displa...	2000	38	31	67	2	1012	0.00%	17.4/sec	51.2
/styles/comm...	2000	38	32	66	2	543	0.00%	17.4/sec	29.5
/scripts/jquery...	2000	140	119	283	8	931	0.00%	17.4/sec	7653.4
/favicon.ico	2000	41	35	74	3	552	0.00%	17.4/sec	564.6
/newspaper/li...	2000	180	153	320	17	1627	0.00%	17.4/sec	295.8
/newspaper/d...	2000	214	176	396	14	1378	0.00%	17.4/sec	271.6
/article/displa...	2000	221	187	405	14	1392	0.00%	17.4/sec	286.5
TOTAL	14000	125	74	305	2	1627	0.00%	121.7/sec	9140.9

Conclusiones:

En esta ocasión utilizaremos una carga total de 20 usuarios concurrentes realizando 100 veces la misma acción. Podemos comprobar que las tres operaciones más notorias en este caso de uso son las tres de listado (listado de Newspaper, vista detallada de un Newspaper y vista detallada de un artículo). Sin embargo, cada uno de éstos presenta tiempos de carga muy pequeños al basarse en simples vistas de listado. Por ello, el total de tiempo de carga es de 1611 ms, lo que equivale a un tiempo medio inferior a los dos segundos, una cantidad más que aceptable.

2.3. UC3- Listar los usuarios del sistema y navegar hacia sus perfiles

Parámetros:

Número de usuarios: 170

Número de bucles: 145

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	49300	27	16	45	2	3809	0.00%	61.8/sec	900.5
/security/lo...	24650	28	17	42	3	2838	0.00%	31.0/sec	472.1
/scripts/md...	24650	23	14	41	1	1576	100.00%	31.0/sec	34.6
/j_spring_s...	24650	20	14	36	1	1747	100.00%	31.0/sec	8.9
/user-list.do	24650	1654	1087	3936	9	21959	0.00%	31.0/sec	489.2
/user-displ...	24650	1824	1230	4230	10	19454	0.00%	31.0/sec	479.6
TOTAL	172550	515	22	1799	1	21959	28.57%	216.4/sec	2381.5

Conclusiones:

En esta ocasión utilizaremos una carga total de 170 usuarios concurrentes realizando 145 veces la misma acción. Podemos ver que en esta ocasión habrá dos operaciones que tomarán tiempos muy similares. La primera, el listado de usuarios (en torno a 3900 ms en el percentil 90). El segundo, la visualización del perfil concreto de un usuario (4200 ms aproximadamente) . Esto se debe a que ambos realizan operaciones similares de listado, por lo que el tiempo requerido será similar. En total, el proceso tardará 10129 ms en ser realizado.

2.4. UC4- Listar periódicos, buscar periódicos por una palabra clave y que estos se listen

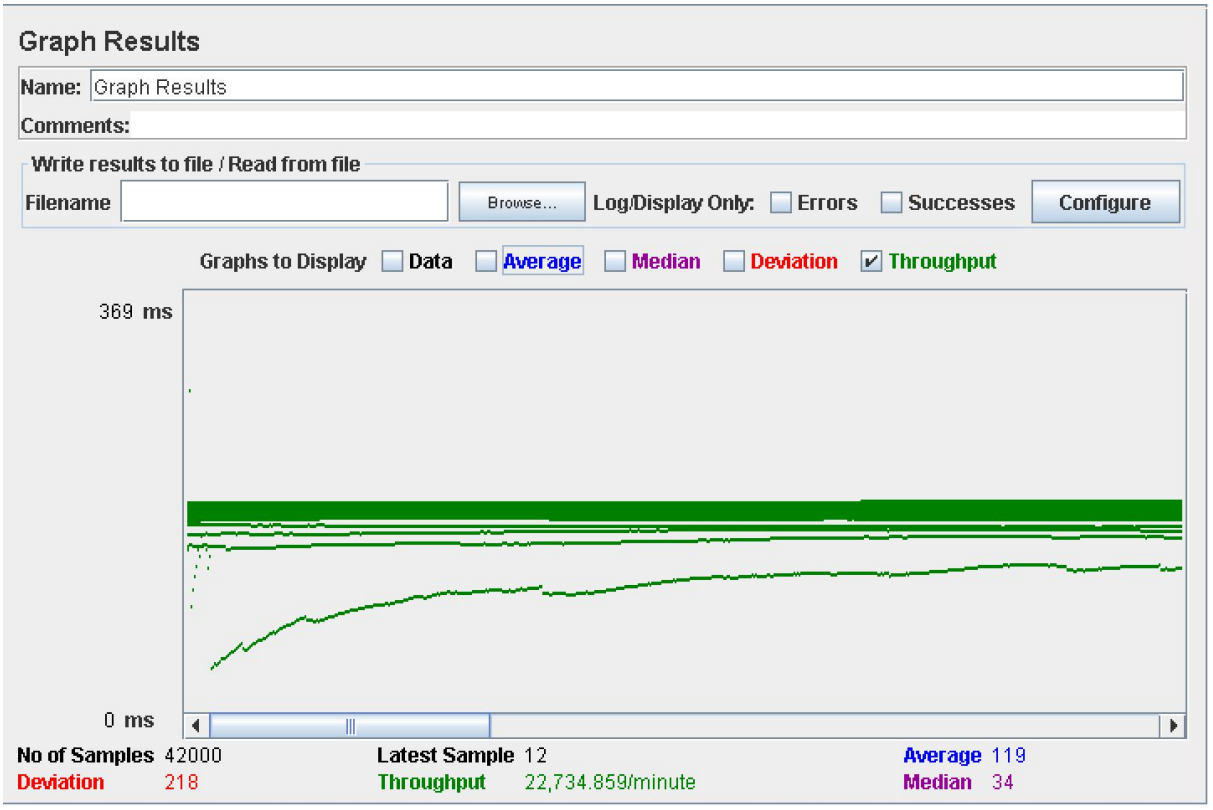
Parámetros:

Número de usuarios: 150

Número de bucles: 40

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6000	33	22	56	3	913	0.00%	55.0/sec	798.5
/scripts/jquery-ui.js	6000	132	104	266	5	1521	0.00%	55.0/sec	24199.3
/styles/displaytag.css	6000	24	17	46	1	924	0.00%	55.0/sec	161.8
/styles/common.css	6000	24	17	42	1	736	0.00%	55.0/sec	93.3
/favicon.ico	6000	27	21	50	2	842	0.00%	55.0/sec	1785.0
/newspaper/list.do	12000	297	162	751	5	2994	0.00%	108.4/sec	1628.7
TOTAL	42000	119	34	340	1	2994	0.00%	378.9/sec	28228.1



Conclusiones:

Como se puede ver se han usado 150 usuarios concurrentes haciendo 40 veces la misma acción, y en el percentil 90% se puede ver que el número más grande corresponde al listado con un total de 751 ms, que es un buen número. Así que estamos satisfechos por el resultado. En la gráfica no se puede ver nada fuera de lo común.

