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: 09 de mayo de 2018

Contenido

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

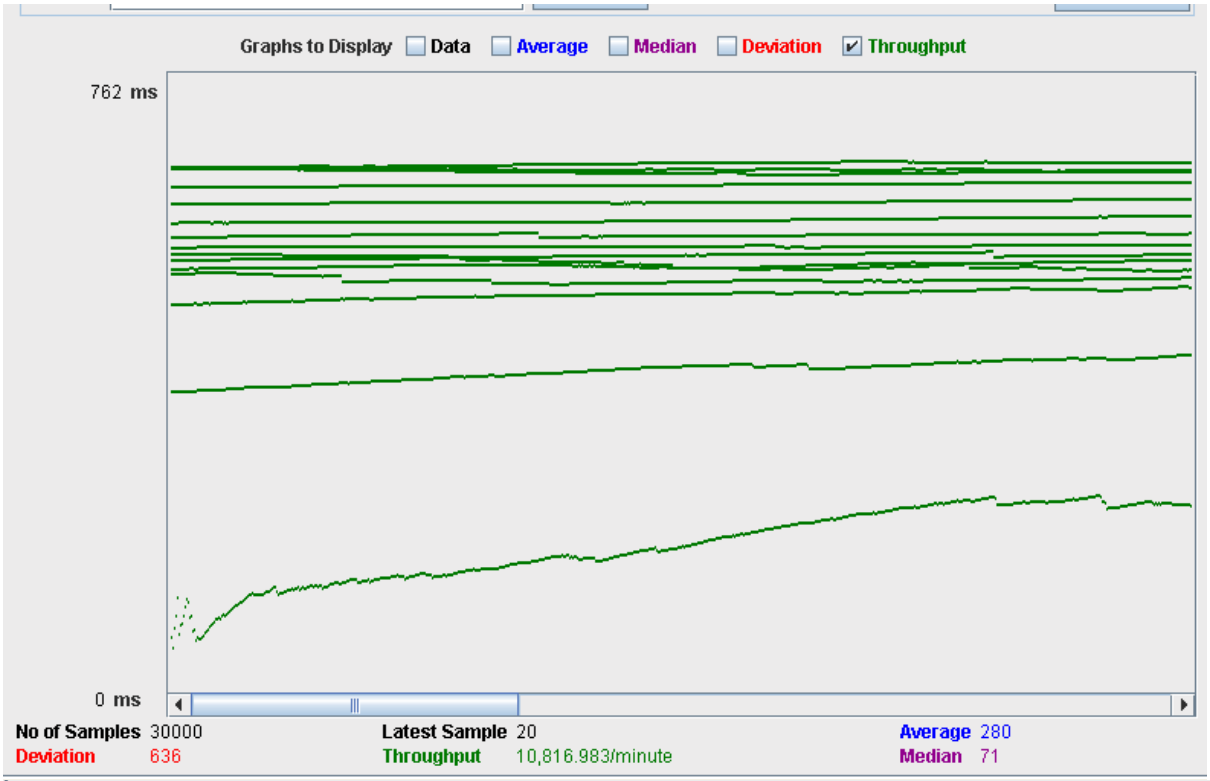
2.1. UC1- Registrarse como agent

Parámetros:

Número de usuarios: 150

Número de bucles: 50

Resultados:



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/	15000	140	54	302	3	6445	0.00%	90.1/sec	1365.1
/register/ag...	15000	420	119	1205	2	12407	50.00%	90.4/sec	895.5
TOTAL	30000	280	71	707	2	12407	25.00%	180.3/sec	2258.4

Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 50 veces la misma acción. Como podemos ver, en el percentil 90 se tardará poco más de 1 segundo en aplicar el registro. En total, la operación tardará 1507 milisegundos en procesar un registro completo de Agente.

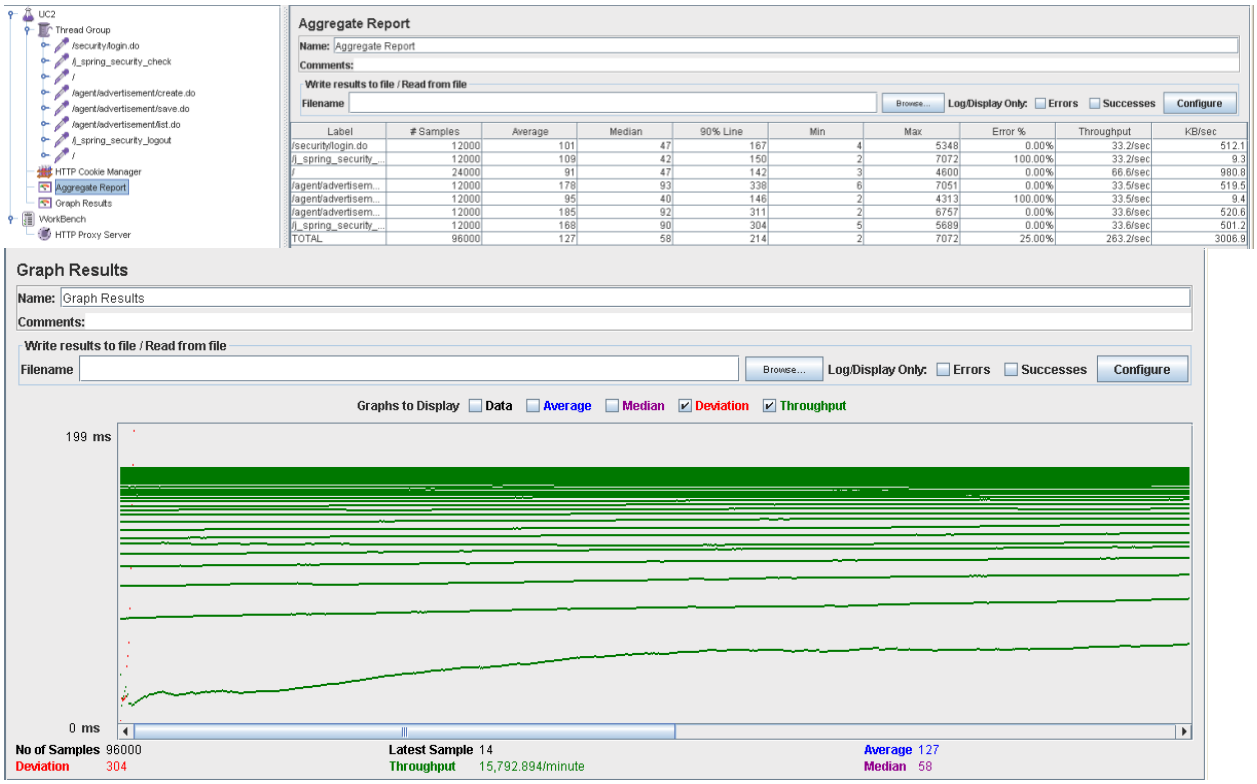
2.2. UC2- Un agente crea un anuncio para un periódico

Parámetros:

Número de usuarios: 160

Número de bucles: 75

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 160 usuarios concurrentes realizando 75 veces la misma acción. En esta ocasión se realizarán tres operaciones claves: la primera, el controlador create.do; el cual carga completamente el formulario de creación del anuncio durando 338 ms en el percentil 90. La segunda es el controlador save.do que se encargará de guardar dicho objeto en la base de datos. Al no tener que cargar ninguna vista, ésta será la operación que menos tiempo lleve de los tres (un total de 146 ms). Por último, al ejecutar dicha operación somos redirigidos al listado de anuncios, llevándonos un total de 311 ms. En total, el proceso de creación de un anuncio para un periódico conllevará segundo y medio (concretamente 1558 milisegundos).

