Ingeniería civil en Informática
Facultad de Ingeniería

Tarea 02: Desarrollo de App Web Proyecto: Cálculo y Optimización de rendimiento usando Jmeter

Integrantes	**Felipe Herrera Moraga**
	Rosa Velásquez Rojas

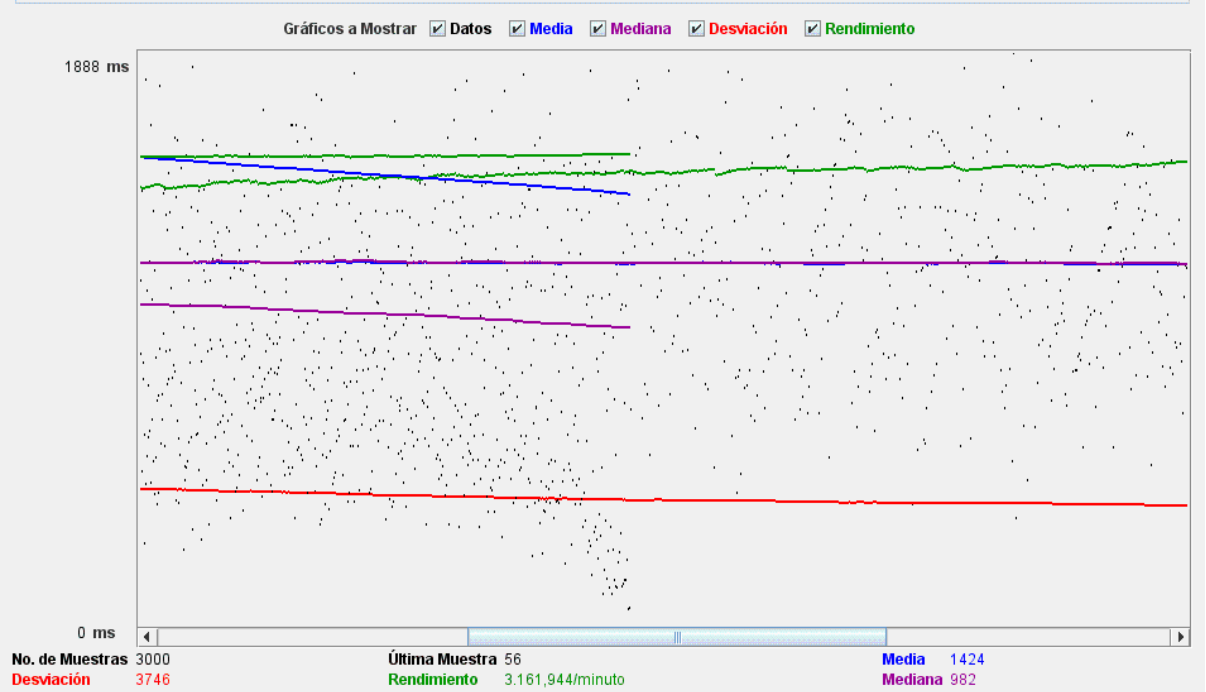
29 Mayo 2015

Contenido

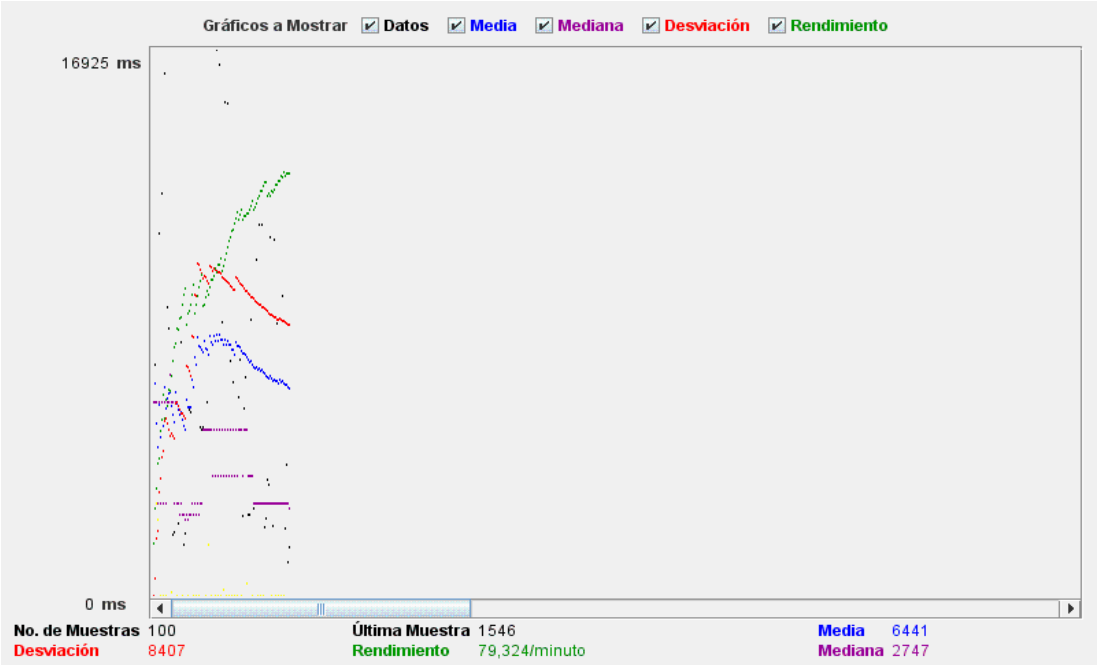
Descripción de la Etapa Inicial

Para la realización de las pruebas, se creó un proyecto realizado en el framework Codeigniter 2.2 y Bootstrap, al cual se le hicieron pruebas de carga con 100 hilos por segundo. Para mostrar el funcionamiento del proyecto a continuación se muestran las imágenes obtenidas desde Jmeter.

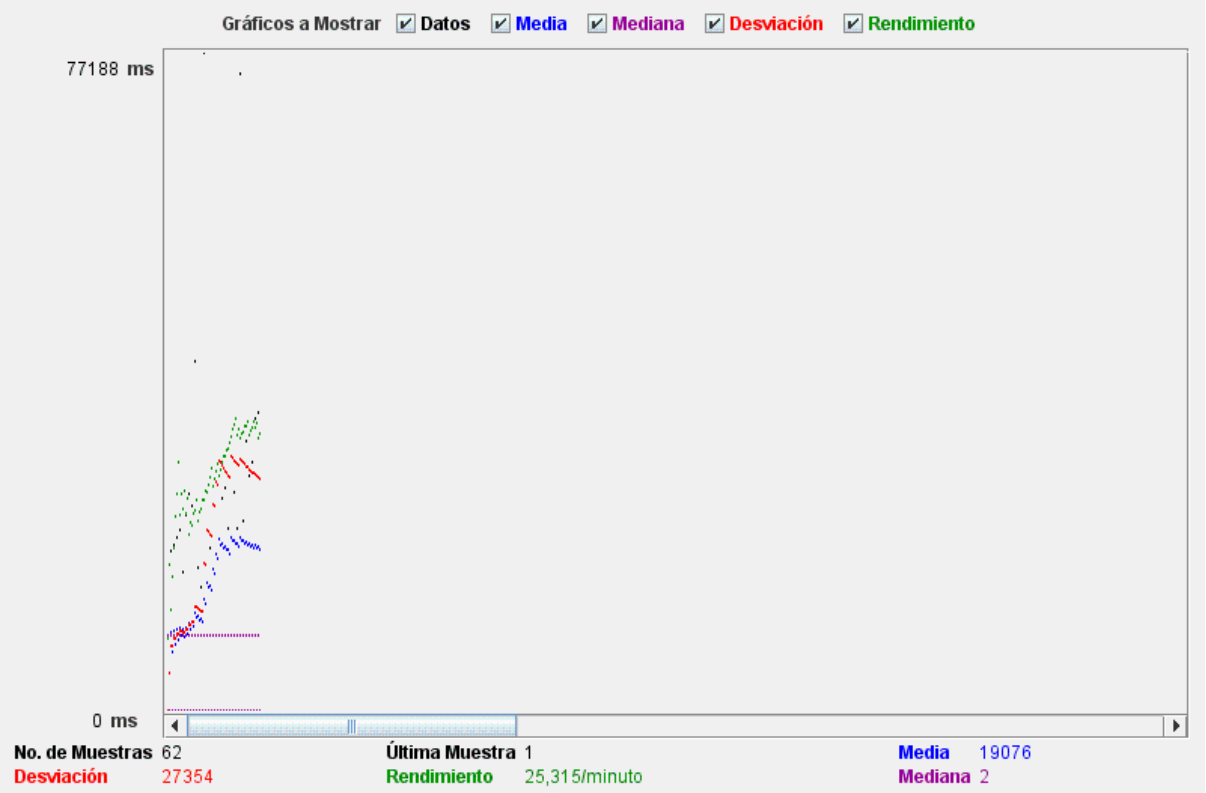
Consulta simple:



Consulta media:



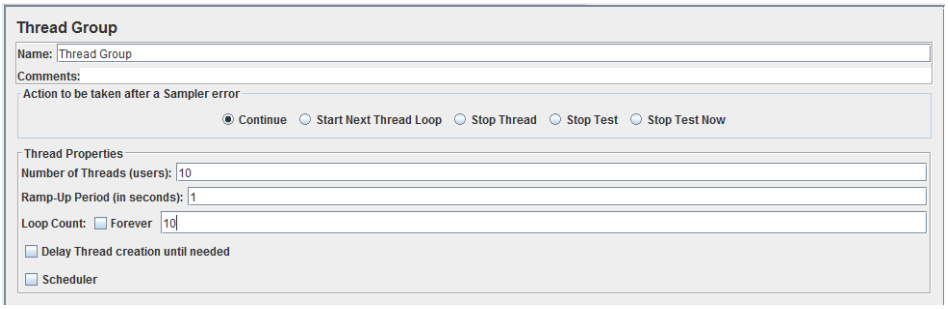
Consulta compleja:



Iteraciones

Debido a los altos márgenes con respecto al porcentaje de error arrojado en cada una de las consultas por Apache Jmeter es que se piensa en optimizar este proyecto. A continuación, se detallan los procedimientos y la justificación de cada una de las iteraciones realizadas, debe considerarse que para la realización de las pruebas hubieron 2 etapas de configuración, las cuales se describen a continuación:

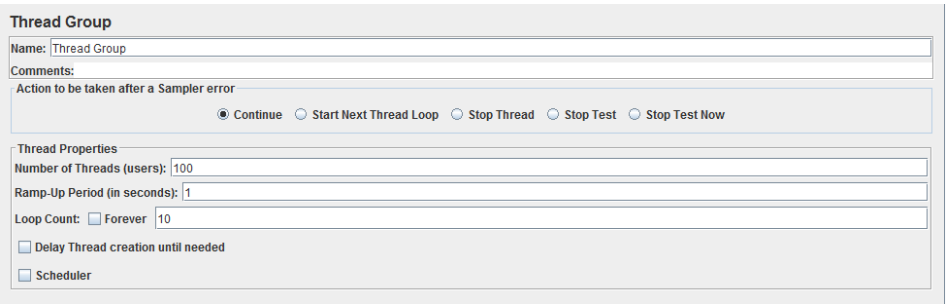
Esta configuración, si bien no genera demasiadas muestras, está basada en el consumo de recursos que proveía el servidor, y permite que el cierre de los hilos sea de forma correcta.



Para esta prueba se envió un usuario cada 100 milisegundos 100 veces en total.

La siguiente configuración se realizó para las consultas medias y complejas desde la iteración 2 en adelante, esto se realizó debido a que el tiempo que se demoraba en

realizar la prueba era demasiado corto, y no mostraba un consumo de recursos significativos en el servidor.



Para esta prueba se envió un usuario cada 1000 milisegundos 100 veces en total.

1era Iteración (Sin bootstrap)

En esta iteración, se decidió eliminar bootstrap como primera optimización, para que el envío de datos desde el servidor disminuyera y repartiera de mejor forma el ancho de banda entre los usuarios. A continuación se presentan los gráficos de las 3 consultas realizadas:simple, media y compleja, respectivamente.

Gráfico de resultados de la consulta Simple:

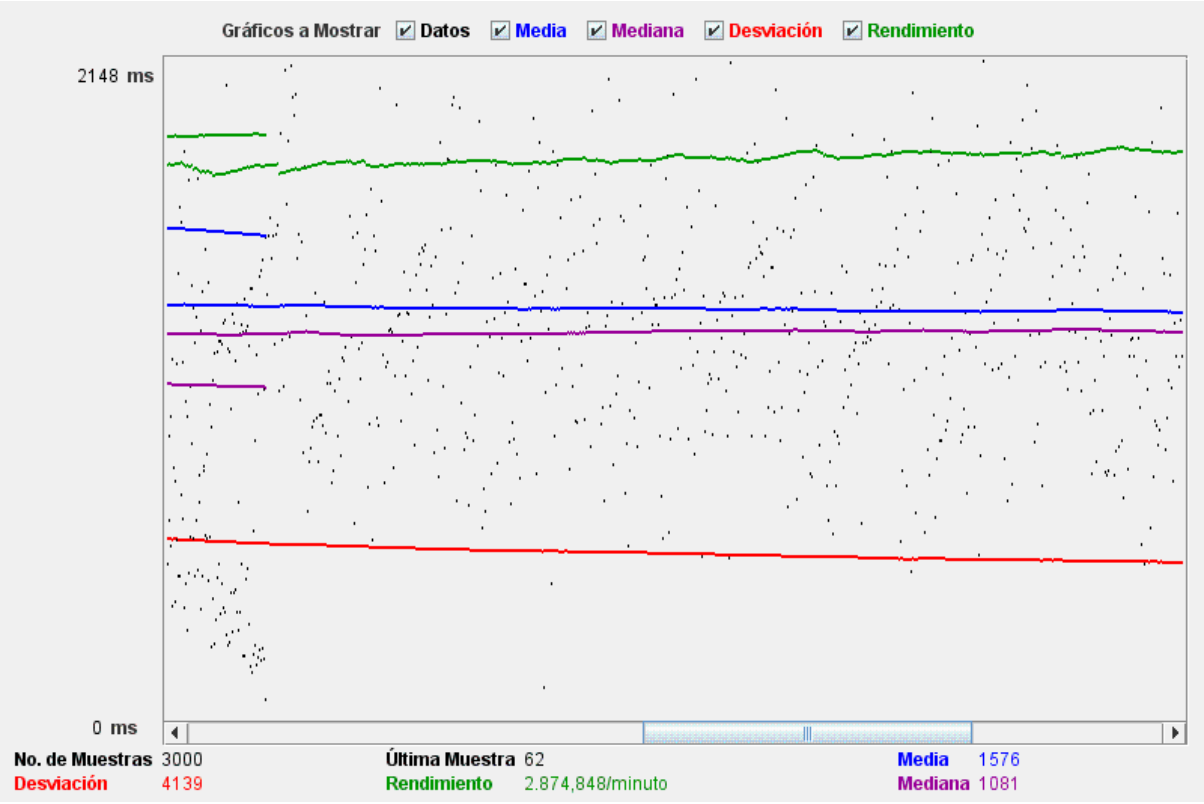


Gráfico de resultados de la consulta media:

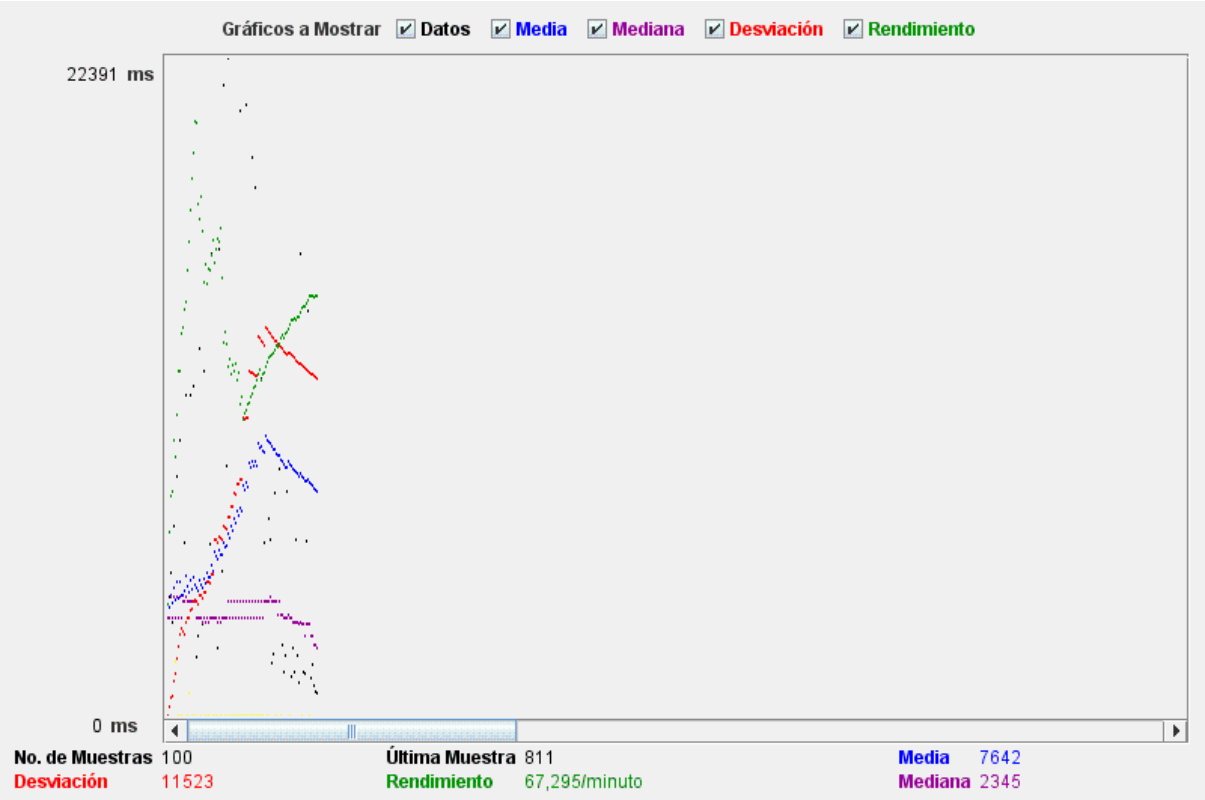
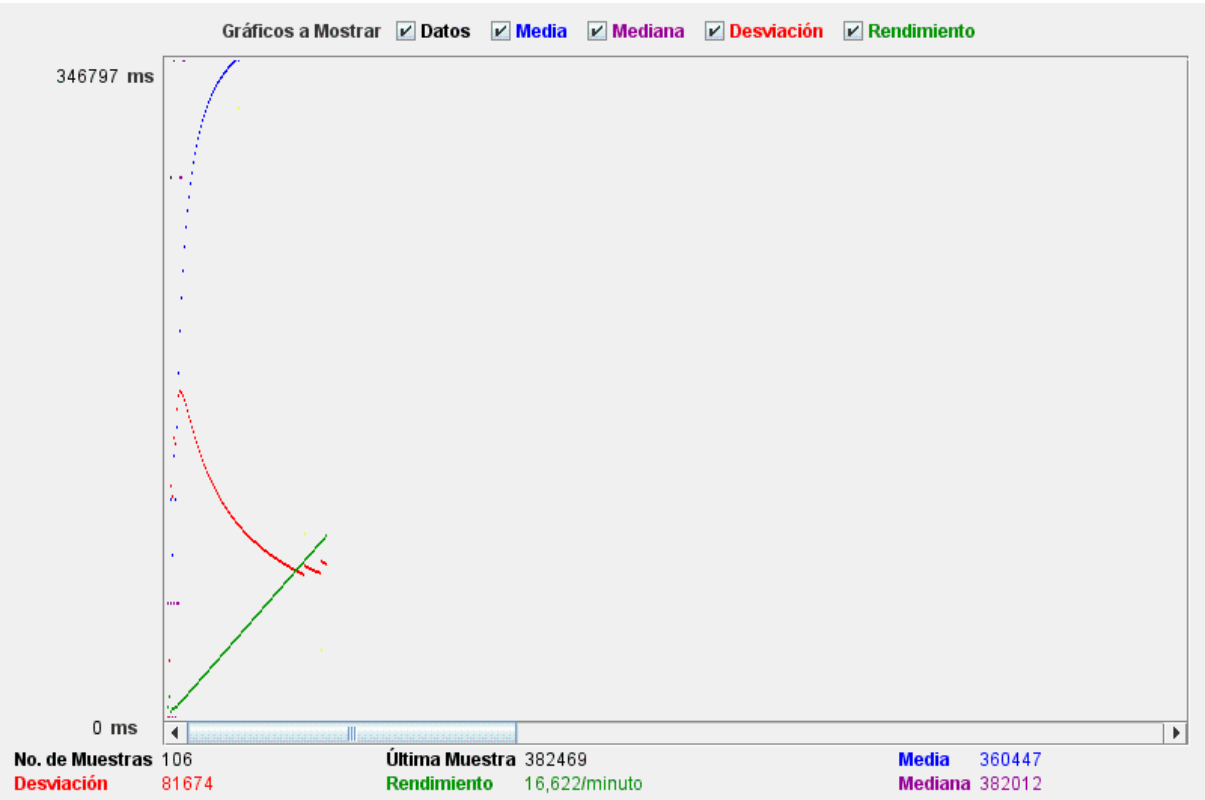


Gráfico de resultados de la consulta compleja:



2da Iteración (Paginación)

Debido a que la consulta media y la consulta compleja arrojaban demasiados datos en una vista, decidimos paginarlas, mostrando 100 filas por página para que el tiempo, ya que teóricamente cargar la vista de 100 datos, es mucho más rápido que cargar la vista de los 8600 datos.

Gráfico de resultados de la consulta Simple:

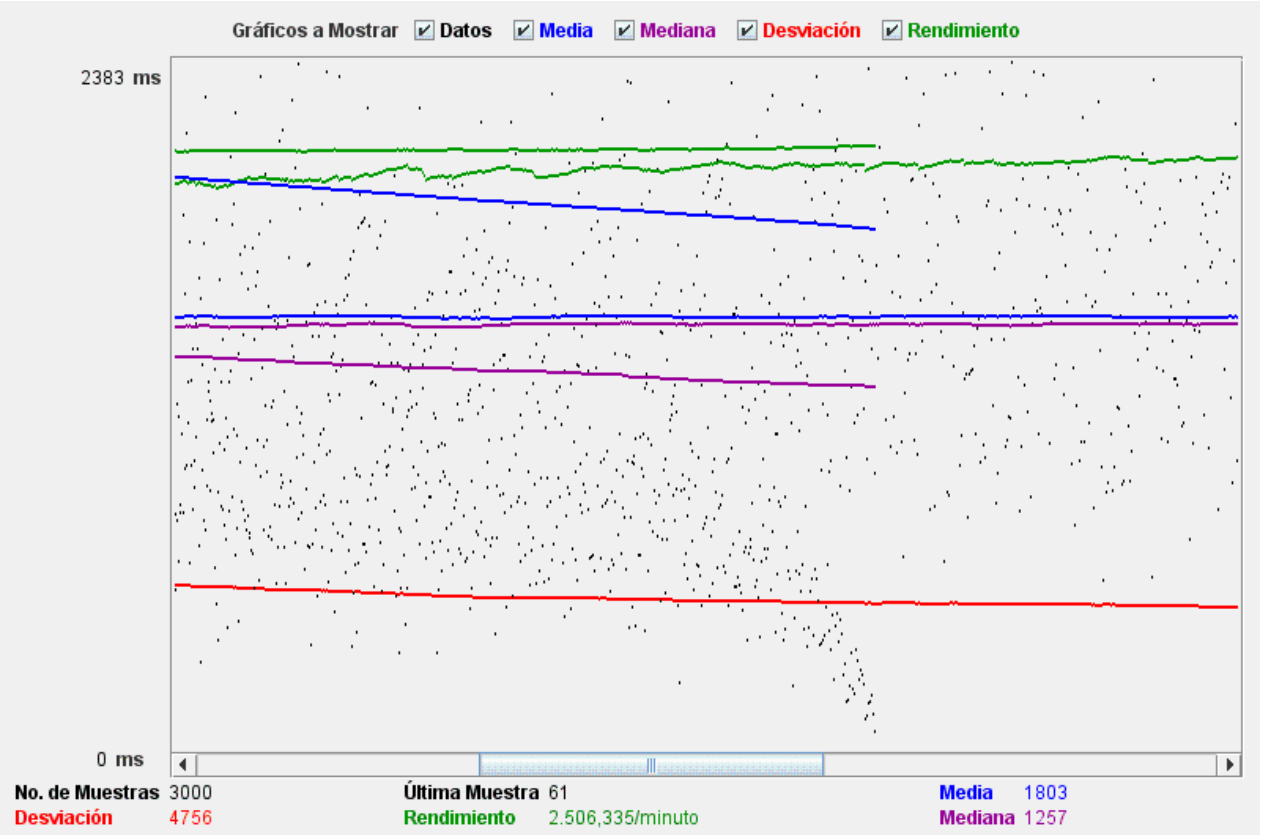


Gráfico de resultados de la consulta media:

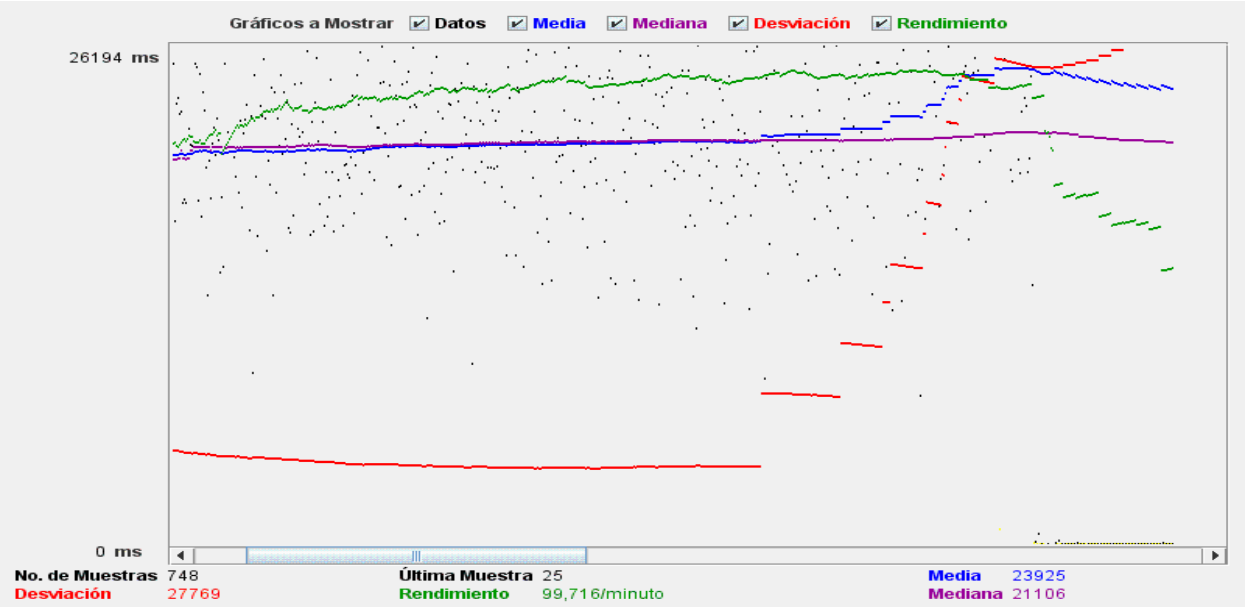
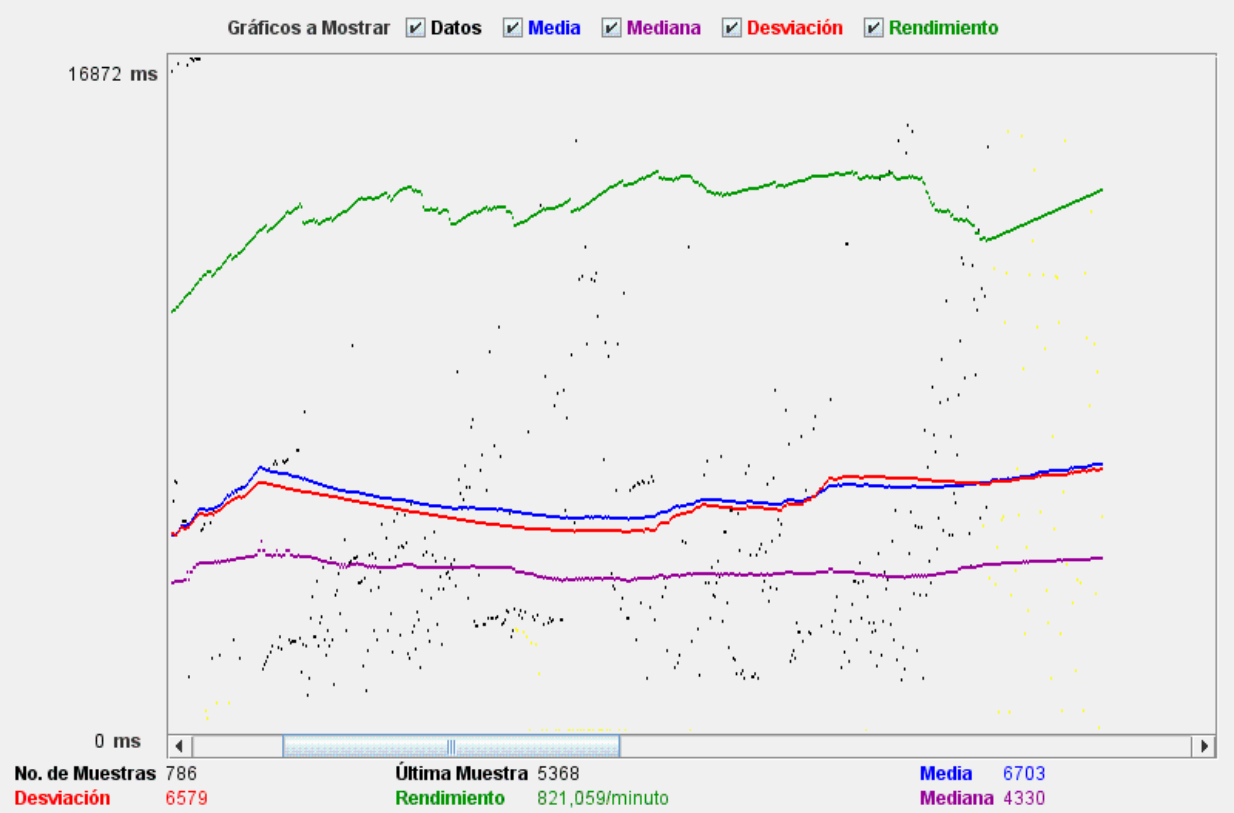


Gráfico de resultados de la consulta compleja:



3era Iteración (caché)

Para esta iteración activamos la memoria caché del controlador de codeigniter y da la base de datos (en la configuración de codeigniter en el archivo database.php) para que al realizar las consultas no se tuvieran que descargar denuevo los datos, y además guardar en RAM las últimas consultas realizadas.