2.5. UC5- Listar artículos, buscar artículos por una palabra clave y que estos se listen

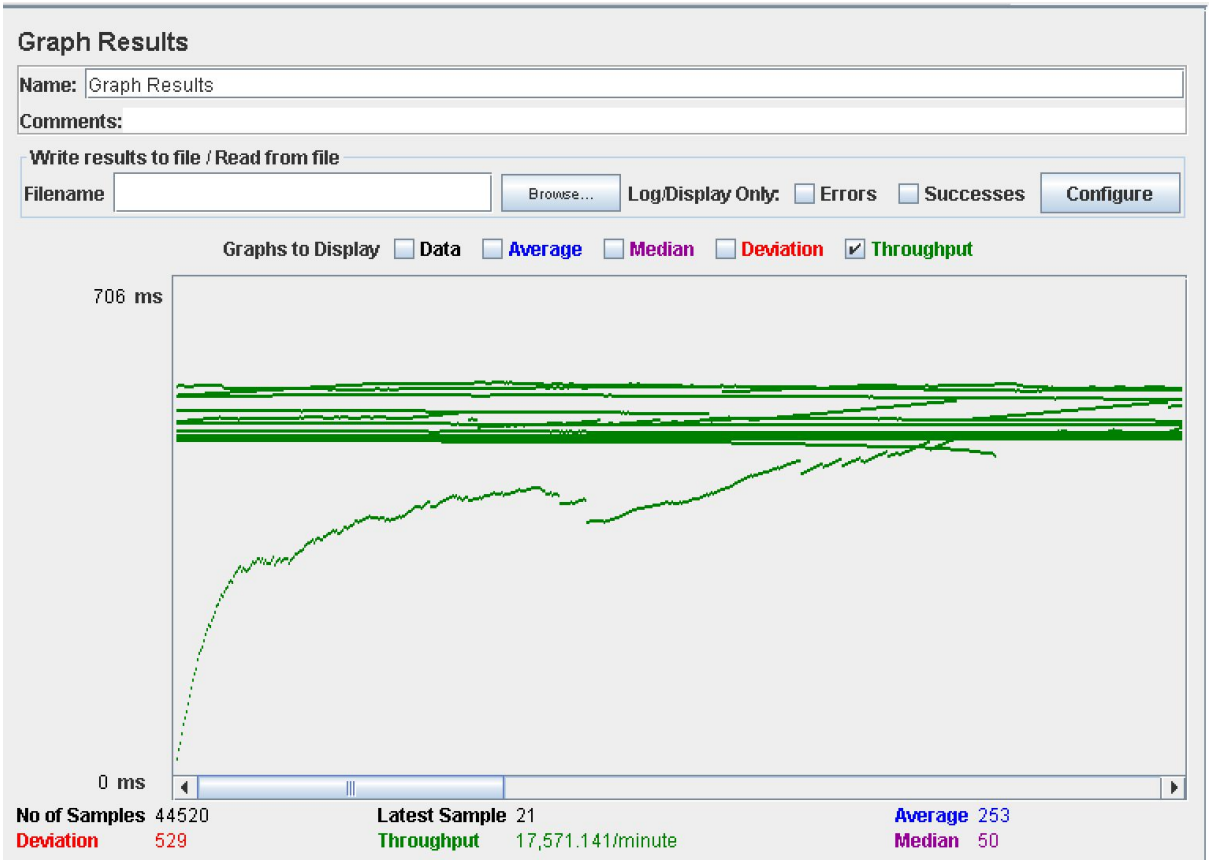
Parámetros:

Número de usuarios: 150

Número de bucles: 40

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6360	50	34	101	2	2092	0.00%	42.3/sec	617.4
/scripts/jqu...	6360	217	170	445	5	2115	0.00%	42.3/sec	18611.5
/styles/com...	6360	36	24	67	1	1191	0.00%	42.3/sec	71.8
/styles/disp...	6360	35	23	66	1	1335	0.00%	42.3/sec	124.5
/favicon.ico	6360	38	27	73	2	904	0.00%	42.3/sec	1373.1
/article/list.do	12720	696	426	1682	5	8052	0.00%	83.7/sec	1258.3
TOTAL	44520	253	50	768	1	8052	0.00%	292.9/sec	21819.8



Conclusiones:

Como se puede ver se han usado 150 usuarios concurrentes haciendo 40 veces la misma acción, y en el percentil 90% la suma total no alcanza ni los 3s, con lo cual es un buen resultado, ya que primero tiene que hacer una búsqueda y luego listar los artículos. En referencia a la gráfica el rendimiento es bastante bueno, por lo anteriormente comentado.

2.6. UC6-Un usuario crea un periódico y tiene la opción de que este sea público o dejarlo privado

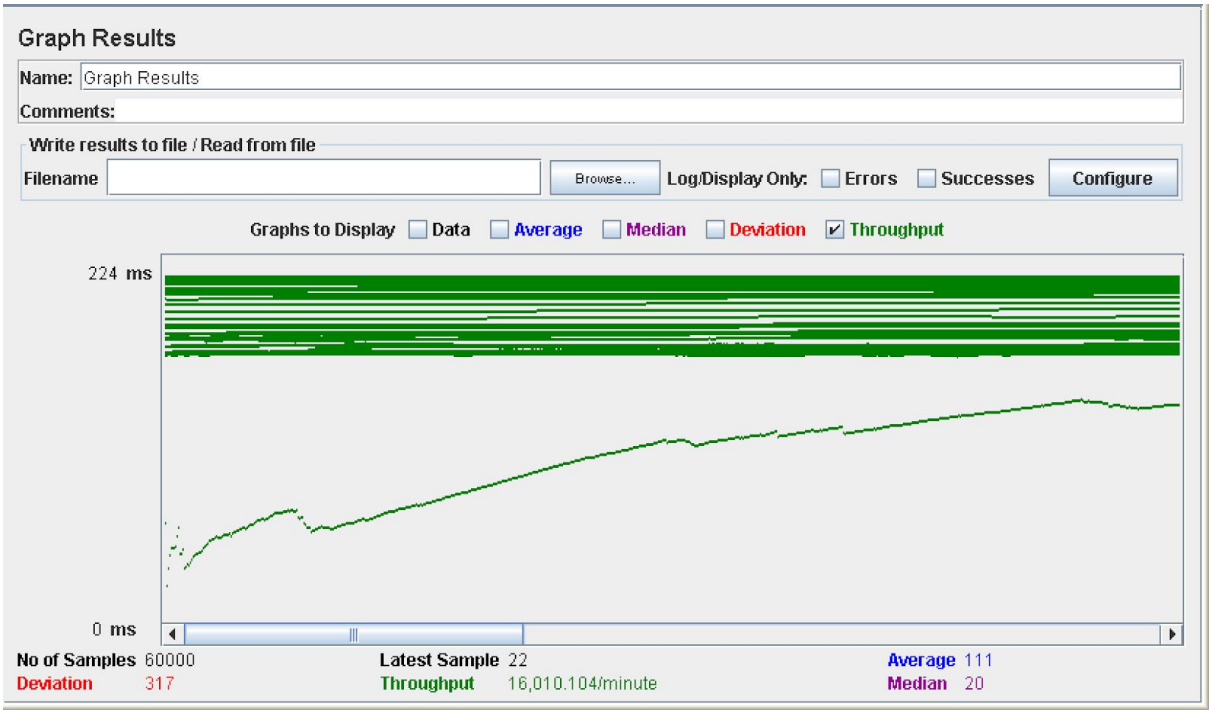
Parámetros:

Número de usuarios: 150

Número de bucles: 50

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throug...	KB/sec
/	15000	56	16	131	2	2666	0.00%	67.2/sec	977.8
/security/login.do	7500	74	19	161	3	2462	0.00%	33.8/sec	515.0
/scripts/md5.min.js	7500	62	15	144	1	2276	100.00%	33.8/sec	37.5
/j_spring_security_check	7500	56	14	130	1	2195	100.00%	33.8/sec	9.5
/user/newspaper/create.do	7500	100	33	225	3	2501	0.00%	33.8/sec	520.1
/user/newspaper/save.do	7500	52	12	117	1	2678	100.00%	33.8/sec	9.5
/newspaper/list.do	7500	430	102	1381	6	7099	0.00%	33.8/sec	535.7
TOTAL	60000	111	20	221	1	7099	37.50%	266.8/s...	2577.0



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 50 veces la misma acción. Realizaremos tres operaciones distintas: creación de un *Newspaper*, guardado de éste mismo y listado del total. Podemos comprobar que la mayor carga de tiempo se produce en el listado final donde somos redirigidos, con un total de 1381 milisegundos en el percentil 90. Se trata de una operación simple, por lo que tanto la creación del formulario como el guardado apenas conllevan tiempo. En total, tardará 2289 milisegundos en el percentil 90.