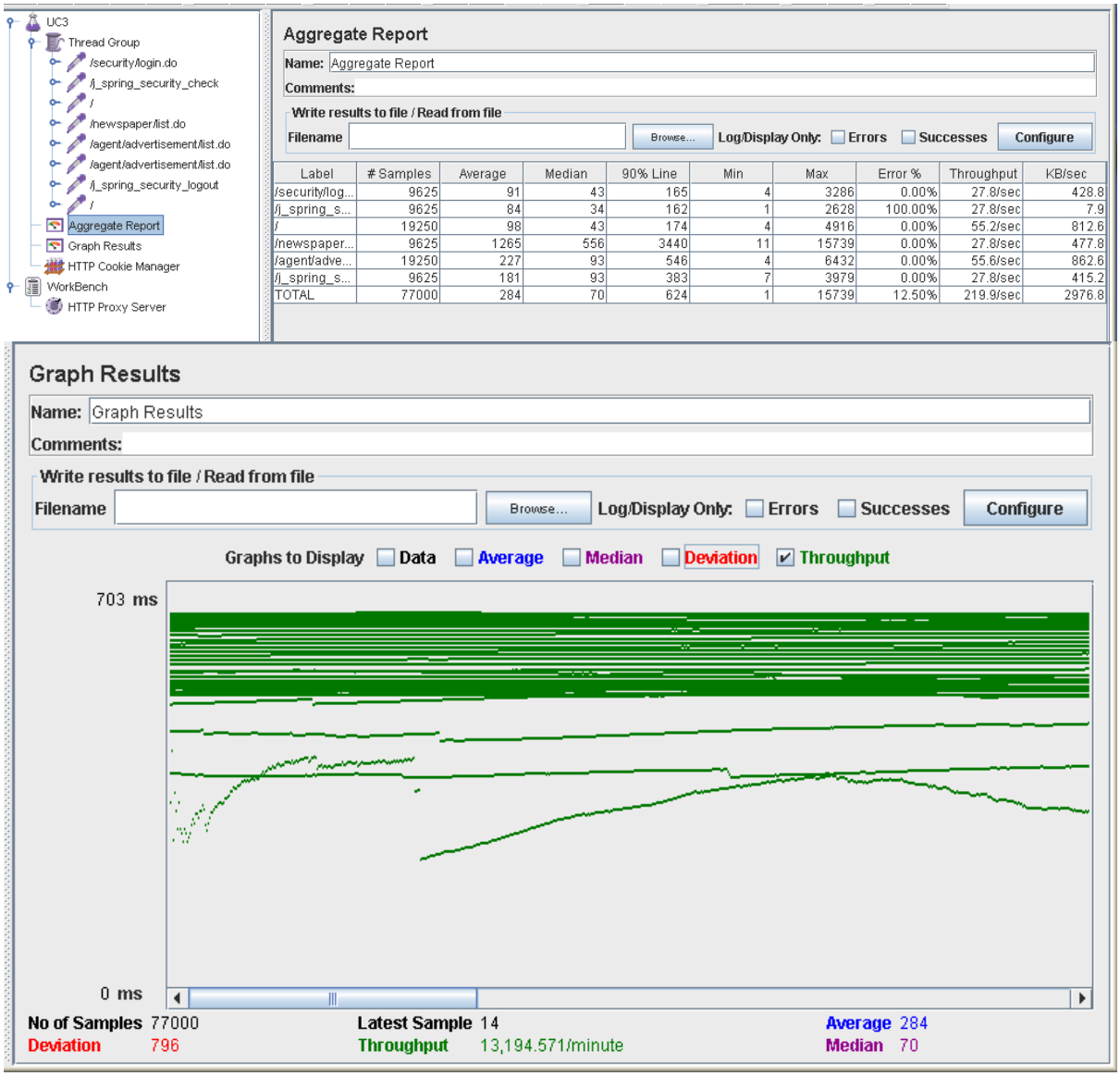
2.3. UC3- Un agente lista los periódicos en los que tiene y no tiene publicidad.

Parámetros:

Número de usuarios: 175

Número de bucles: 55

Resultados:



Conclusiones:

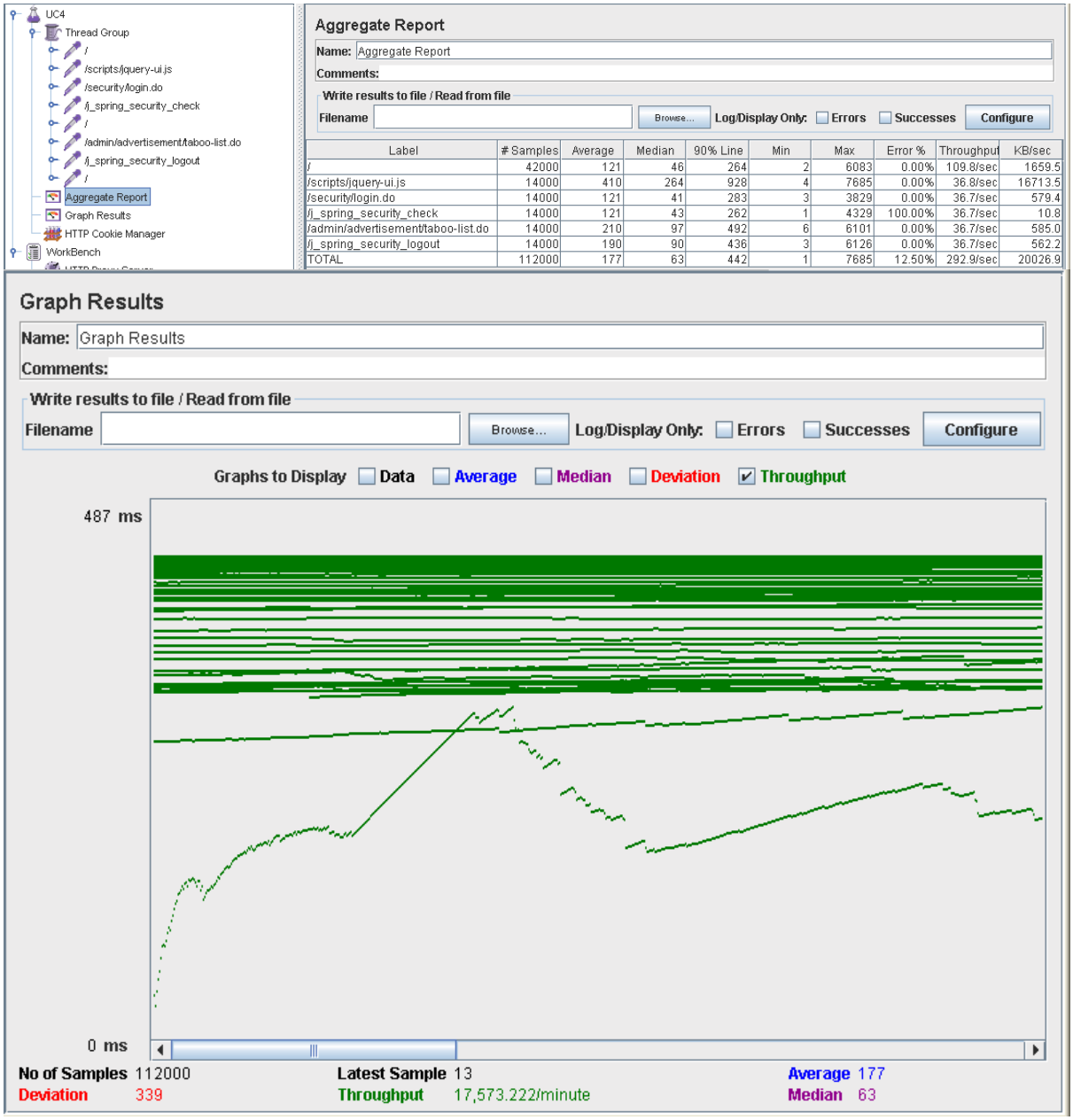
En esta ocasión utilizaremos una carga total de 175 usuarios concurrentes realizando 55 veces la misma acción. En este caso podemos ver dos operaciones importantes. La primera es el acceso al listado de periódicos, el cual se lleva la mayor parte del tiempo por el hecho de tener que cargar todos los objetos (casi tres segundos y medio). La segunda es la operación que buscamos, la cual actúa como un filtro que buscará los periódicos donde tengamos o no tengamos publicidad. Al aprovechar recursos de la anterior operación, ésta tardará menos, concretamente 546 ms. En total, la operación llevará casi 5 segundos (4870 ms).