Gráfico de resultados de la consulta Simple:

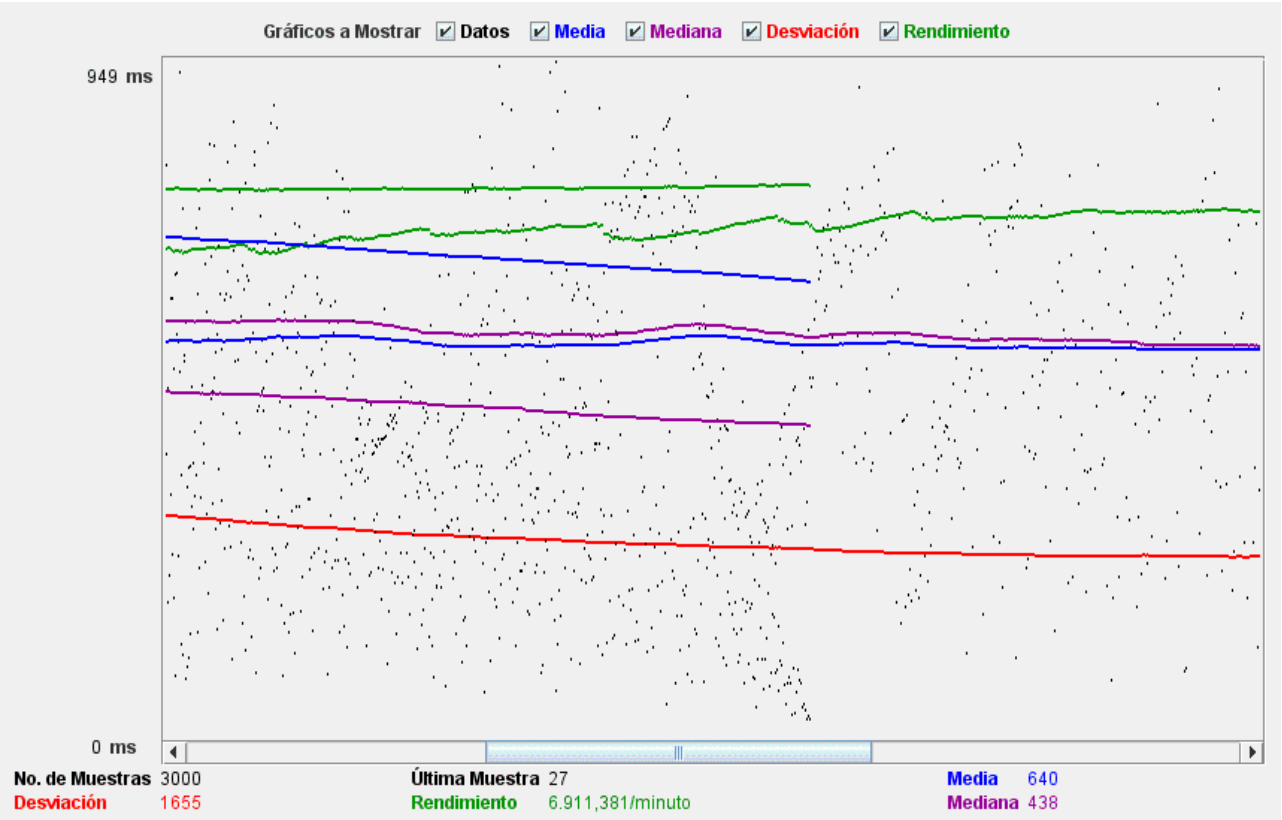


Gráfico de resultados de la consulta media:

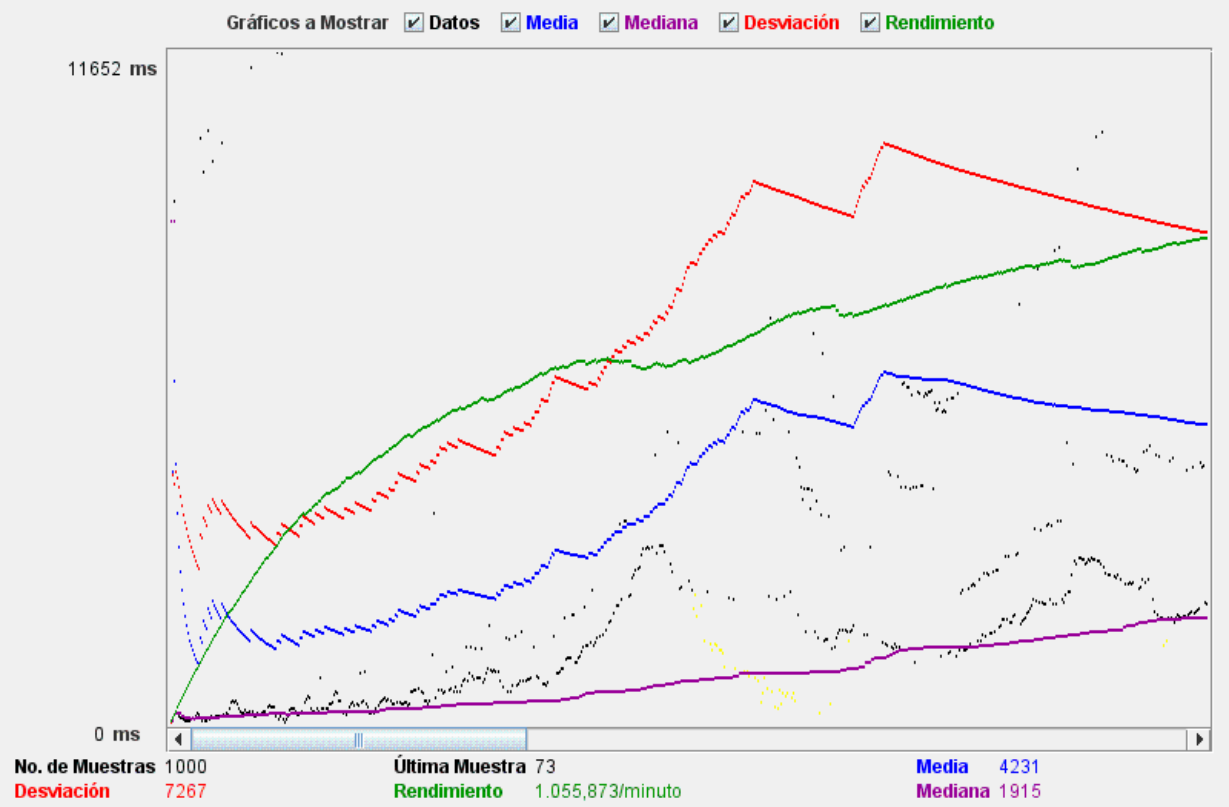
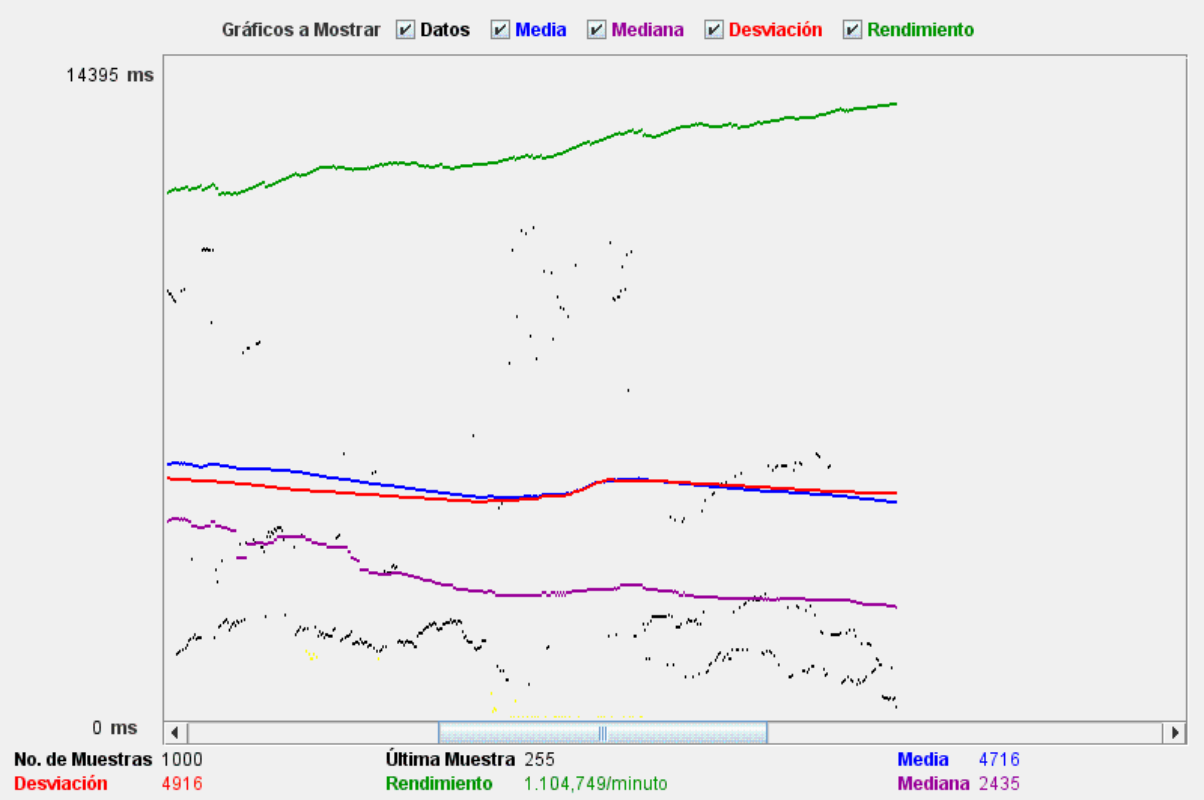