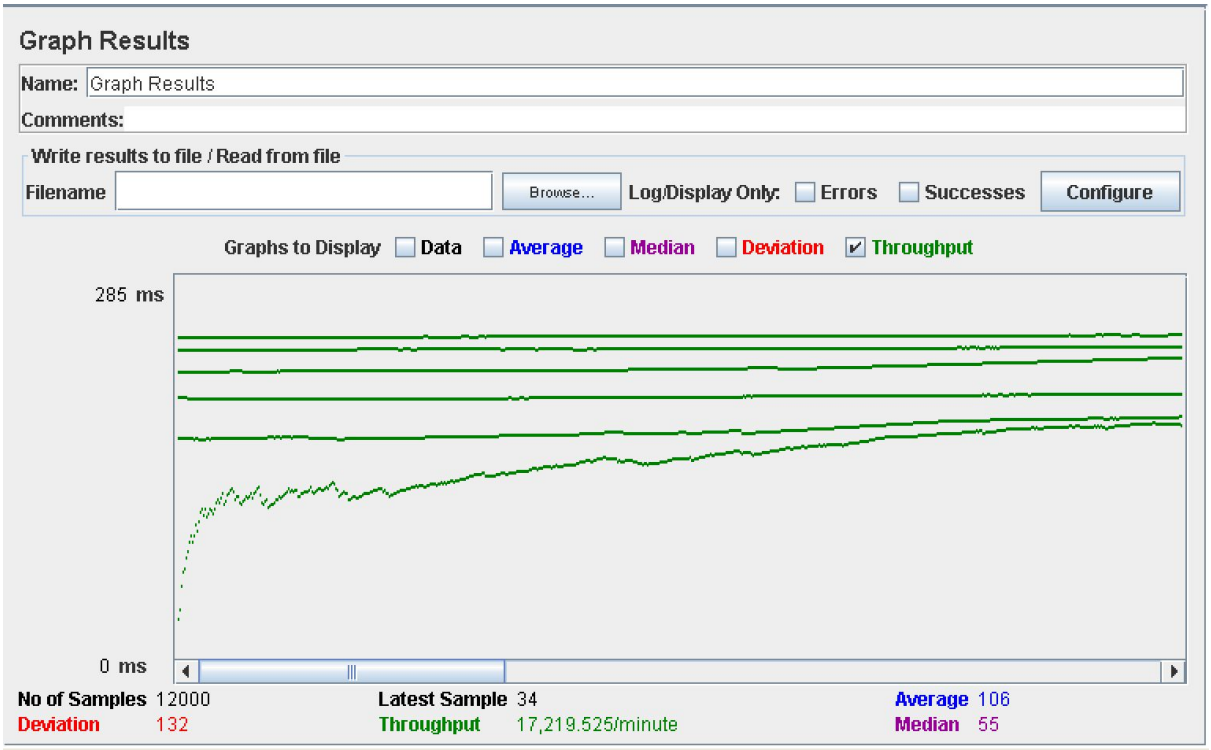
2.7. UC7- Crear un artículo para ese periódico y ponerlo en final mode para que se publique

Parámetros:

Número de usuarios: 30

Número de bucles: 100

Resultados:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/newspaper/list.do	3000	169	121	349	13	1691	0.00%	71.8/sec	1243.4
/newspaper/displa...	3000	184	144	372	12	1654	0.00%	71.9/sec	1120.1
/user/article/save.do	3000	22	19	35	2	433	100.00%	71.9/sec	20.1
/user/article/list.do	3000	47	43	71	8	286	0.00%	71.8/sec	1105.4
TOTAL	12000	106	55	253	2	1691	25.00%	287.0/sec	3484.0

Conclusiones:

En esta ocasión utilizaremos una carga total de 30 usuarios concurrentes realizando 100 veces la misma acción. Tendremos un total de cuatro operaciones: acceder a la lista de Newspaper, ver uno en concreto, guardar un artículo y visualizarlo en la lista de artículos a donde somos redirigidos. Como podemos comprobar, los dos primeros procesos tienen una mayor carga de tiempo notable que el resto (en torno a 10 veces más lento). Esta carga de trabajo es debida al proceso de carga de imágenes, la cual requerirá mucho más tiempo por cada uno de los elementos que visualicemos en el listado. En total, el proceso lleva unos 827 ms según el percentil 90, cantidad muy positiva que no sobrepasa ni un segundo de carga.

2.8. UC8- Un admin marca como inapropiado un artículo, chirp o newspaper

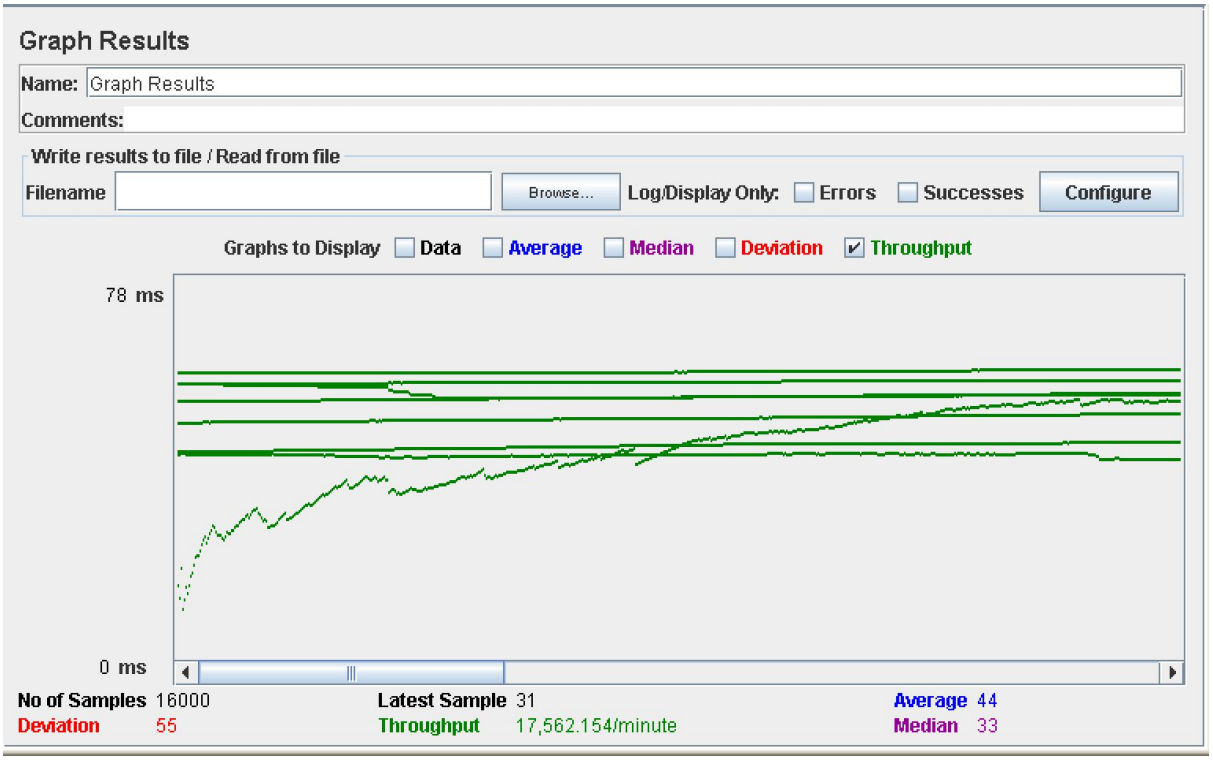
Parámetros:

Número de usuarios: 20

Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	4000	33	26	53	3	914	0.00%	73.3/sec	1067.3
/security/lo...	2000	34	27	54	4	885	0.00%	36.7/sec	559.5
/scripts/md...	2000	29	23	50	1	521	100.00%	36.7/sec	40.8
/j_spring_s...	2000	29	24	50	2	411	100.00%	36.7/sec	10.3
/admin/chir...	4000	64	52	99	12	2047	0.00%	73.3/sec	1128.4
/admin/chir...	2000	67	52	99	11	1158	0.00%	36.7/sec	565.1
TOTAL	16000	44	33	76	1	2047	25.00%	292.7/sec	3362.5



Conclusiones:

En esta ocasión utilizaremos una carga total de 20 usuarios concurrentes realizando 100 veces la misma acción. Como podremos observar, ninguna de las operaciones de este caso de uso supera la centésima de segundo. Esto es debido a la sencillez de la operación, la cual marca como inapropiado un *Chirp* o Chirrido y nos redirige al mismo listado, ahorrando tiempo gracias a la caché. En total, el proceso se realiza en un total de 404 ms (ni medio segundo).