2.4. UC4- Un administrador lista los anuncios con palabras prohibidas en su título

Parámetros:

Número de usuarios: 175
Número de bucles: 80

Resultados:



Conclusiones:

Como se puede ver se han usado 175 usuarios concurrentes haciendo 80 veces la misma acción. Se trata de una simple operación de listado que tan solo alcanza los 492 ms en el percentil 90. En total, la operación llevará poco más de dos segundos y medio (2665 ms).

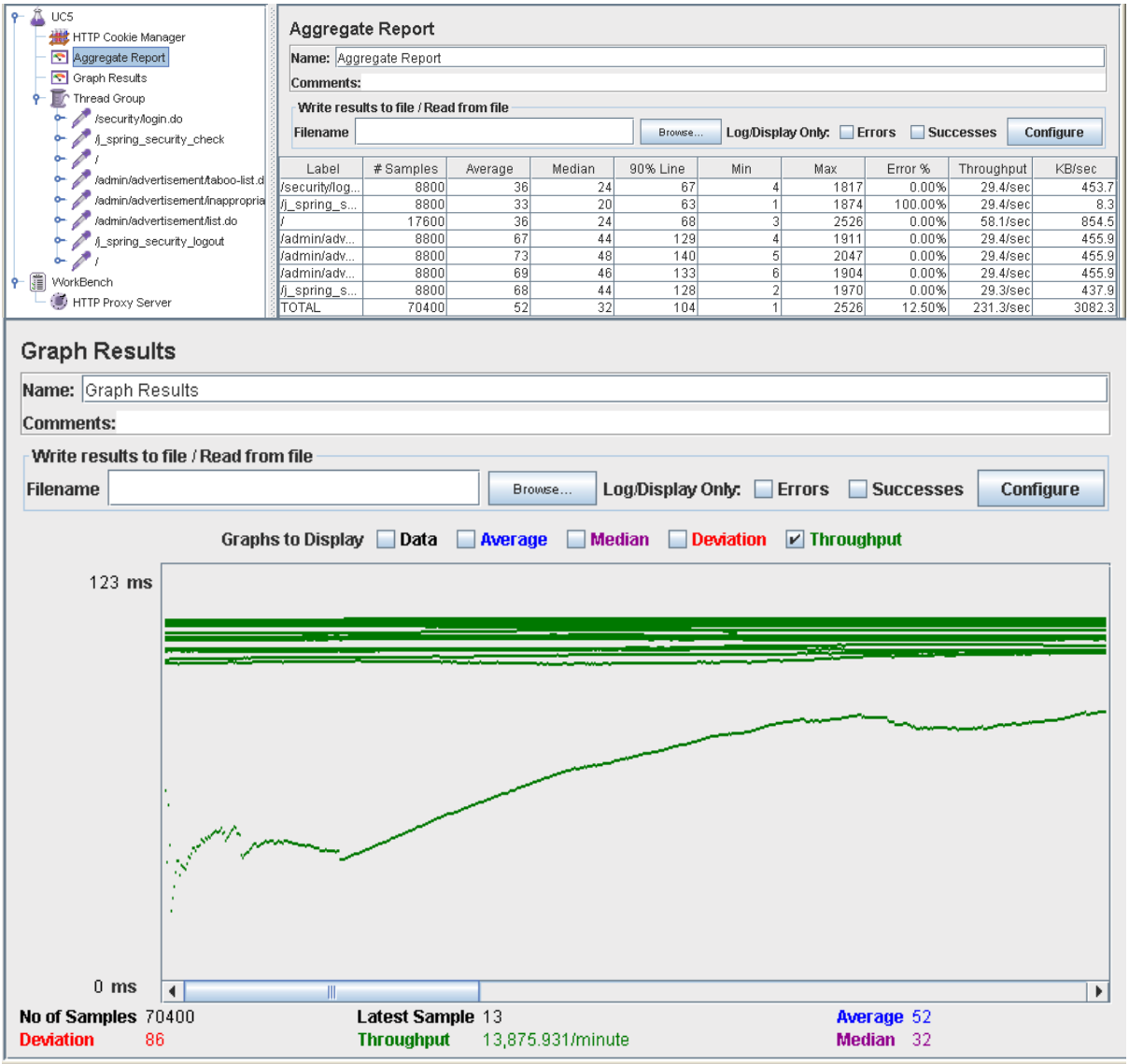
2.5. UC5-Un administrador puede marcar como inapropiado un anuncio

Parámetros:

Número de usuarios: 165

Número de bucles: 55

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 165 usuarios concurrentes realizando 55 veces la misma acción. Este proceso se basa en dos operaciones fundamentales. La primera (ya como admin), carga la lista total de anuncios (133 ms en el percentil 90). La segunda, marcará como inapropiado uno de estos anuncios, redirigiendo de nuevo a la lista de anuncios (140 ms). En total, la operación llevará unos 728 milisegundos.

