Caso 3

Infraestructura computacional

Nicolás Gómez - 201426109
Christian Potdevin - 201424177

1. Cambios para medir los indicadores

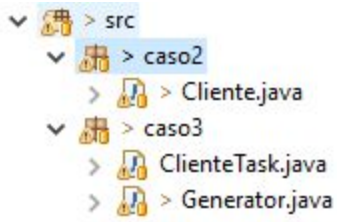
Se decidió que el uso de la CPU en % se iba a medir en el Servidor, y todos los demás datos, es decir tiempo de respuesta para autenticar a los participantes y tiempo de respuesta a una consulta, en el cliente. A continuación se presentan los cambios que tuvieron que realizarse para implementar la generación de estos datos.

Cambios en el servidor:

Para generar los datos de uso de CPU se desarrolló una clase llamada CPUDataRecorder que extiende Thread. Esto con el propósito de que el servidor la pueda correr al principio antes de que empiece a recibir clientes. Al correr esta clase esta empieza a grabar el porcentaje de uso de CPU cada segundo en un archivo csv. Finalmente, al terminar de atender todos los clientes de una prueba y detener el servidor el escritor de archivos BufferedWriter se cierra y el archivo queda guardado en una carpeta data del proyecto.

Cambios en el cliente:

Para generar los datos de tiempos de autorización y de consulta, se utilizó la librería GLoad. La estructura del cliente quedó de la siguiente manera:



La clase ClienteTask era la encargada de crear los Clientes y ejecutar su método, mientras que la clase Generator generaba la cantidad deseada de Clientes con el tiempo que se deseara entre ellos.

En la clase del cliente también se tuvieron que efectuar unos cuantos cambios. En primer lugar, se modificó la dirección a la que se conectaba el cliente de localhost a la dirección IP del servidor. En segundo lugar, se creó un atributo de tipo BufferedWriter mediante el cual el cliente era capaz de escribir los tiempos que obtuviera al archivo .csv apropiado. Para medir el tiempo de autenticación, se toma el tiempo inicial en el momento que se recibe la llave

simétrica cifrada con la llave pública, y se resta este tiempo inicial al momento en que el servidor confirma que se ha realizado la autenticación. Por otro lado, el tiempo de consulta se mide desde que se genera la consulta hasta que el servidor la responde de la misma manera que el tiempo de autorización. Finalmente, las fallas se miden mediante una variable en Generator que cada cliente puede incrementar mediante un método estático en caso que no pueda conectarse, y se imprime este valor al final del archivo que se genera.

2. Identificación de la plataforma

MacBook Pro (Servidor):

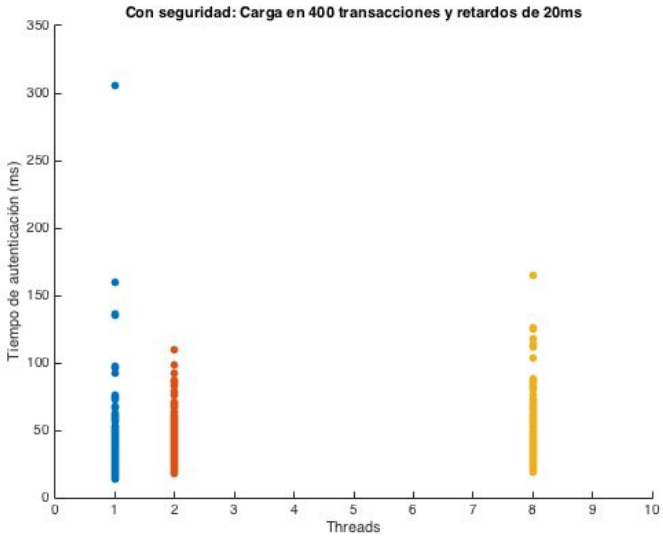
- Arquitectura de 64 bits
- Procesador quad-core de 2.3 GHz, intel core i7.
- 8Gb de RAM, 1600MHz DDR3
- 1Gb asignado a la JVM

Dell Inspiron 5547 (Cliente):