2.9. UC09-Un admin visualiza su dashboard

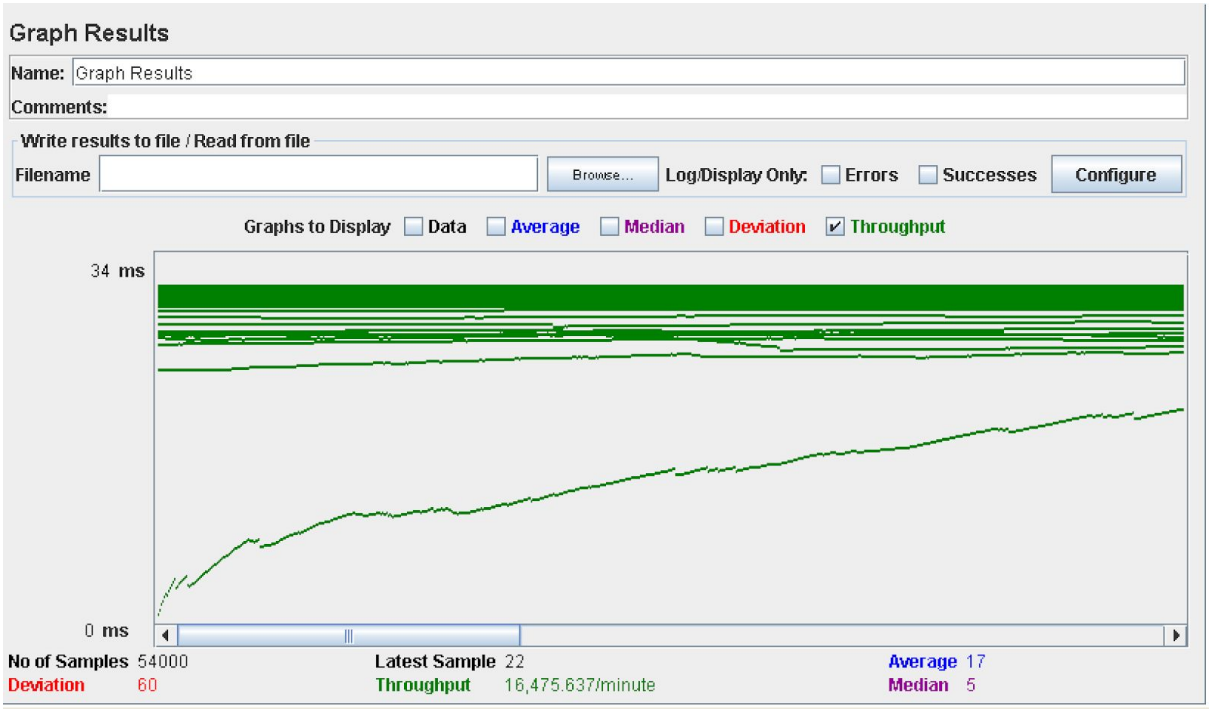
Parámetros:

Número de usuarios: 150

Número de bucles: 60

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	18000	14	5	29	2	1162	0.00%	92.2/sec	1342.3
/security/logi...	9000	19	5	32	3	1730	0.00%	46.5/sec	708.0
/scripts/md5...	9000	13	3	26	1	1404	100.00%	46.5/sec	51.6
/j_spring_se...	9000	12	3	25	1	1501	100.00%	46.5/sec	13.0
/admin/dash...	9000	30	9	53	5	1862	0.00%	46.5/sec	715.6
TOTAL	54000	17	5	31	1	1862	33.33%	274.6/sec	2797.7



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 60 veces la misma acción. El proceso de visualizado de *dashboard* es bastante simple, ya que la vista únicamente muestra tablas con datos estadísticos, con ausencia de imágenes u otros elementos que pudieran retrasar el tiempo de carga. Así, el tiempo en el percentil 90 es de 53 milisegundos, una cantidad sorprendente y muy positiva. En total, el caso de uso tardará un total de 165 ms.

3. Tests de rendimiento del nivel B

3.1. UC10- Un usuario publica un chirp

Parámetros:

Número de usuarios: 150
Número de bucles: 70

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Through...	KB/sec
/	31500	54	40	91	2	1617	0.00%	95.3/sec	1383.8
/scripts/jquery-ui.js	10500	314	226	687	6	2078	0.00%	32.1/sec	14102.4
/styles/common.css	10500	47	35	84	2	1325	0.00%	32.1/sec	54.3
/styles/displaytag.css	10500	49	35	82	2	1581	0.00%	32.1/sec	94.3
/security/login.do	10500	54	41	90	3	1802	0.00%	32.1/sec	488.3
/scripts/md5.min.js	10500	55	38	91	1	1590	100.00%	32.1/sec	35.6
/j_spring_security_ch...	10500	53	38	89	2	1850	100.00%	32.1/sec	9.0
/user/chirp/create.do	10500	105	78	179	7	1577	0.00%	32.1/sec	493.3
/user/chirp/save.do	10500	50	37	88	2	1590	100.00%	32.0/sec	9.0
TOTAL	115500	81	44	148	1	2078	27.27%	349.4/sec	16525.5

Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 70 veces la misma acción. Podemos comprobar que las principales operaciones de este caso de uso duran poco tiempo. Esto se debe al simple proceso de creación de un *Chirp* o chirrido, la cual se basa en la creación y guardado a través de un breve formulario. En total, el proceso dura en torno a 1481 ms en el percentil 90, lo equivalente a un segundo y medio.

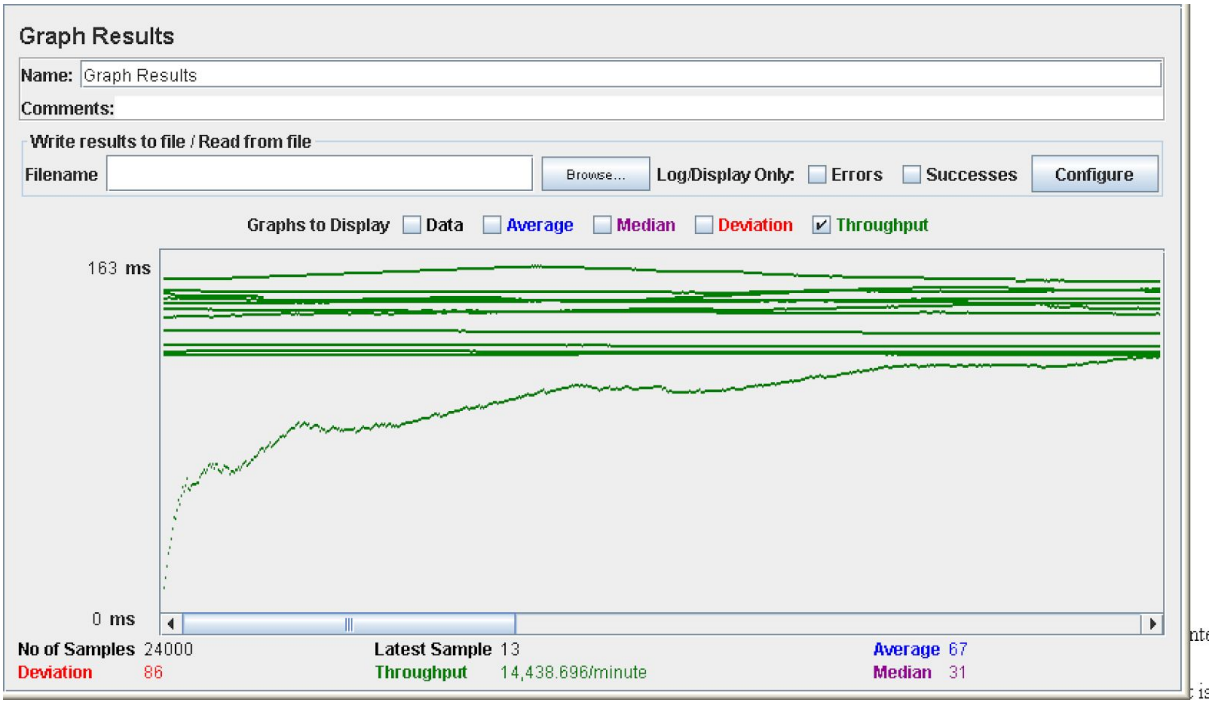
3.2. UC11- Un usuario sigue a otro

Parámetros:

Número de usuarios: 20
Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	8000	24	22	37	3	427	0.00%	80.2/sec	1166.4
/security/logi...	2000	24	23	38	4	297	0.00%	20.1/sec	305.8
/scripts/md5...	2000	21	20	34	2	411	100.00%	20.1/sec	22.3
/j_spring_se...	2000	21	20	33	2	145	100.00%	20.1/sec	5.6
/user-list.do	2000	185	163	328	11	664	0.00%	20.1/sec	317.1
/user-displa...	4000	183	162	318	17	1181	0.00%	40.2/sec	631.0
/user/follow....	2000	47	43	69	14	407	0.00%	20.1/sec	309.4
/j_spring_se...	2000	45	41	67	4	415	0.00%	20.1/sec	296.3
TOTAL	24000	67	31	190	2	1181	16.67%	240.6/sec	3050.1



Conclusiones:

En esta ocasión utilizaremos una carga total de 20 usuarios concurrentes realizando 100 veces la misma acción. Como podremos observar, la mayor carga de trabajo se da en las operaciones previas al propio *follow*, dado que este proceso solamente se encarga de asignar al usuario a la lista de seguidos de otra persona y redireccionar a la misma página. Así, el listado de usuarios y el *display* consumen 328 y 318 ms en el percentil 90 respectivamente, mientras que la operación *follow* tan solo lleva 69 ms en realizarse. En total, este caso de uso tarda 924 ms: poco menos de un segundo.

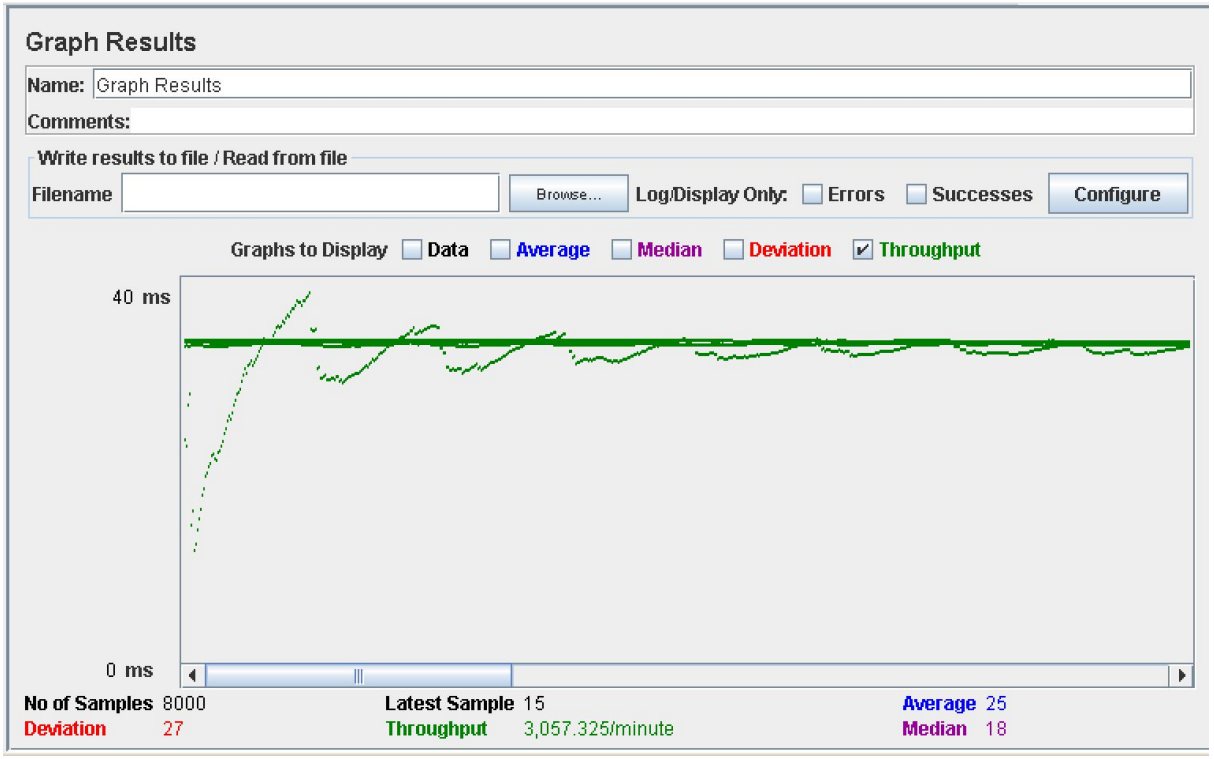
3.3. UC12- Un usuario lista los usuarios que sigue y que le siguen

Parámetros:

Número de usuarios: 40
Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/user-display.do	8000	25	18	37	4	508	0.00%	51.0/sec	831.9
TOTAL	8000	25	18	37	4	508	0.00%	51.0/sec	831.9



Conclusiones:

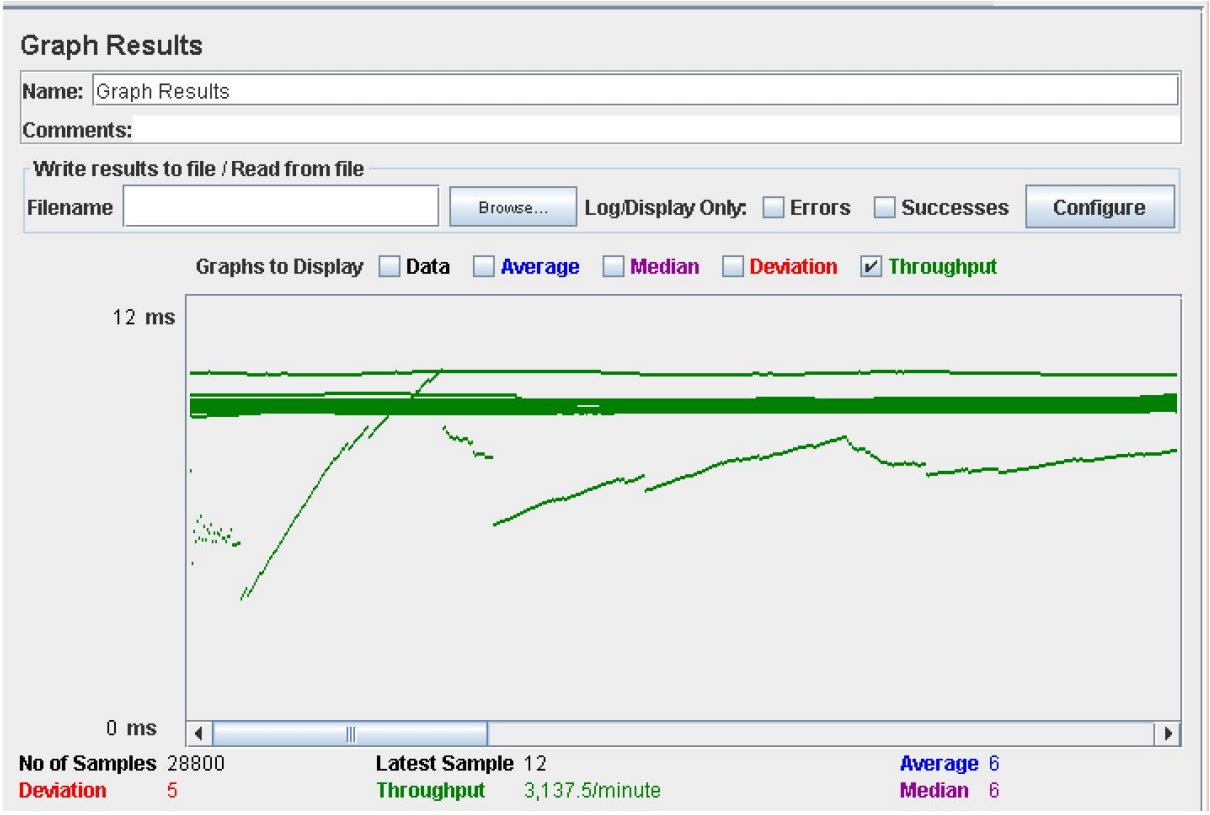
En esta ocasión utilizaremos una carga total de 40 usuarios concurrentes realizando 100 veces la misma acción. Gracias a nuestra vista de *display* de usuario, la opción de observar los usuarios seguidores y seguidos de uno de éstos es simplificada dado que alcanzamos a ver ambas en la misma vista que ya hemos observado en casos de uso anteriores. Así, se trata de un proceso muy simple donde solo toma lugar la visualización de los detalles del usuario en cuestión, tomando un total de 37 ms en el percentil 90.