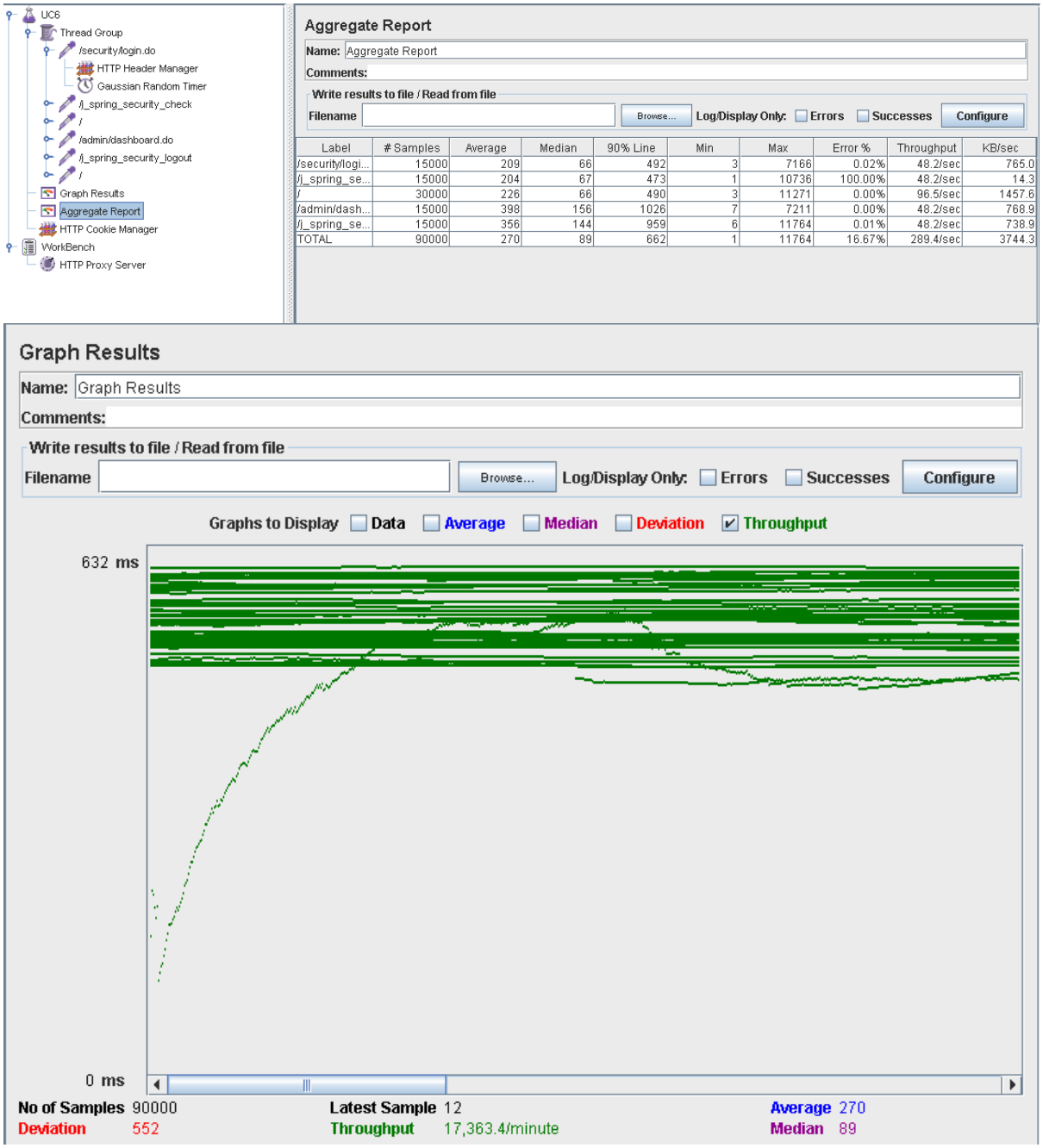
2.6. UC6- Un administrador puede visualizar su dashboard

Parámetros:

Número de usuarios: 200

Número de bucles: 80

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 200 usuarios concurrentes realizando 80 veces la misma acción. A pesar de ser una simple operación de listado sin imágenes, la vista presenta numerosas listas y, por tanto, numerosos objetos que cargar; de ahí a que le lleve un segundo completo cargar la *dashboard*. Finalmente, el tiempo total de carga será de 3440 ms.

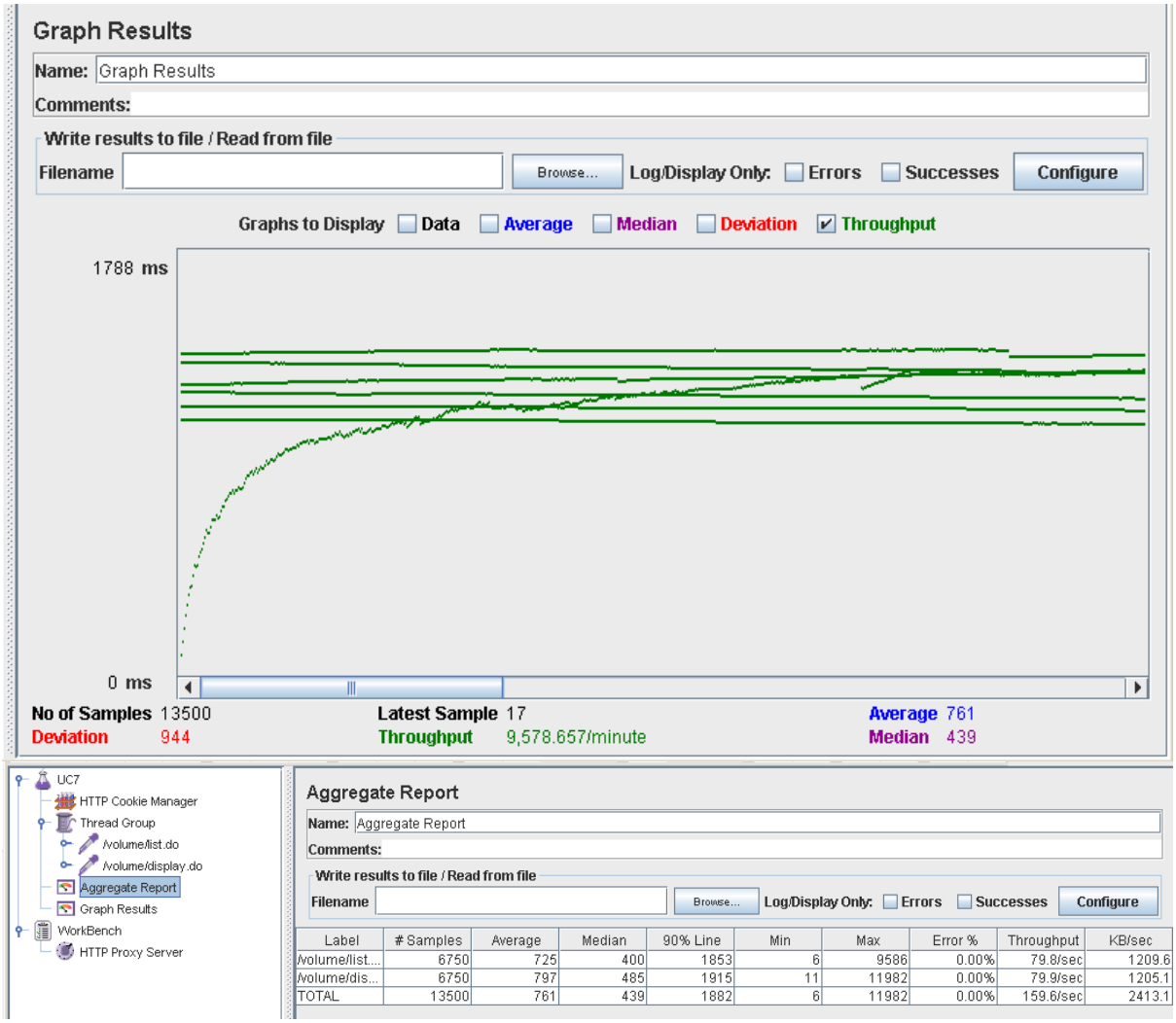
3. Tests de rendimiento del nivel B

3.1. UC7- Listar los *volumes* y navegar a su periódicos

Parámetros:

Número de usuarios: 150
Número de bucles: 45

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 45 veces la misma acción. Se basa en dos operaciones de listado básicas: la primera, que nos llevará hasta la lista de *volumes* en un total de 1853 ms en el percentil 90. La segunda será la vista de *display* de uno de los elementos anteriores, donde podremos ver los datos con

detalle junto a sus periódicos (en 1915 ms). Finalmente, la operación completa llevará unos 3768 ms (casi cuatro segundos en total).