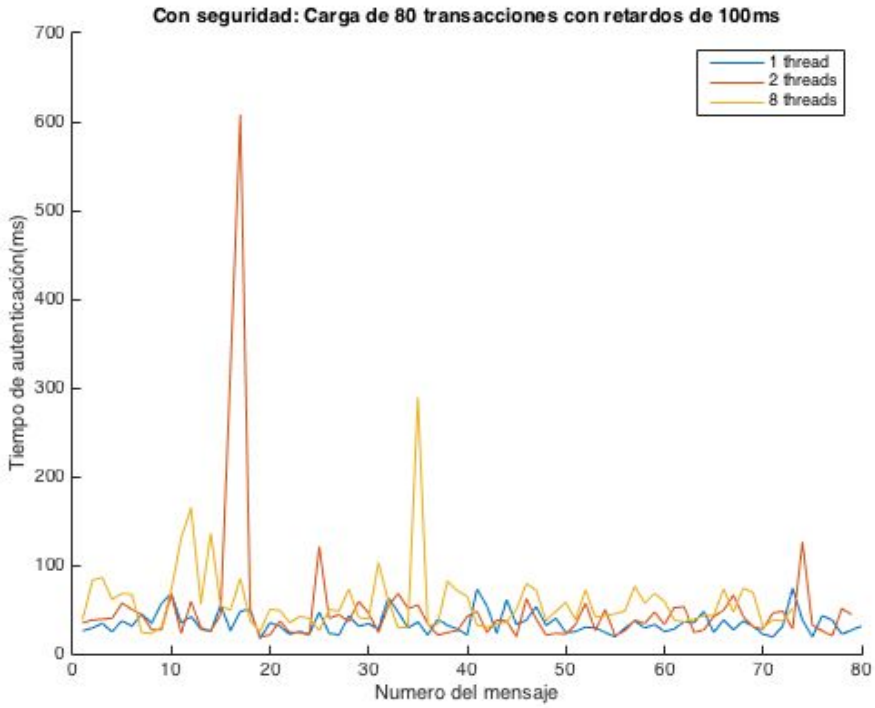
- Arquitectura de 64 bits
- Procesador quad-core de 2.6 GHz, intel core i7.
- 8Gb de RAM
- 1Gb asignado a la JVM

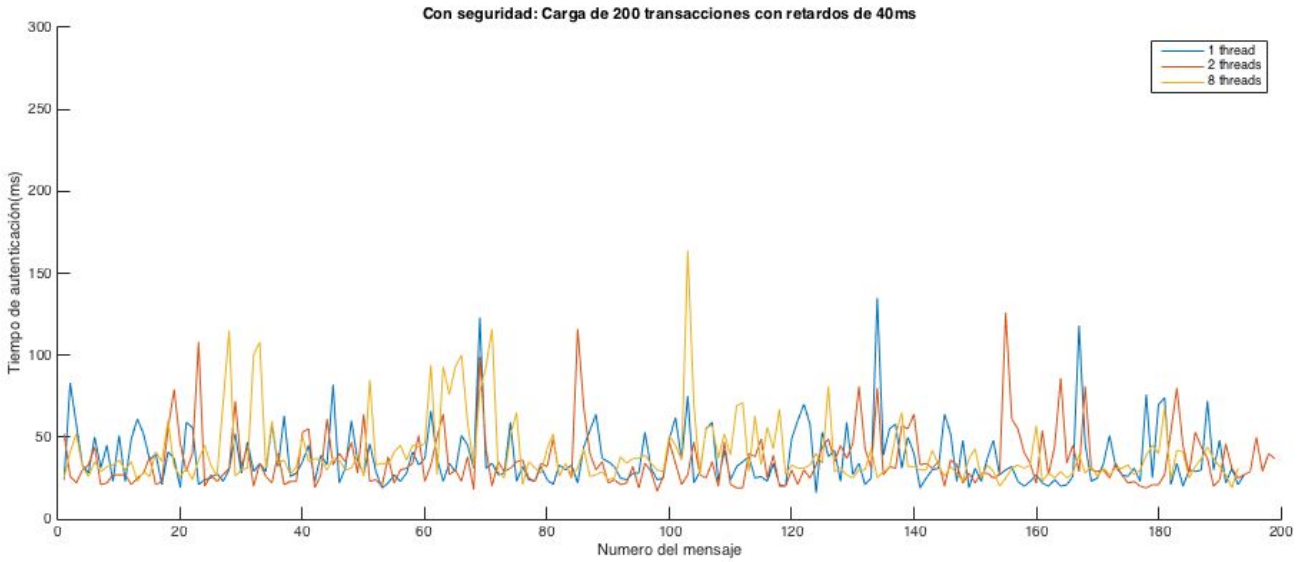
3. Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia

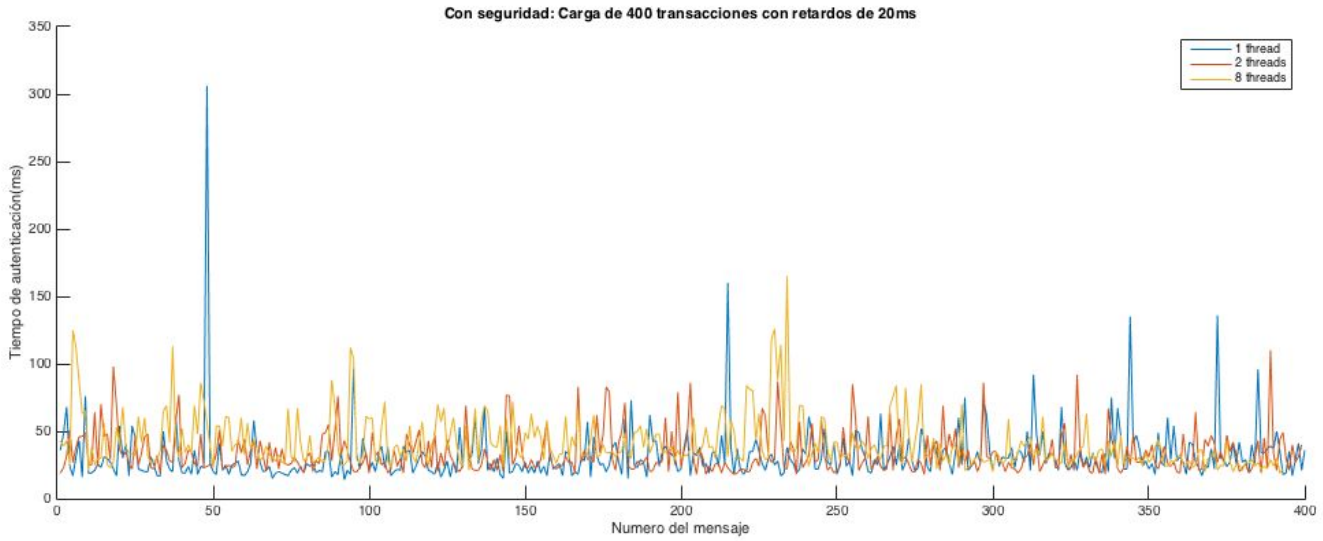
a. Luego de graficar el número de threads contra el tiempo de autenticación para una carga de 400 transacciones y iniciadas con retardos de 20ms se obtuvo la siguiente gráfica.



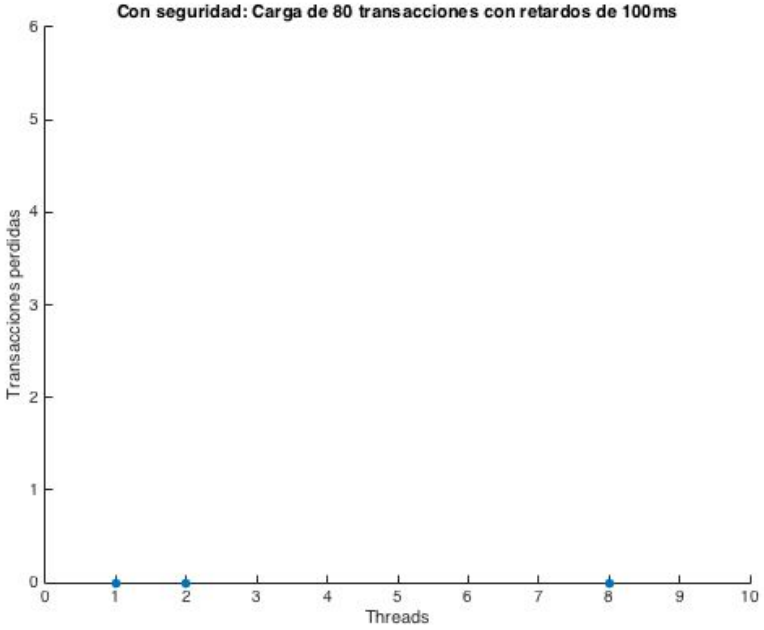
Al observar esta grafica se determino que esta no mostraba los datos de forma concreta. Por esta razón se decidió graficar el tiempo de autenticación de los clientes contra su orden cronológico y generar tres curvas diferentes para 1, 2 y 8 threads. Estas gráficas son más significativas porque evidencian el comportamiento del tiempo de autenticación a lo largo de una prueba de un servidor de 1, 2 u 8 threads.