3.4. UC13- Visualizar los chirps de todos los usuarios que sigue

Parámetros:

Número de usuarios: 30
Número de bucles: 120

Resultados:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throu...	KB/sec
/	10800	6	5	10	2	258	0.00%	19.6/s...	285.3
/security/login.do	3600	6	6	10	3	245	0.00%	6.6/sec	100.3
/scripts/md5.min.js	3600	3	3	5	1	54	100.00%	6.6/sec	7.3
/j_spring_security_check	3600	4	3	6	1	281	100.00%	6.6/sec	1.8
/user/chirp/timeline.do	3600	10	9	15	6	224	0.00%	6.6/sec	101.4
/j_spring_security_logout	3600	10	8	14	5	277	0.00%	6.6/sec	97.3
TOTAL	28800	6	6	11	1	281	25.00%	52.3/s...	590.8

Conclusiones:

En esta ocasión utilizaremos una carga total de 30 usuarios concurrentes realizando 120 veces la misma acción. Aunque el proceso pueda parecer más complejo, el caso de uso que muestra la timeline de un usuario es tan simple como el listado de todos los *Chirps* de los usuarios que éste sigue en el sistema. De este modo, ninguno de los procesos de este caso de uso supera los 20 ms en el percentil 90, requiriendo un total de 60 milisegundos para que la acción se complete con éxito.

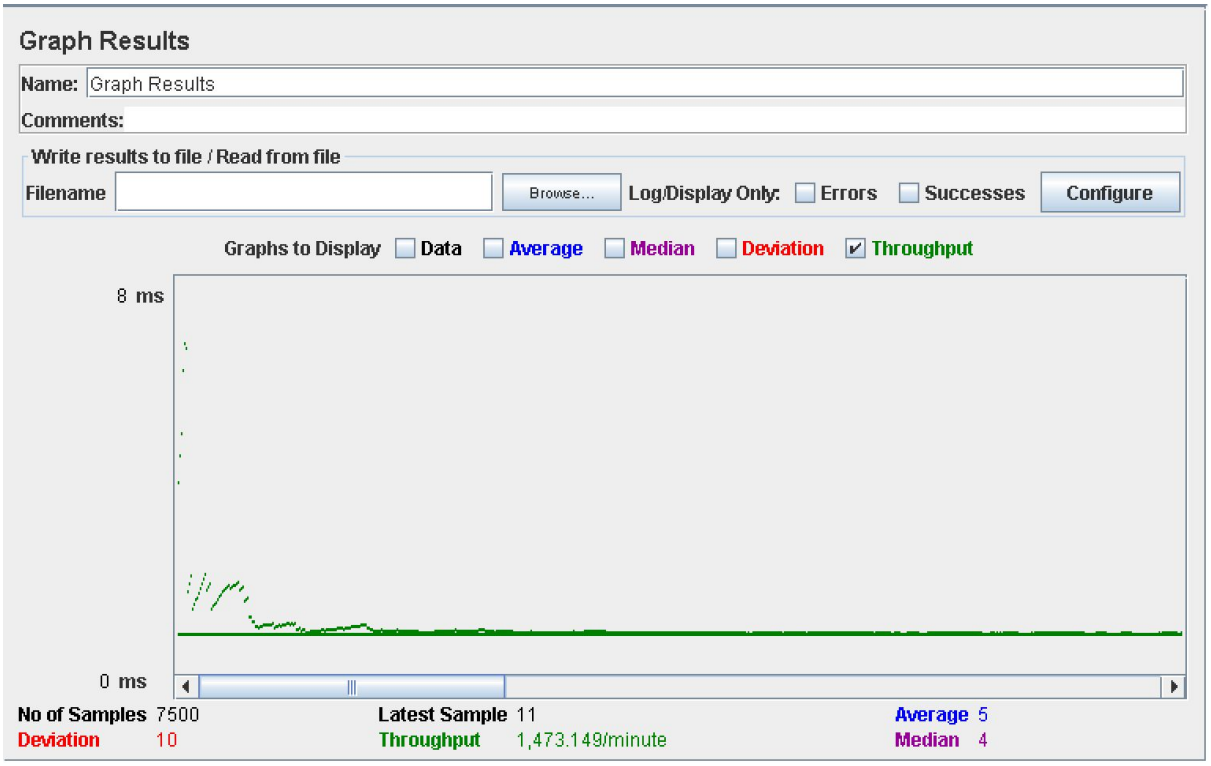
3.5. UC14- Un admin crea, edita y borra palabras tabú

Parámetros:

Número de usuarios: 20
Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1500	5	5	7	3	401	0.00%	4.9/sec	75.5
/j_spring_security_check	1500	3	3	4	1	397	100.00%	4.9/sec	1.4
/	1500	4	5	6	2	22	0.00%	4.9/sec	71.7
/systemConfig/edit.do	1500	8	8	10	3	412	0.00%	4.9/sec	75.9
/systemConfig/save.do	1500	3	3	4	1	21	100.00%	4.9/sec	1.4
TOTAL	7500	5	4	8	1	412	40.00%	24.6/sec	224.8



Conclusiones:

En esta ocasión utilizaremos una carga total de 20 usuarios concurrentes realizando 100 veces la misma acción. El proceso de creación, edición y borrado de palabras tabú es bastante simple (limitándose a las funciones básicas de las operaciones CRUD), por lo que el tiempo en el percentil 90 es bastante positivo. En total, este caso de uso tardará 31 milisegundos en ser procesado.

3.6. UC15-Un admin lista los periódicos con palabras tabú

Parámetros:

Número de usuarios:15

Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1500	18	18	30	3	222	0.00%	99.1/sec	1517.6
/j_spring_security_check	1500	16	16	27	1	56	100.00%	99.1/sec	27.8
/	1500	18	18	29	3	213	0.00%	99.0/sec	1438.3
/admin/newspaper/taboo-lis...	1500	39	34	52	6	883	0.00%	98.9/sec	1521.4
TOTAL	6000	23	20	39	1	883	25.00%	394.9/sec	4491.7

Conclusiones:

Como se puede ver se han utilizado 15 administradores concurrentes y han hecho está acción 100 veces. El listado de palabras tabú carece de imágenes y dispone de pocas columnas que mostrar, por lo que el tiempo tardado en mostrar dicha operación es muy positivo (52 milisegundos en el percentil 90). En total, este caso de uso tardará 138 ms en ser procesado.