3.2. UC8- Un Customer puede suscribirse a un volume

Parámetros:

Número de usuarios: 140
Número de bucles: 45

Resultados:

Conclusiones:

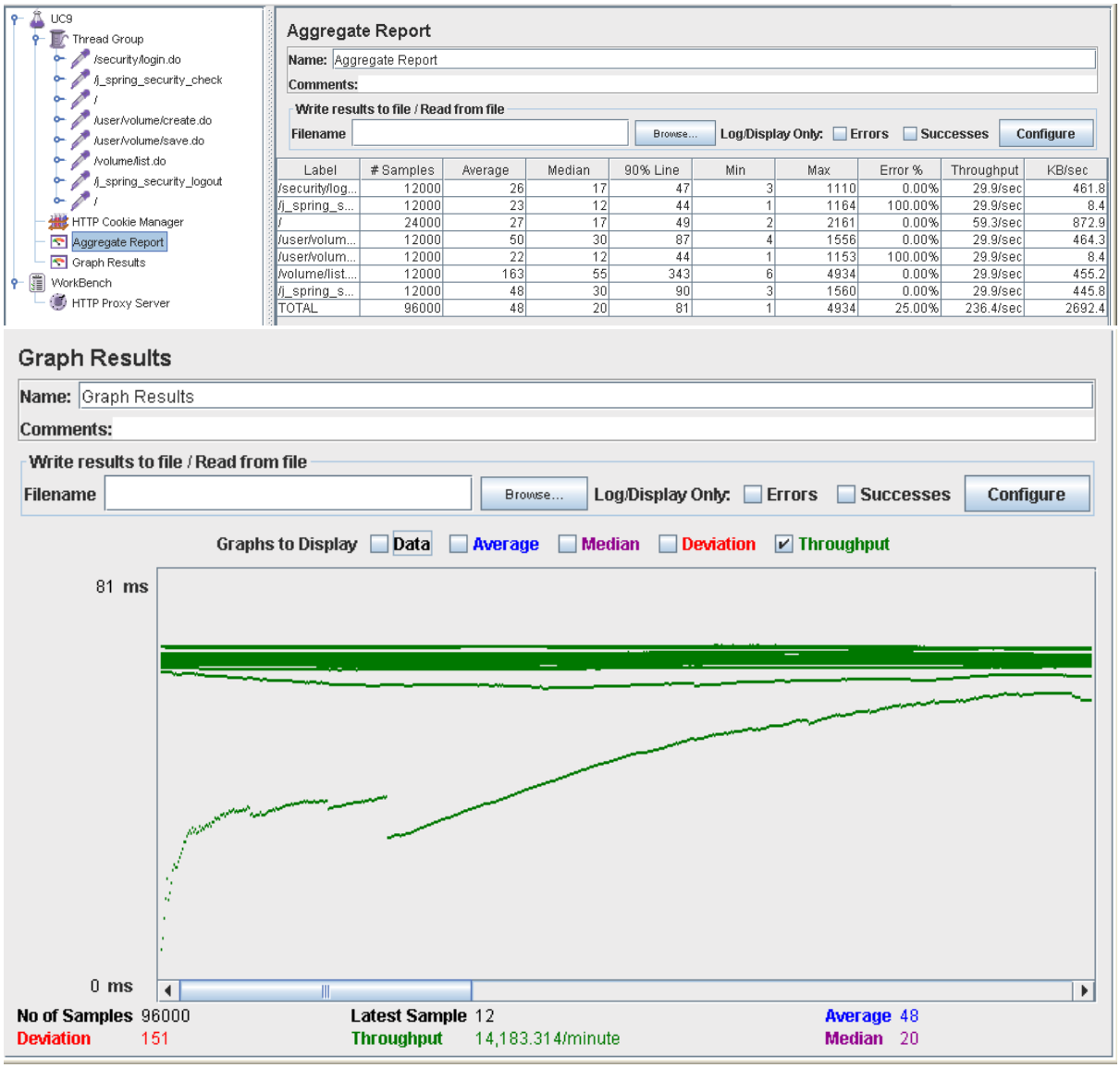
En esta ocasión utilizaremos una carga total de 140 usuarios concurrentes realizando 45 veces la misma acción. Como podremos observar, la suscripción se realizará a través de cuatro pasos: listado de *volume* (2929 ms en el percentil 90), vista en detalle de uno de éstos (3176 ms), carga del formulario de suscripción (124 ms) y guardado de éste (51 ms), siendo notoriamente los dos últimos los más rápidos en ser realizados.En total, el proceso llevará seis segundos y medio en realizar todas las operaciones.

3.3. UC9- Un Usuario puede crear un Volume

Parámetros:

Número de usuarios: 160
Número de bucles: 75

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 160 usuarios concurrentes realizando 75 veces la misma acción. En esta ocasión se realizará el clásico proceso de creado, guardado y redirección al listado de *volumes*, durando cada uno 87, 44 y 343 ms en el percentil 90 (nótese que es la operación de listado la que más tiempo llevaría). Unido a los procesos necesarios de login, la acción llevaría en total unos 704 ms, número bastante positivo ya que ni siquiera superaría un segundo.

4. Tests de rendimiento del nivel A

4.1. UC10 - Cualquier actor registrado puede intercambiar mensajes con otro actor

Parámetros:

Número de usuarios: 160
Número de bucles: 55

Resultados:

Aggregate Report

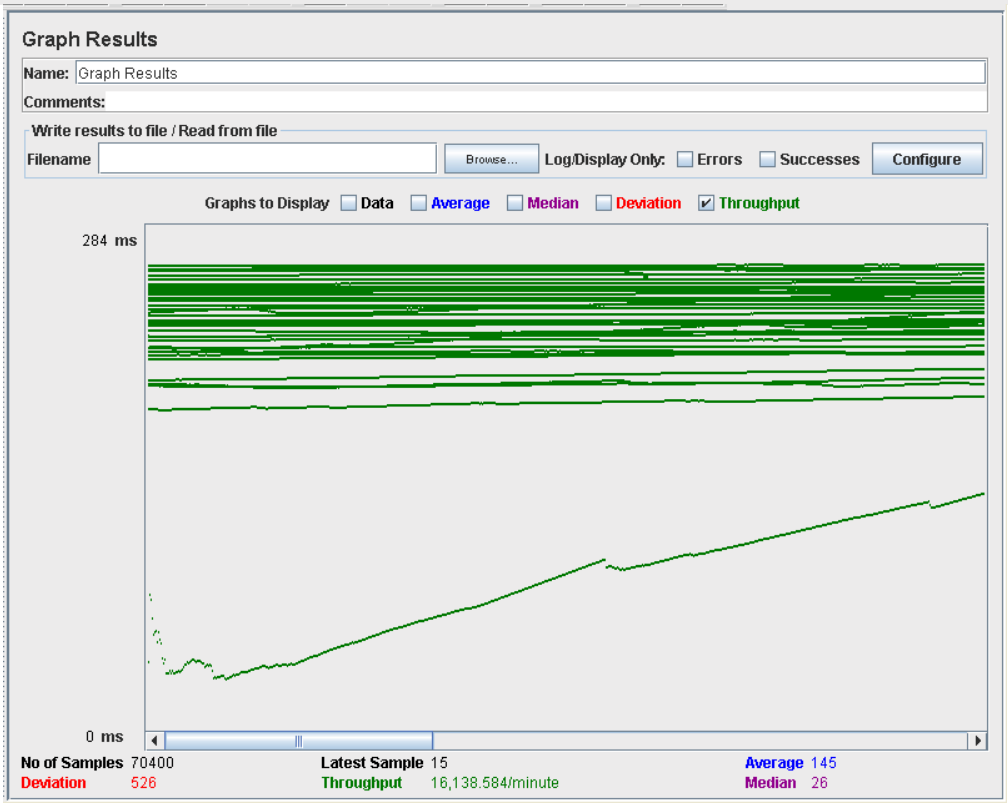
Name: Aggregate Report

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
/security/login.do	8800	35	19	60	3	6109	0.00%	34.1/sec	540.3
/j_spring_security_check	8800	39	16	75	1	3800	100.00%	34.1/sec	10.0
/	17600	37	19	64	3	5441	0.00%	67.7/sec	1023.1
/profile/message/list.do	17600	456	116	1174	4	11760	0.00%	67.7/sec	1034.1
/profile/message/send.do	8800	38	16	73	1	3283	100.00%	34.0/sec	10.0
/j_spring_security_logout	8800	65	35	118	2	6175	0.00%	34.0/sec	520.8
TOTAL	70400	145	26	244	1	11760	25.00%	269.0/sec	3110.5



Conclusiones:

En esta ocasión utilizaremos una carga total de 160 usuarios concurrentes realizando 55 veces la misma acción. Gracias a las ventajas que nos proporciona AJAX, este proceso solo requerirá dos operaciones. La primera, el listado de mensajes, necesario para poder proceder a enviar el mensaje (1174 ms en el percentil 90). En esta vista ya podremos escribir nuestro mensaje y enviarlo, tardando 73 ms en ello. Nótese que al no tener que cargar de vista, la operación de envío se realiza bastante rápido. En total, el proceso llevará 1564 milisegundos.