Gráfico de resultados de la consulta Compleja:



4ta Iteración (indexación DB)

Para esta iteración se consideró indexar la BD, con el fin de obtener un acceso rápido a esta, ya que al agregar un identificador a cada fila de una tabla permite aumentar la velocidad de acceso ya que permite ir directamente a la página asociada con cada entrada del índice. Para la indexación se usó el script indexación.sql que se encuentra en el directorio principal del proyecto. En los puntos 2.4.1, 2.4.2 y 2.4.3 se muestran los gráficos que resultan de las pruebas una indexado todas las tablas de la DB.

Gráfico de resultados de la consulta Simple:

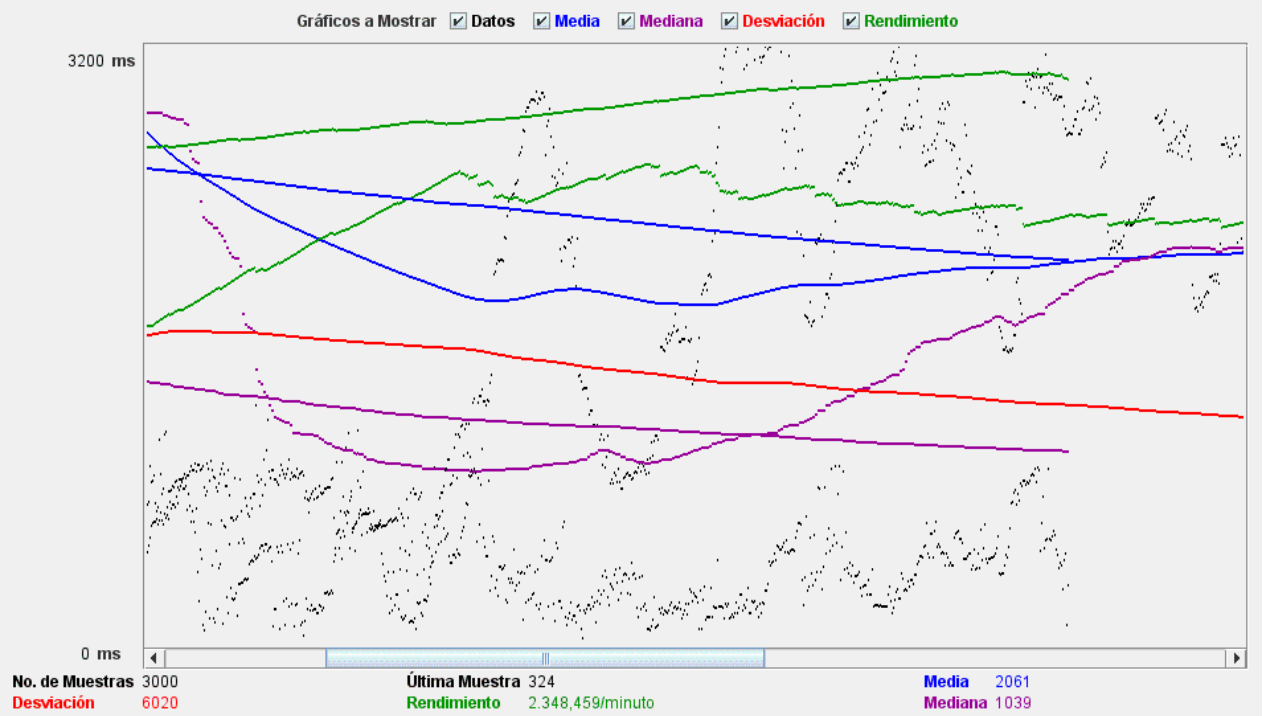


Gráfico de resultados de la consulta media:

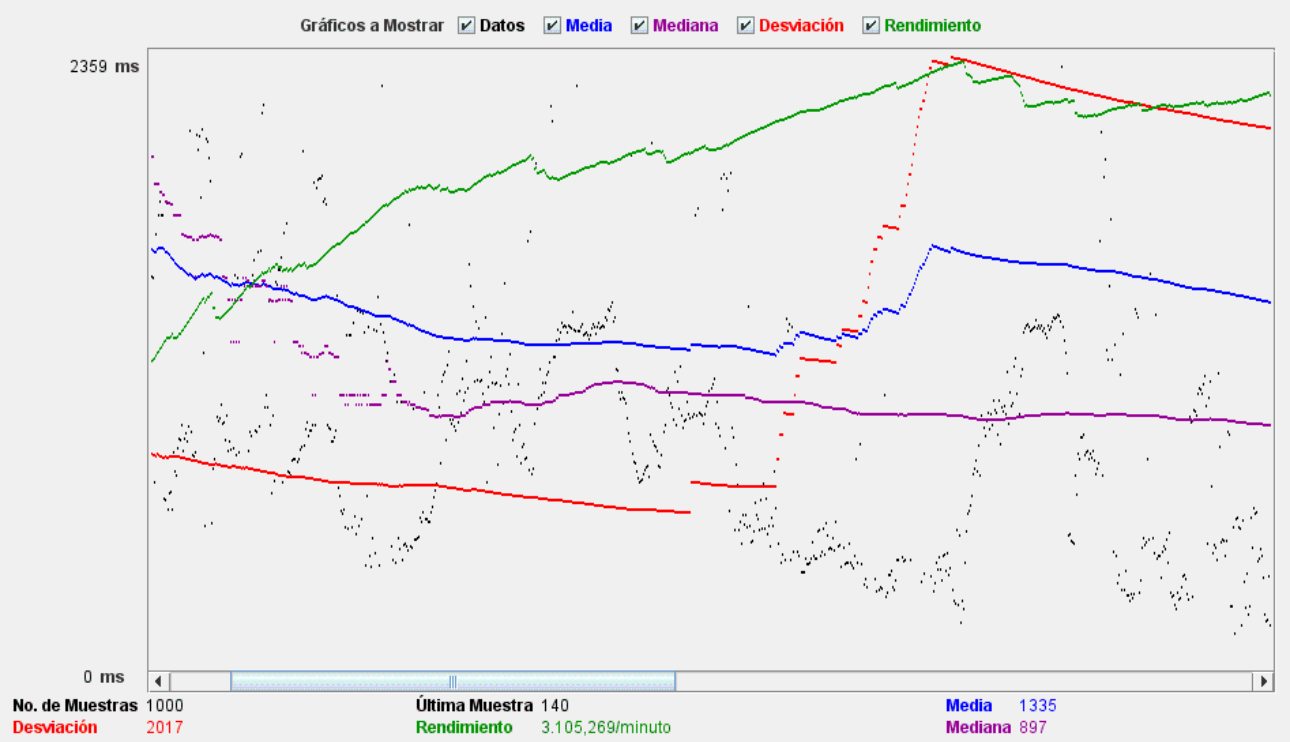
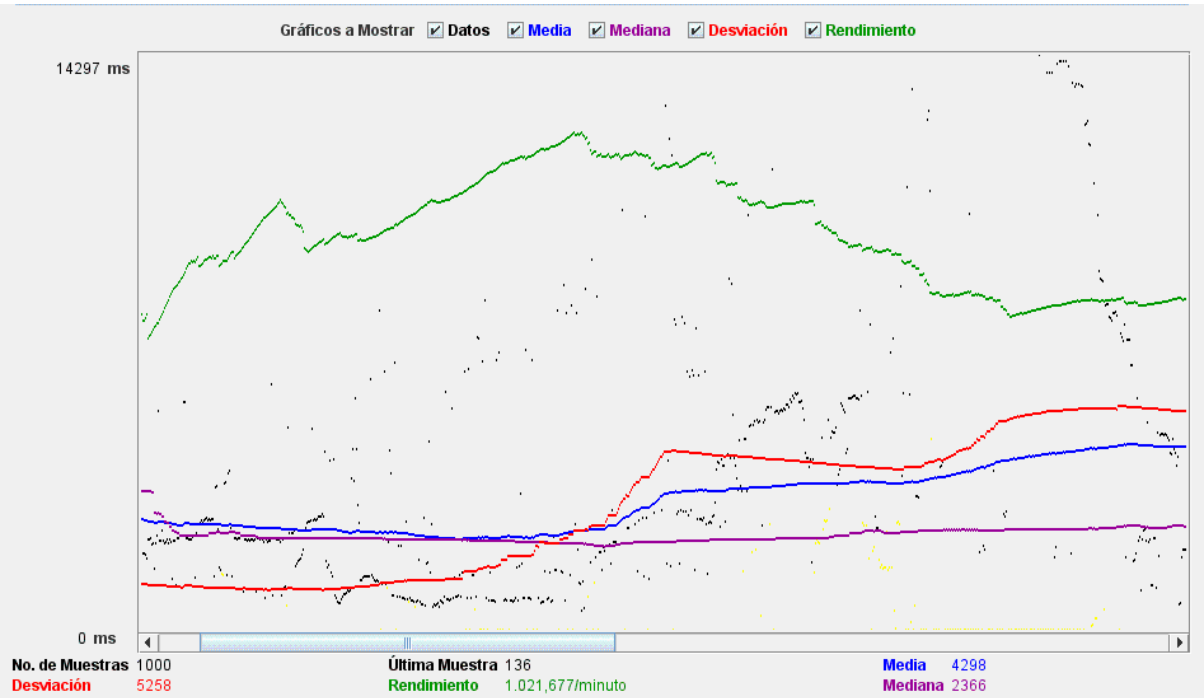


Gráfico de resultados de la consulta compleja:



5ta Iteración (Cambio de versión en la tecnología, y sistema operativo).

En esta iteración decidimos cambiar de sistema operativo a uno que tuviera menos procesos en segundo plano (en este caso ubuntu), además de ofrecernos un sistema más limpio, ligero y estable, de esta manera se puede obtener un mejor rendimiento en las consultas. Debido a que para montar el servidor utilizamos xampp, algunas incompatibilidades de codeigniter imposibilitaba su correcto funcionamiento, por lo cual lo actualizamos a la versión 3.

Gráfico de resultados de la consulta Simple:

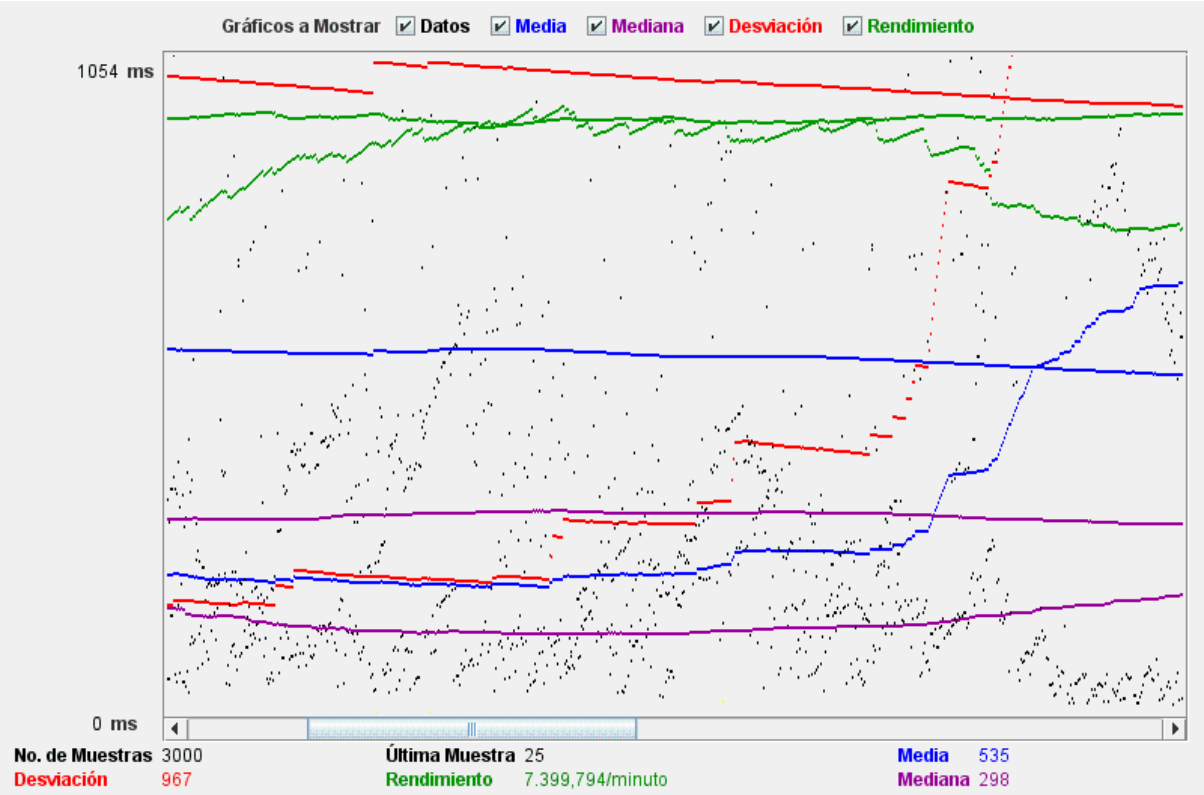


Gráfico de resultados de la consulta media:

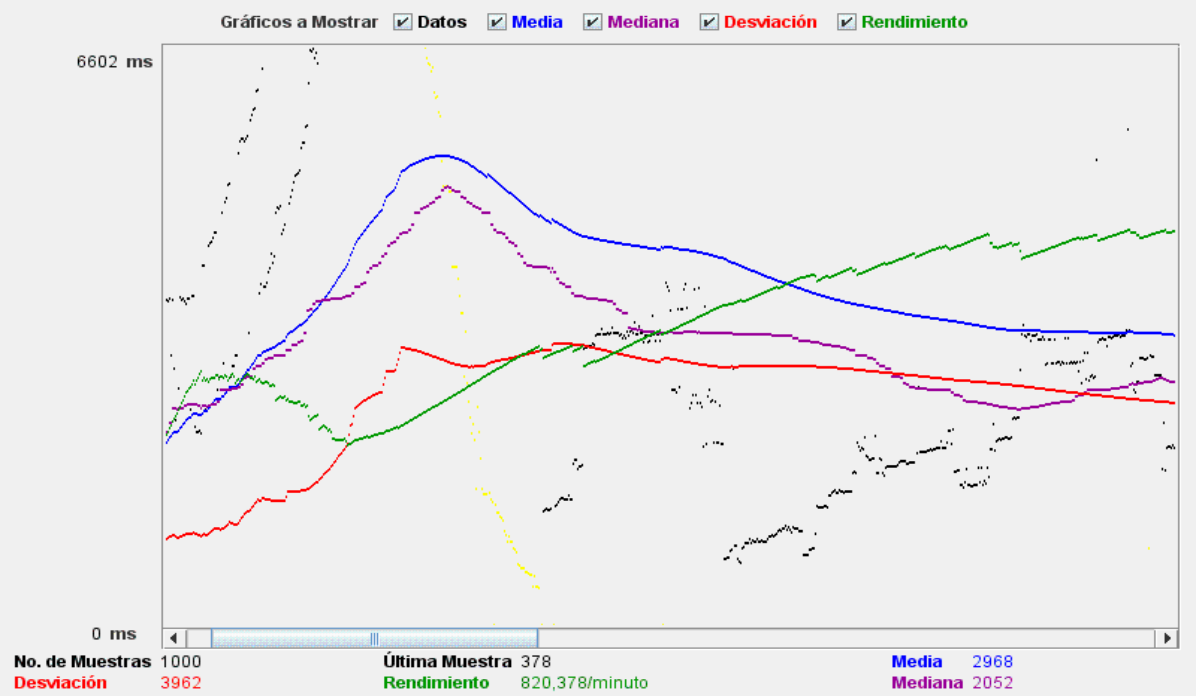


Gráfico de resultados de la consulta compleja:

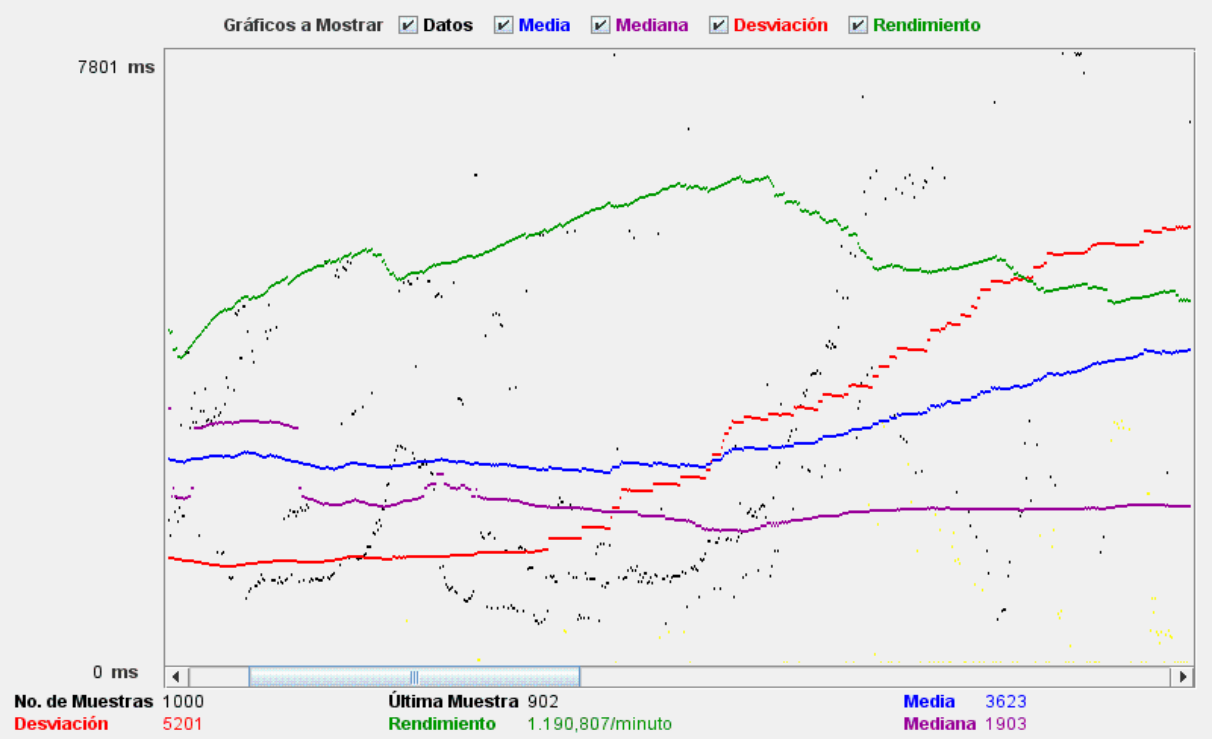


Tabla comparativa de las iteraciones

Consulta Simple

N° Iteración	N° Pruebas	Margen error	Throughput
1	3000	0.00%	49.7/s
2	3000	0.00%	41.8/s
3	3000	0.00%	115.2/s
4	3000	0.00%	39.1/s
5	3000	0.13%	123.3/s

Consulta Media

N° Iteración	N° Pruebas	Margen error	Throughput
1	100	32%	1.1/s
2	748	6.55%	1.7/s
3	100	5.10%	17.6/s
4	1000	0.00%	51.8/s
5	1000	0.10%	13.7/s

Consulta Compleja

N° Iteración	N° Pruebas	Margen error	Throughput
1	106	27.17%	26.6/m
2	786	21.50%	13.7/s
3	1000	20.60%	18.4/s
4	1000	14.00%	17.0/s
5	1000	10.60%	19.8/s

Conclusión

Al finalizar las optimizaciones, se puede apreciar que si bien los primeros cambios no fueron significativos, ya que en las primeras optimizaciones la consulta media y compleja seguían mostrando un margen de error promedio de 25%, sin embargo a medida que aumentaban las iteraciones se pudo lograr reducir este porcentaje.
Como resultado de esta experiencia se pudo apreciar que para optimizar los tiempos de respuesta, no es necesario cambiar la tecnología utilizada, framework y versiones, ya que no muestran una mejora significativa. Por otro lado, la indexación de la BD y optimización de las consultas siguen mostrando aumentos importantes en lo que corresponde a rendimiento frente a estas.
Además se pudo apreciar que la conexión a internet que se tenga entre el servidor y el cliente influye de manera significativa, una conexión deficiente genera errores sobre el 70%, lo que conlleva a ensuciar las pruebas que se realizan, ya que no entregar resultados reales.