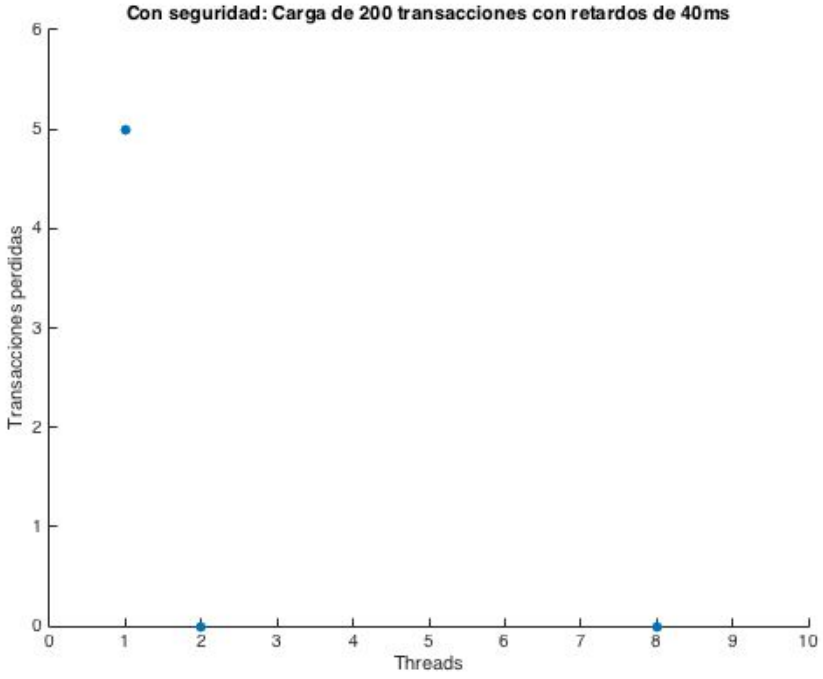


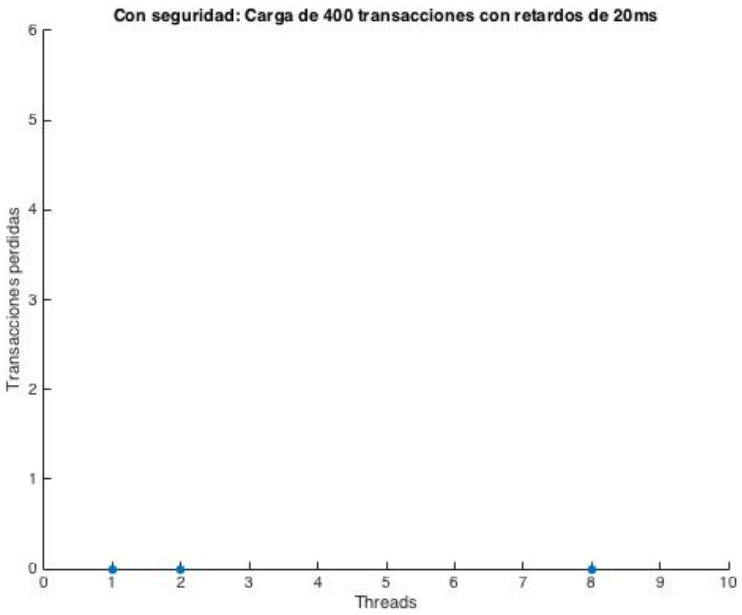




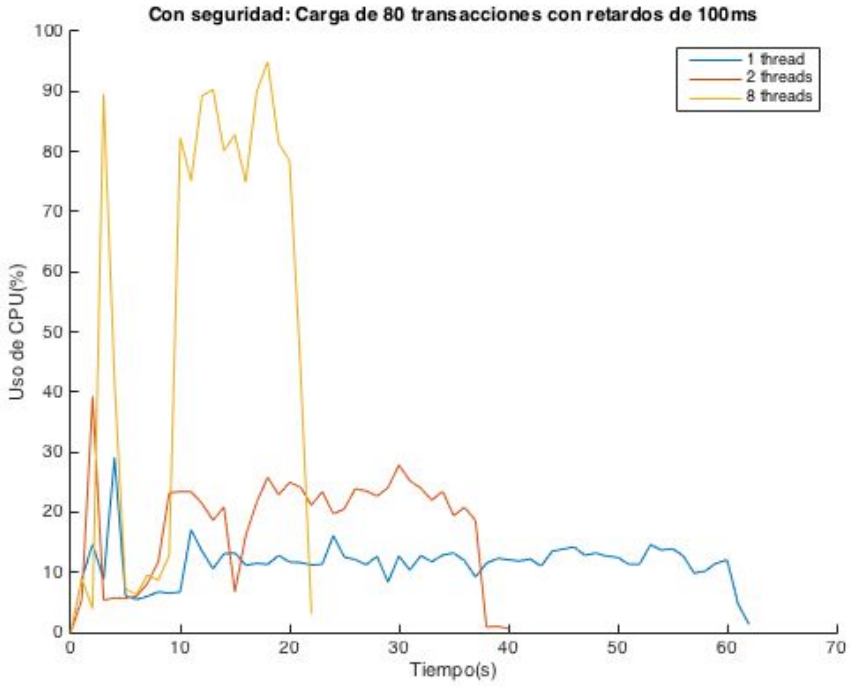
b. Debido a que la mayoría de pruebas presentaron cero o muy pocas fallas las graficas de número de threads contra fallas no son muy utiles. Estas se muestran a continuación.