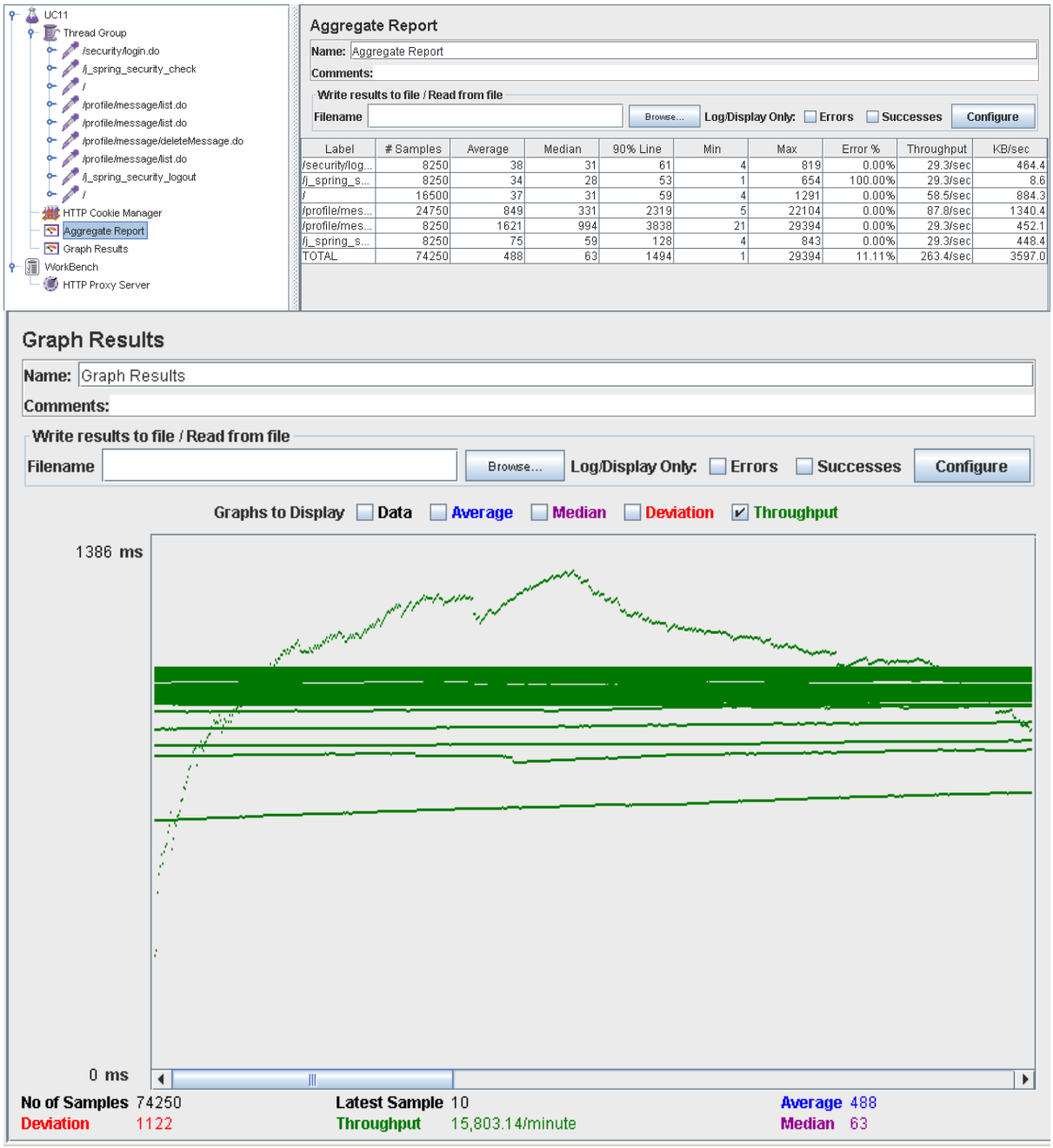
4.2. UC11 - Los actores pueden eliminar mensajes

Parámetros:

Número de usuarios: 150

Número de bucles: 55

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 55 veces la misma acción. En esta ocasión, el usuario accede a los mensajes con el primer list.do de mensajes, accede a una carpeta distinta con el segundo listado, borra uno de los mensajes con el delete.do y finalmente es redirigido de nuevo al mismo lugar con el último

listado. En total, podemos ver que la eliminación lleva poco más de dos segundos, mientras que los tres listados ocupan 3838 milisegundos en el percentil 90. En total, este proceso ha durado 6458 milisegundos.

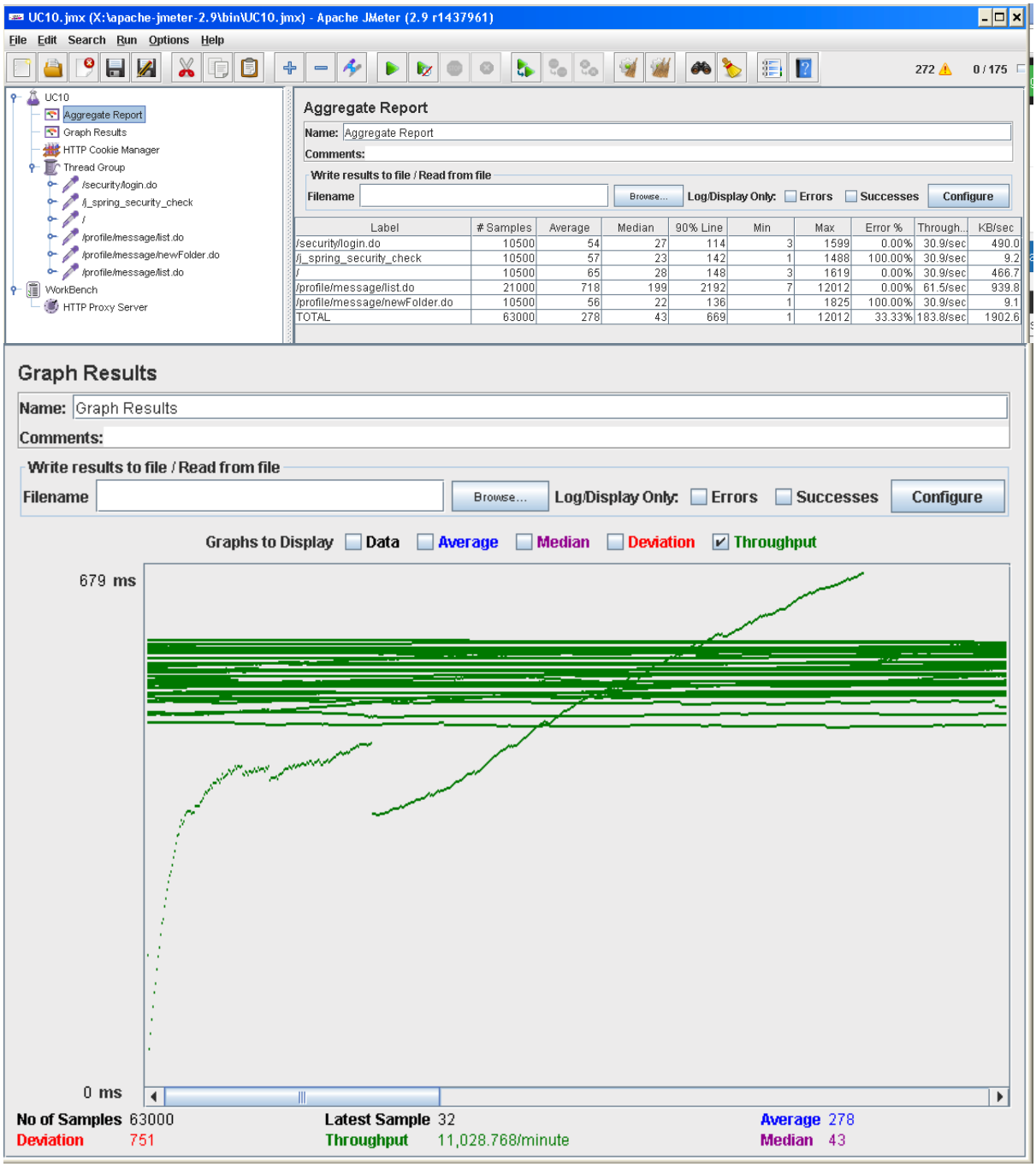
4.3. UC12 - Los actores pueden crear carpetas