3.7. UC18-Un admin lista los artículos con palabras tabú

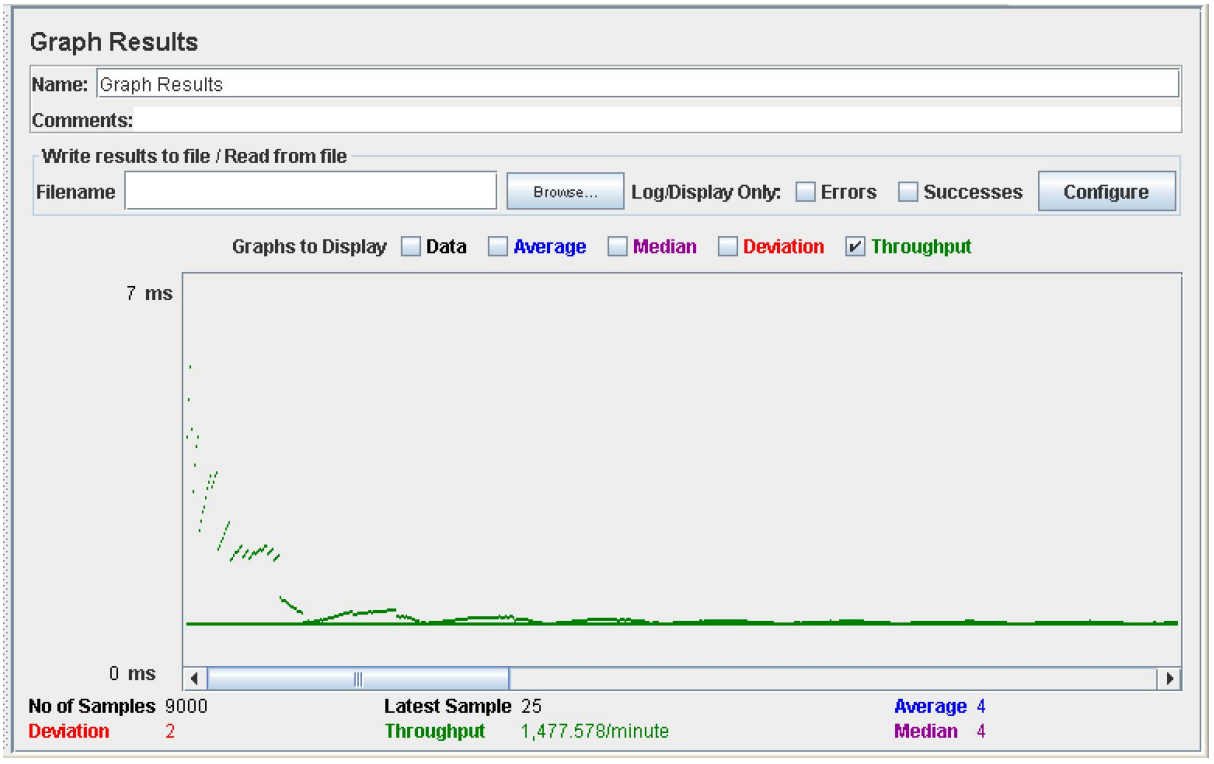
Parámetros:

Número de usuarios:15

Número de bucles: 120

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/security/login.do	1800	5	5	7	3	30	0.00%	4.9/sec	75.7
/scripts/md5.min.js	1800	2	3	4	1	39	100.00%	4.9/sec	5.5
/j_spring_security_check	1800	2	3	4	1	16	100.00%	4.9/sec	1.4
/	1800	4	5	6	3	20	0.00%	4.9/sec	71.8
/admin/article/taboo-list.do	1800	8	8	11	2	46	0.00%	4.9/sec	76.1
TOTAL	9000	4	4	8	1	46	40.00%	24.6/sec	229.6



Conclusiones:

En esta ocasión utilizaremos una carga total de 15 usuarios concurrentes realizando 120 veces la misma acción. El tiempo de respuesta es bastante bueno, ya que el proceso solo se encargará de visualizar los distintos artículos que están registrados con palabras tabú. En total, el proceso durará 32 milisegundos en el percentil 90.

3.8. UC18-Un admin lista los chirp con palabras tabú

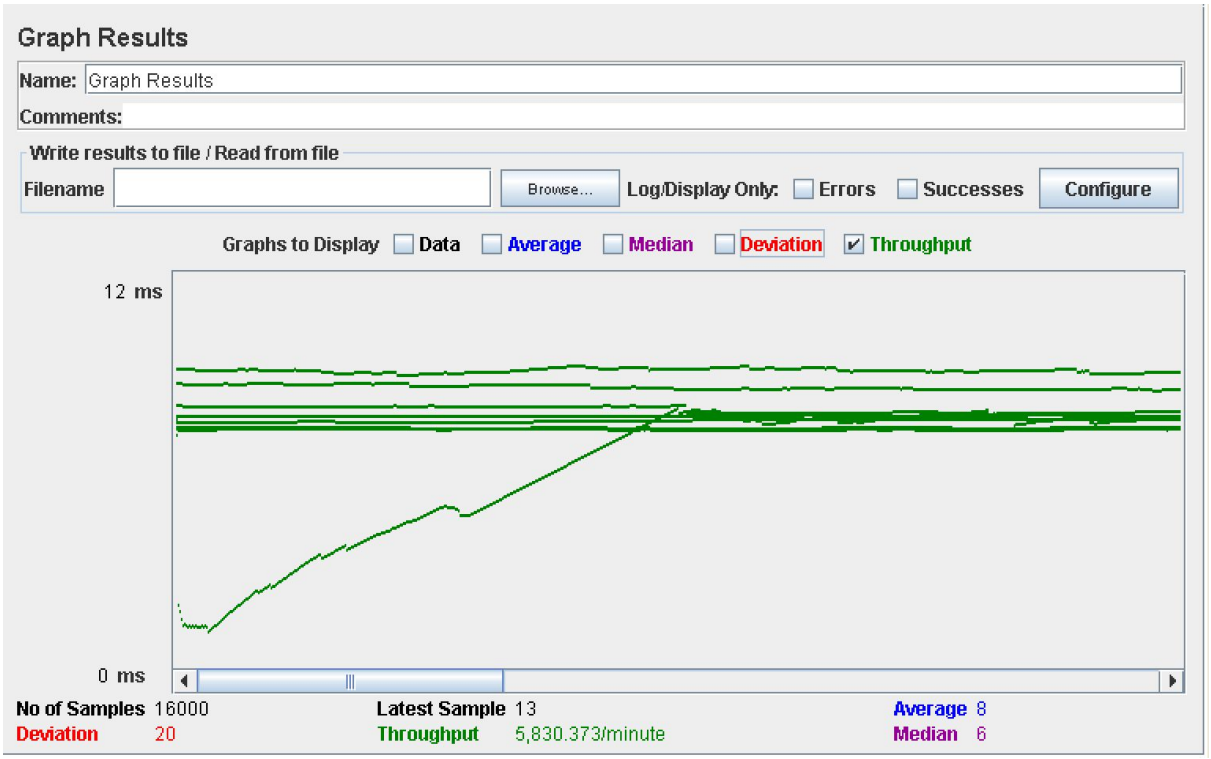
Parámetros:

Número de usuarios: 20

Número de bucles: 100

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	6000	7	5	12	2	268	0.00%	36.4/sec	530.1
/security/login.do	2000	8	6	12	3	368	0.00%	12.3/sec	187.0
/scripts/md5.min.js	2000	4	3	7	1	211	100.00%	12.3/sec	13.6
/j_spring_security_check	2000	4	3	8	1	267	100.00%	12.3/sec	3.4
/admin/chirp/taboo-list.do	2000	13	10	20	5	1176	0.00%	12.3/sec	188.8
/j_spring_security_logout	2000	13	9	18	5	1174	0.00%	12.3/sec	180.7
TOTAL	16000	8	6	13	1	1176	25.00%	97.2/sec	1097.9



Conclusiones:

En esta ocasión utilizaremos una carga total de 20 usuarios concurrentes realizando 100 veces la misma acción. El tiempo de respuesta es bastante bueno, ya que el proceso solo se encargará de visualizar los distintos *chirp* que están registrados con palabras tabú, mostrando solo texto y con ausencia de imágenes. En total, el proceso durará 77 milisegundos en el percentil 90.