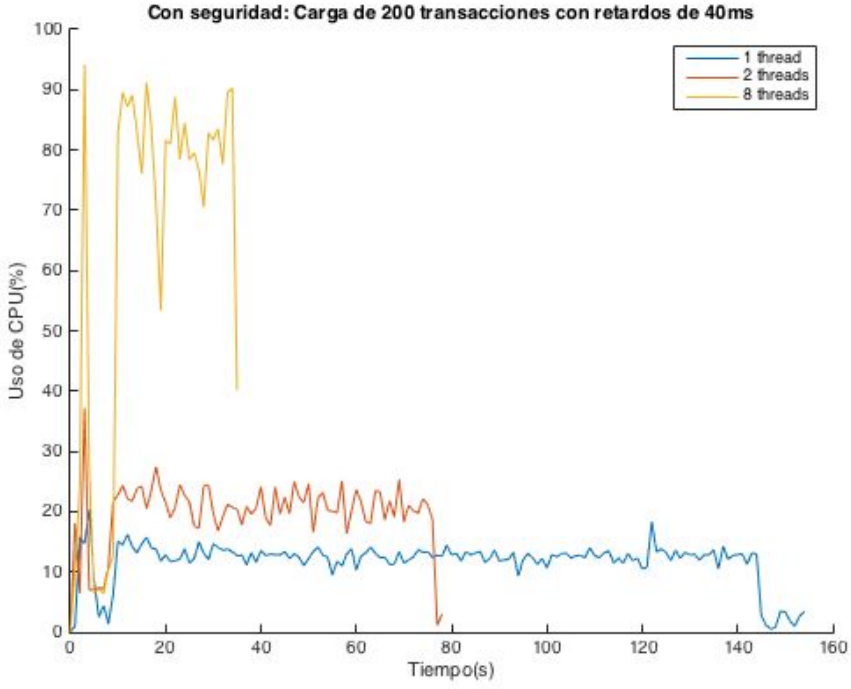

Con seguridad: Carga de 80 transacciones con retardos de 100ms
Transacciones perdidas
Threads
0
1
2
3
4
5
6
7
8
9
10

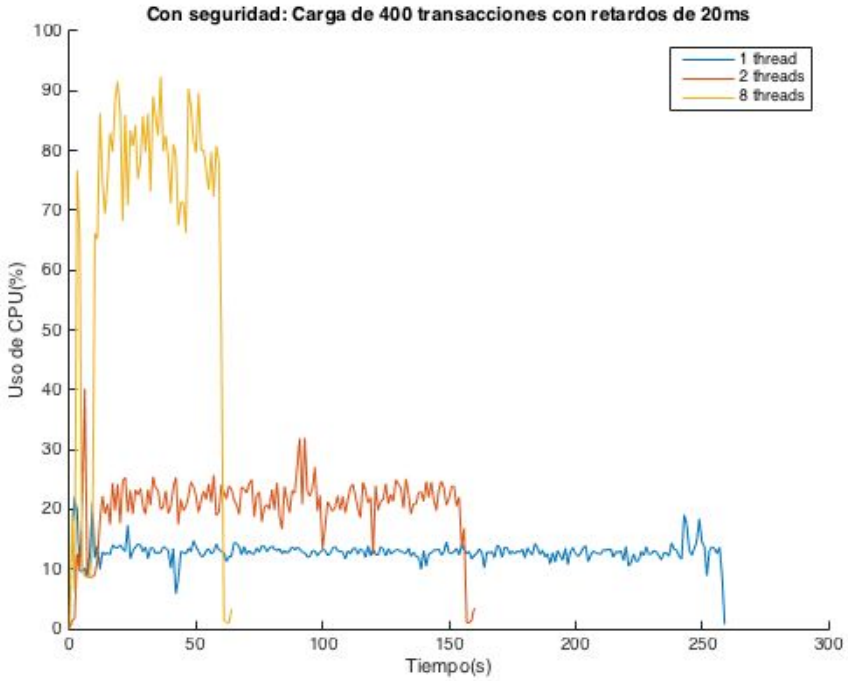

Con seguridad: Carga de 200 transacciones con retardos de 40ms
Transacciones perdidas
Threads
0
1
2
3
4
5
6
7
8
9
10