Parámetros:

Número de usuarios: 175

Número de bucles: 60

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 175 usuarios concurrentes realizando 60 veces la misma acción. De nuevo, AJAX nos permite reducir a dos simples operaciones este proceso: el primero de listado para cargar el sistema de mensajería, durando 2192 ms en el percentil 90 y por último el operador de guardado de la nueva carpeta, el cual al no tener que cargar una vista tan solo durará 136 ms. En total, el proceso llevará unos 2732 ms.

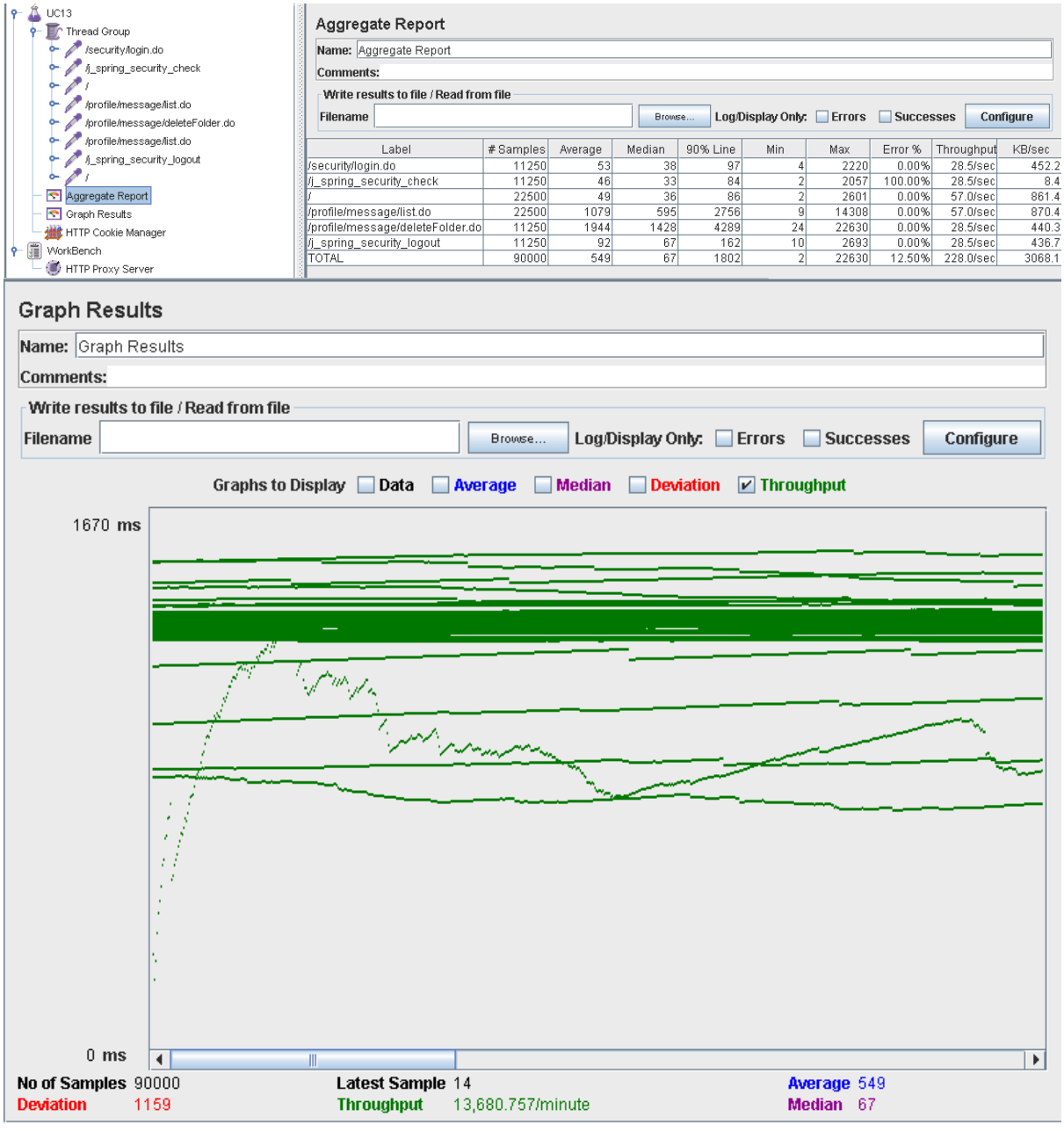
4.4. UC13 - Los actores eliminar o editar carpetas

Parámetros:

Número de usuarios: 150

Número de bucles: 75

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 75 veces la misma acción. Con la misma metodología que en las operaciones anteriores, el listado llevará unos 2756 ms en ser realizado mientras que el borrado se realizaría en 4289 ms. El proceso duraría un total de 7474.