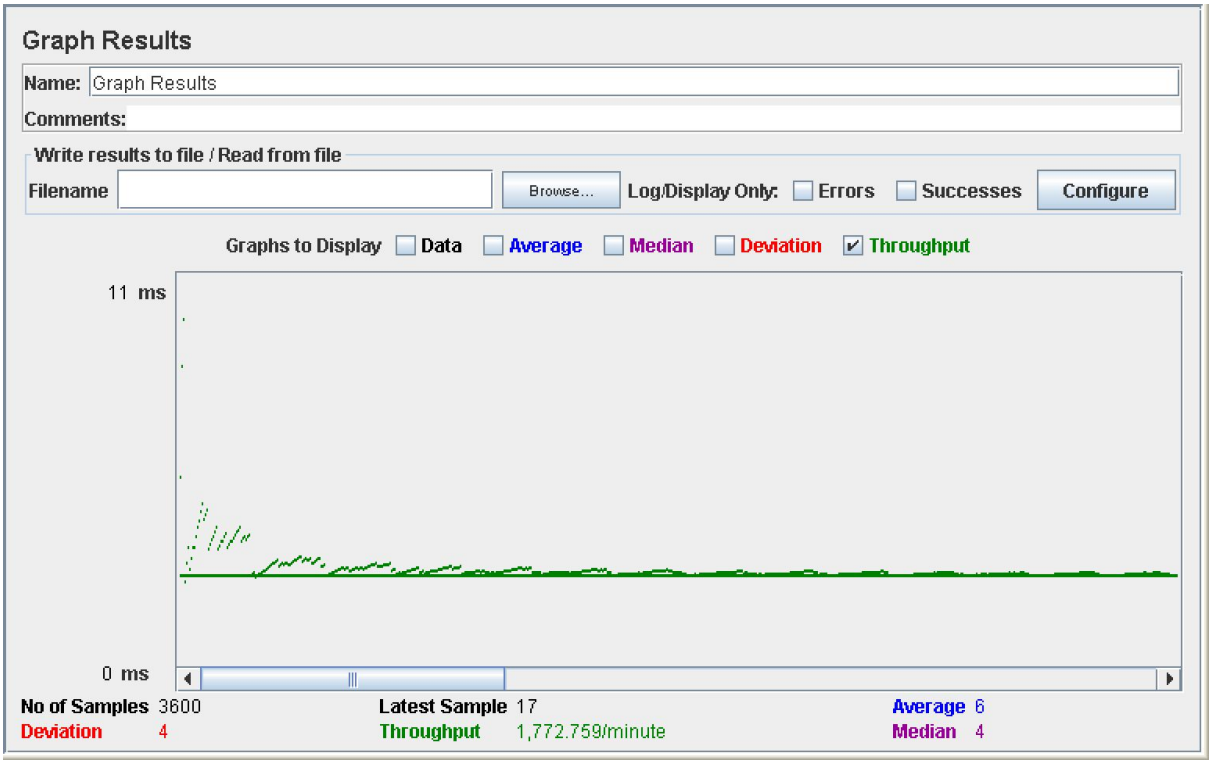
4. Tests de rendimiento del nivel A

4.1. UC18 - Registrarse como customer

Parámetros:

Número de usuarios: 15
Número de bucles: 80

Resultados:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/register/customer.do	2400	6	7	11	1	61	50.00%	19.7/sec	185.4
/	1200	4	4	6	3	25	0.00%	9.9/sec	143.1
TOTAL	3600	6	4	10	1	61	33.33%	29.5/sec	328.4

Conclusiones:

En esta ocasión utilizaremos una carga total de 15 usuarios concurrentes realizando 80 veces la misma acción. Como podemos ver, el tiempo de respuesta es bastante bueno, llegando a necesitar solamente 17 ms para el registro.

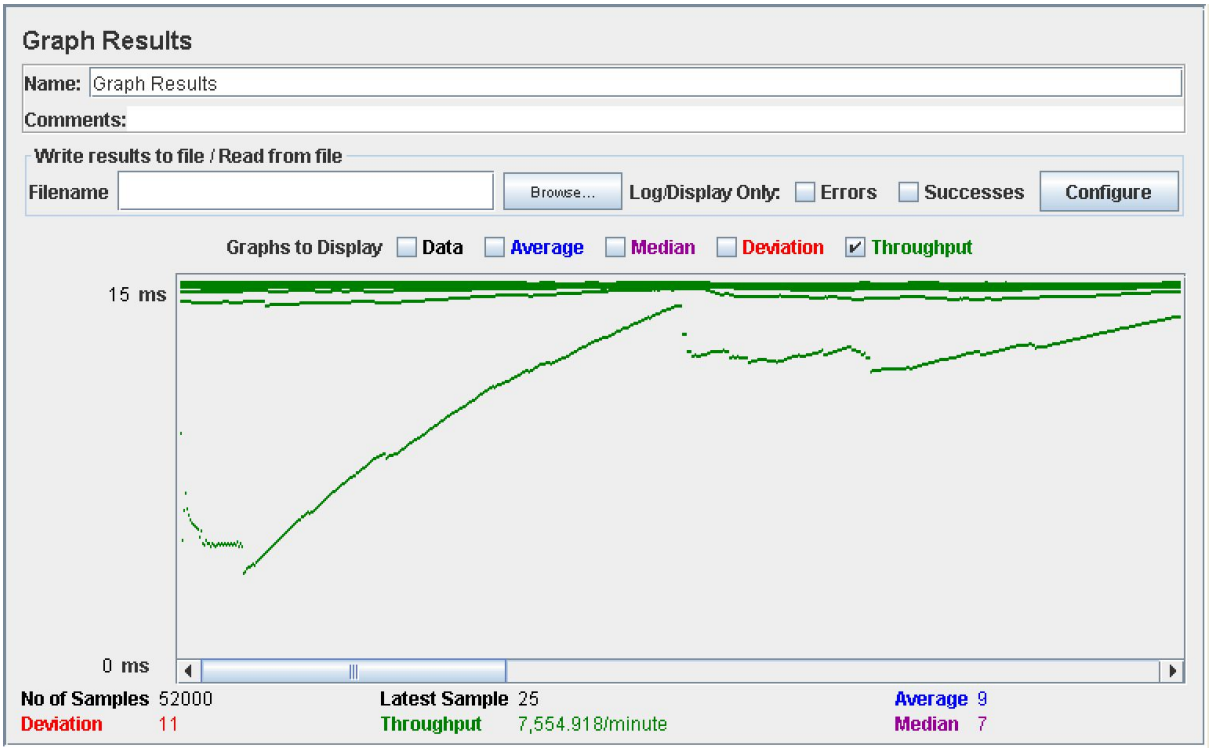
4.2. UC19 - Un customer se suscribe a periódicos privados

Parámetros:

Número de usuarios: 40
Número de bucles: 130

Resultados:

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	10400	6	5	10	2	343	0.00%	25.3/sec	368.1
/security/login.do	5200	7	6	11	3	376	0.00%	12.7/sec	193.3
/scripts/md5.min.js	5200	3	3	6	1	146	100.00%	12.7/sec	14.1
/j_spring_security...	5200	4	3	6	1	150	100.00%	12.7/sec	3.6
/newspaper/list.do	5200	17	14	28	8	437	0.00%	12.7/sec	218.6
/newspaper/displa...	10400	17	14	29	8	404	0.00%	25.3/sec	384.2
/customer/subscri...	5200	11	9	17	4	380	0.00%	12.7/sec	195.2
/customer/subscri...	5200	4	3	7	1	342	100.00%	12.7/sec	3.6
TOTAL	52000	9	7	18	1	437	30.00%	125.9/sec	1373.1



Conclusiones:

En esta ocasión utilizaremos una carga total de 40 usuarios concurrentes realizando 130 veces la misma acción. Podemos comprobar que la ausencia de cargado de imágenes en este proceso hace que la carga de trabajo sea mucho menor. En total, el caso de uso requerirá aproximadamente 114 ms en el percentil 90, un número bastante positivo.

5. Conclusión

Como resultados finales, podemos comprobar que nuestro servidor soportará en cada operación una cantidad aproximada de 30 usuarios de forma concurrente y 100 operaciones al mismo tiempo. Esto no es del todo un mal número, puesto que es importante tener en cuenta la capacidad de nuestro ordenador. En nuestro caso, no hemos podido tener la ventaja de utilizar un ordenador de altas prestaciones, por lo que el rendimiento esperado era similar a lo visto en este documento. Además, hemos realizado las pruebas a través de una máquina virtual en un sistema operativo Windows XP, lo cual nos obliga a reducir las prestaciones de nuestro servidor.

Esto nos hace apreciar la importancia de un servidor de altas prestaciones a la hora de desplegar nuestro sistema de información web si queremos que éste sea capaz de soportar las numerosas solicitudes que un sistema real podría necesitar. El correcto tratamiento del código también nos permitirá obtener una mayor eficacia en éste, pudiendo reducir considerablemente el tiempo de cada una de las operaciones de la aplicación.