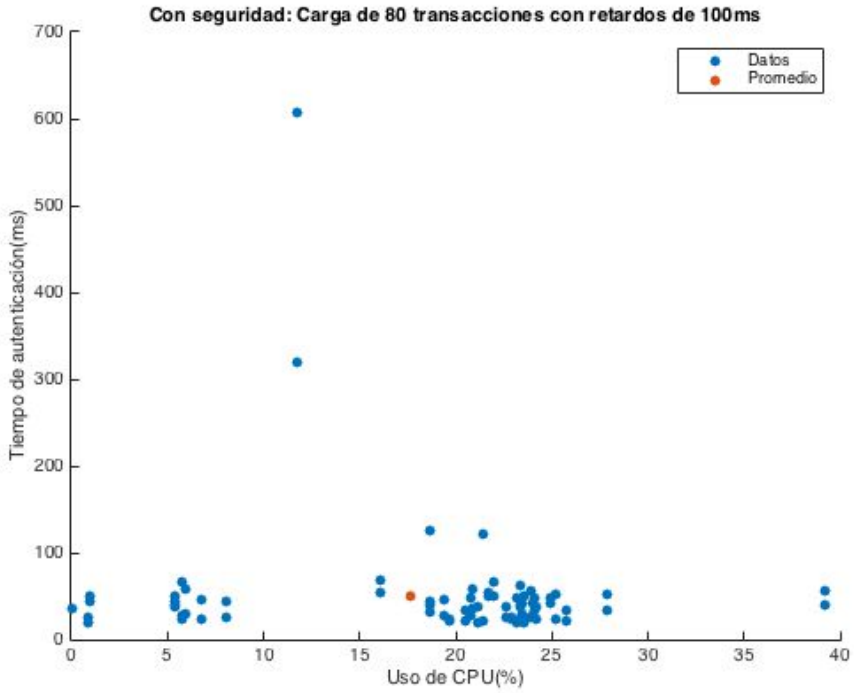
c. Al graficar los datos de uso de CPU contra número de threads se vio un problema similar al ocurrido en el punto a. Por esta razón se decidió generar la gráfica de porcentaje de uso de CPU contra tiempo y hacer una curva para 1, 2 y 8 threads. Estas gráficas se pueden observar abajo. Al observar estas gráficas se puede concluir que entre mas threads se utilizan para una aplicación con seguridad menos se demora la prueba y mas porcentaje de la CPU se utiliza. En los tres casos se puede ver como al pasar de 1 a 2 threads la duración de la prueba pasa a tomar poco mas de la mitad del tiempo y utilizar poco menos del doble del porcentaje de CPU. Similarmente, si se observa la diferencia entre las pruebas con 2 y 8 threads se puede ver que el comportamiento sigue siendo similar al descrito anteriormente y da la conclusión de que entra mas threads se utilicen mayor porcentaje de la CPU se va a utilizar y menor tiempo va a tomar realizar la prueba. Ademas, si se observa la zona en la que se encuentran la mayoría de los datos de porcentaje de uso de CPU se puede concluir que estos tienen una relación casi lineal con la cantidad de threads con los que se corre la prueba. En este caso concluimos una ecuación lineal y simple que podría modelar este comportamiento es C = 10*T donde C es el porcentaje de uso de CPU y T es el número de threads utilizados en la prueba.