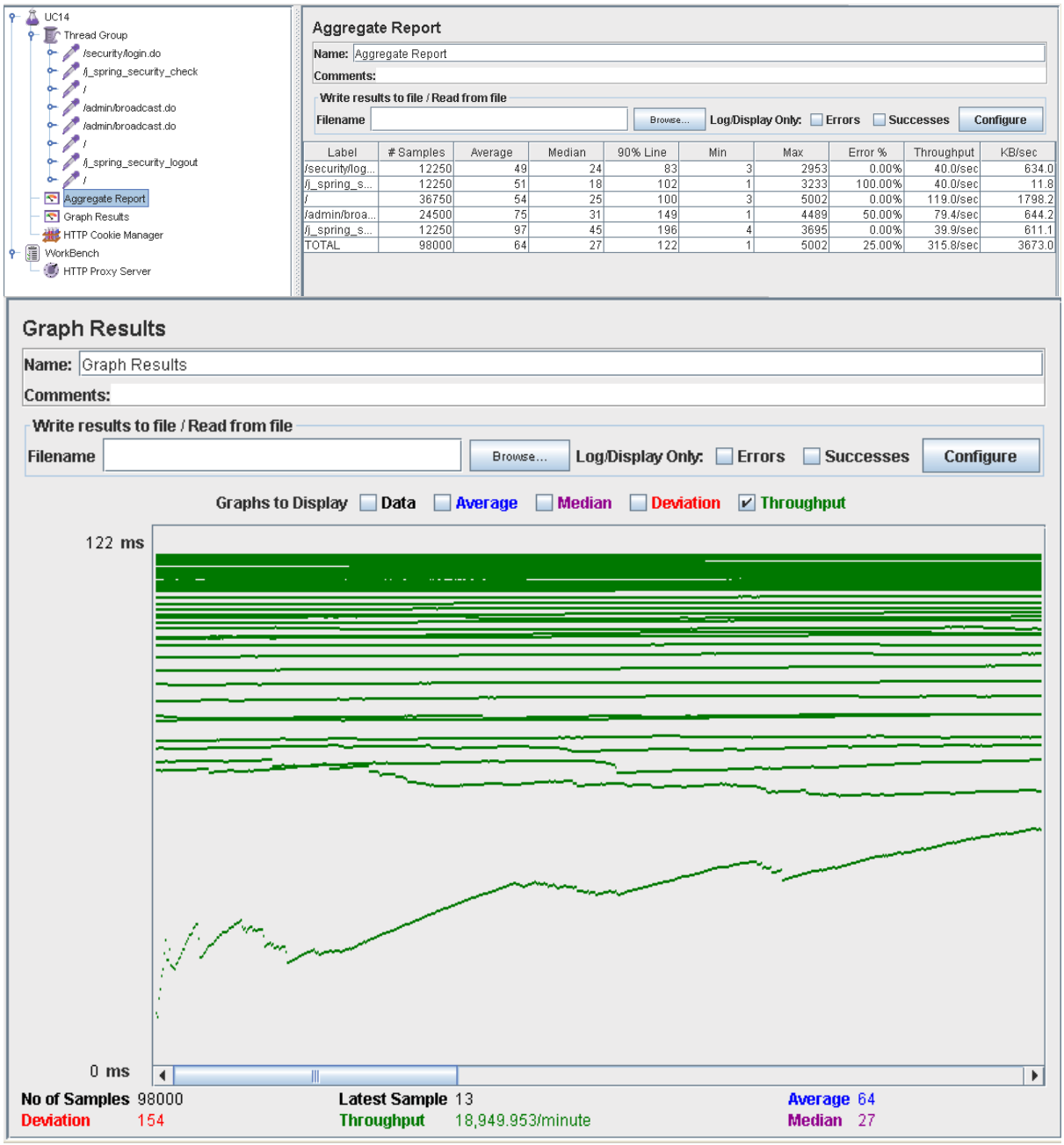
4.5. UC14 - El administrador puede enviar un mensaje a los demás usuarios del sistema.

Parámetros:

Número de usuarios: 175

Número de bucles: 70

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 175 usuarios concurrentes realizando 70 veces la misma acción. Se trata de un formulario bastante sencillo (149 ms en ser cargado) que nos redirigirá a la pantalla principal, durando 100 ms). En total, el proceso llevará unos 630 ms, cantidad inferior a un segundo.

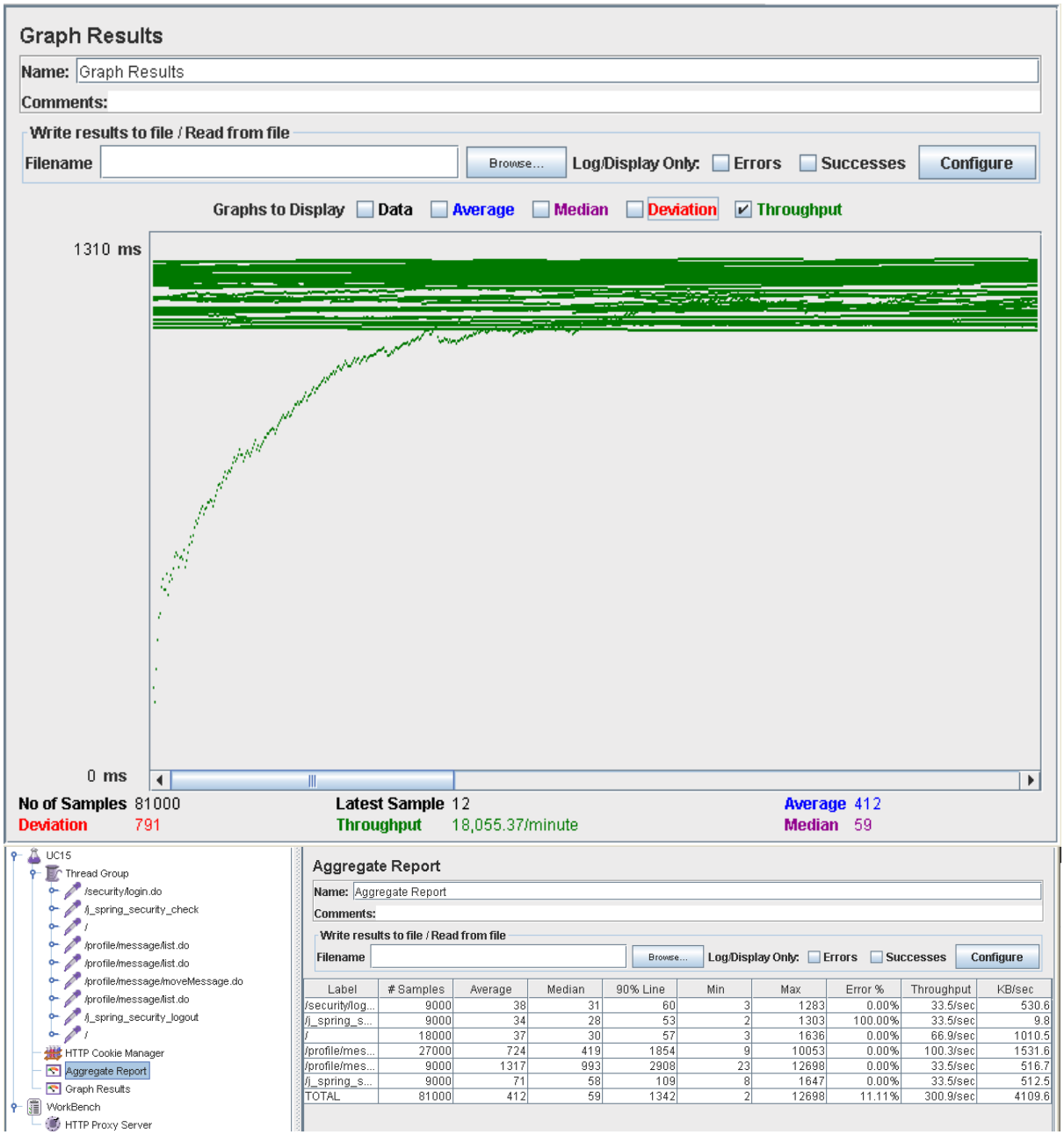
4.6. UC15 - Los mensajes pueden moverse a carpetas creadas por un usuario

Parámetros:

Número de usuarios: 150

Número de bucles: 60

Resultados:



Conclusiones:

En esta ocasión utilizaremos una carga total de 150 usuarios concurrentes realizando 60 veces la misma acción. Al igual que en el resto operaciones del mismo tipo, el proceso se resume en 4 pasos donde tres de ellos serán movimientos a través de las carpetas de mensajería, mientras que el último movería el mensaje de una carpeta a otra (ocupando

1854 ms en el percentil 90). Gracias a AJAX, el proceso se ve bastante agilizado, durando en total 5 segundos.

5. Conclusión

Como resultados finales, podemos comprobar que nuestro servidor soportará en cada operación una cantidad aproximada de 150 usuarios de forma concurrente y 50 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.