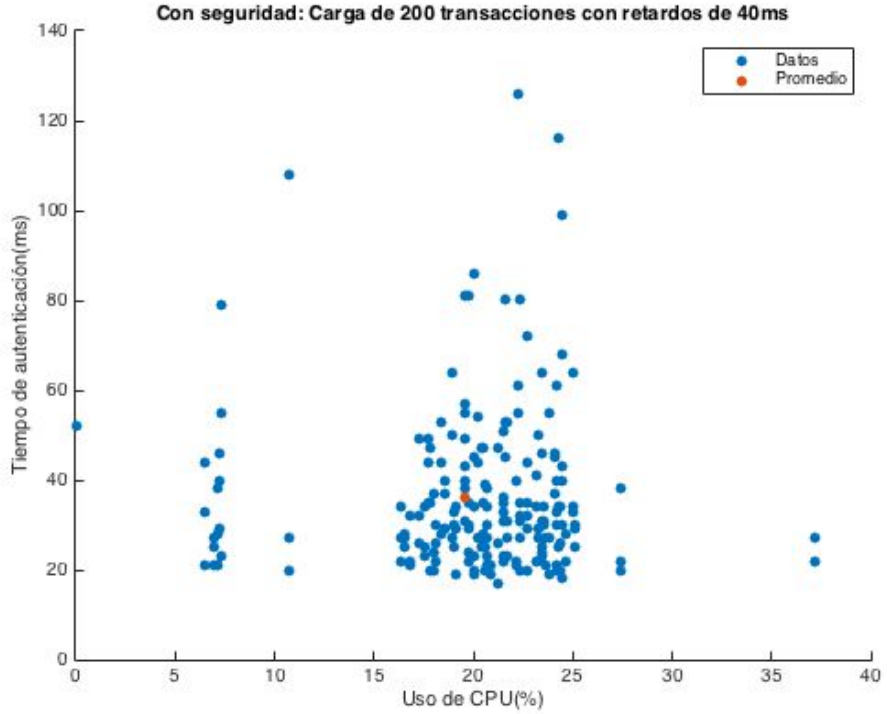

Con seguridad: Carga de 80 transacciones con retardos de 100ms
1 thread
2 threads
8 threads
Uso de CPU(%)
Tiempo(s)
0
10
20
30
40
50
60
70
80
90
100

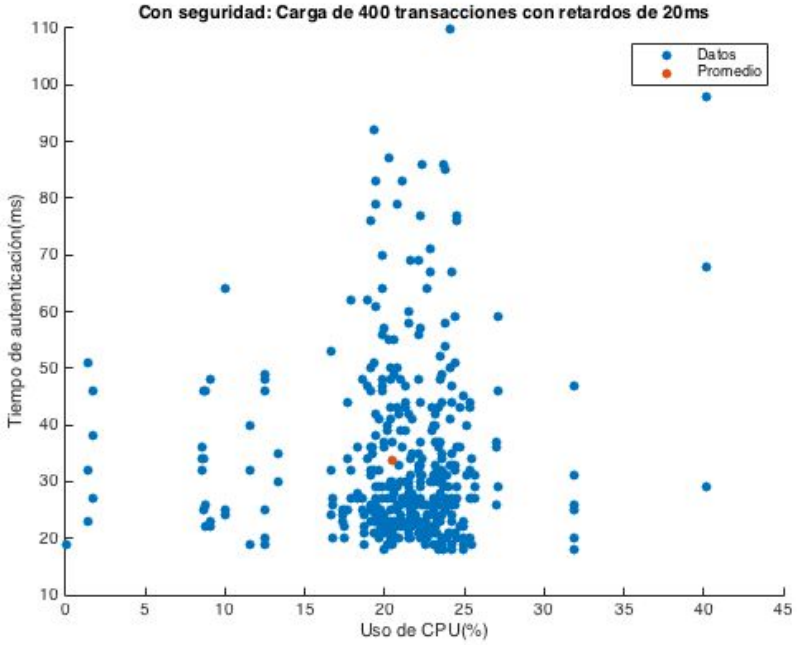

Con seguridad: Carga de 200 transacciones con retardos de 40ms
1 thread
2 threads
8 threads
Uso de CPU(%)
Tiempo(s)
0
20
40
60
80
100
120
140
160



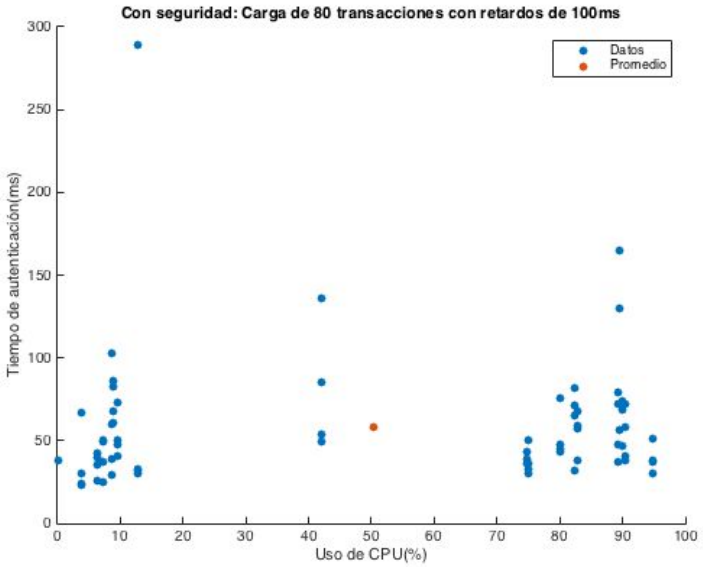
d. Para las gráficas de tiempo de autenticación contra uso de CPU nos encontramos con el problema de que no había un dato de uso de CPU por cada dato de tiempo de autenticación. Sin embargo como se tienen los datos de autenticación en orden cronológico podemos relacionar uno de estos datos con un momento en el tiempo del corrido del programa (asumiendo que están separados cada uno por la misma cantidad de tiempo) y ese momento se puede relacionar con un porcentaje de uso de CPU. De esta forma podemos relacionar los datos de tiempo de autenticación con el uso de CPU. El único problema es que debido a que a veces el programa dura muy poco comparado con la cantidad de clientes que hay entonces al mapear los datos de CPU a los de tiempos de autenticación los segundos tienden a repetirse mucho y pueden llegar a distorsionar el gráfico. Estos datos se pueden observar en las siguientes gráficas junto con el promedio de cada set y además se pueden ver en el siguiente punto (c) donde fue necesario aplicar el mismo procedimiento. Para este punto (d) podemos observar que la ubicación de la mayoría de puntos en estas gráficas, y el promedio, se encuentran cerca de la misma zona. Esto nos dice que un uso de entre el 15 y el 25% de CPU corresponde a un tiempo de autenticación de entre 20 y 60ms para una prueba con una cantidad de 2 threads.

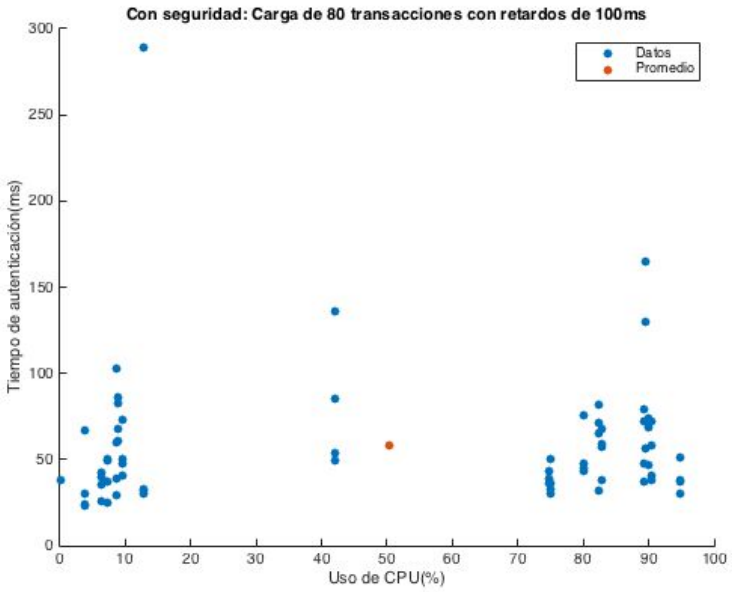

Con seguridad: Carga de 80 transacciones con retardos de 100ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
100
200
300
400
500
600
700
0
5
10
15
20
25
30
35
40

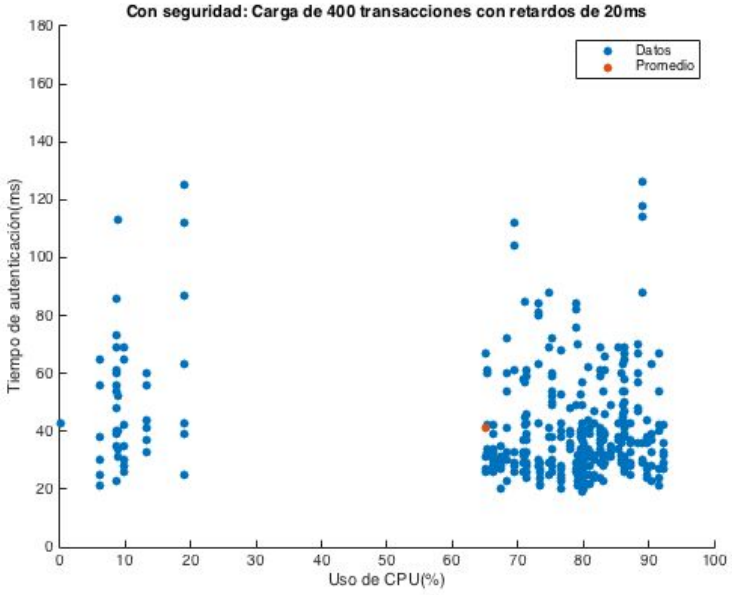

Con seguridad: Carga de 200 transacciones con retardos de 40ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
20
40
60
80
100
120
140
0
5
10
15
20
25
30
35
40



e. En este punto podemos observar que al aumentar el número de threads a 8 la zona en la que estan la mayoria de datos se mueve hacia la derecha pero no hacia arriba. Es decir que aumenta el rango de porcentajes de uso de CPU pero no aumenta el rango de tiempos de autenticación. En este caso un uso de entre el 70 y el 95% de CPU corresponde a un tiempo de autenticación de entre 20 y 60ms para una prueba con una cantidad de 8 threads.



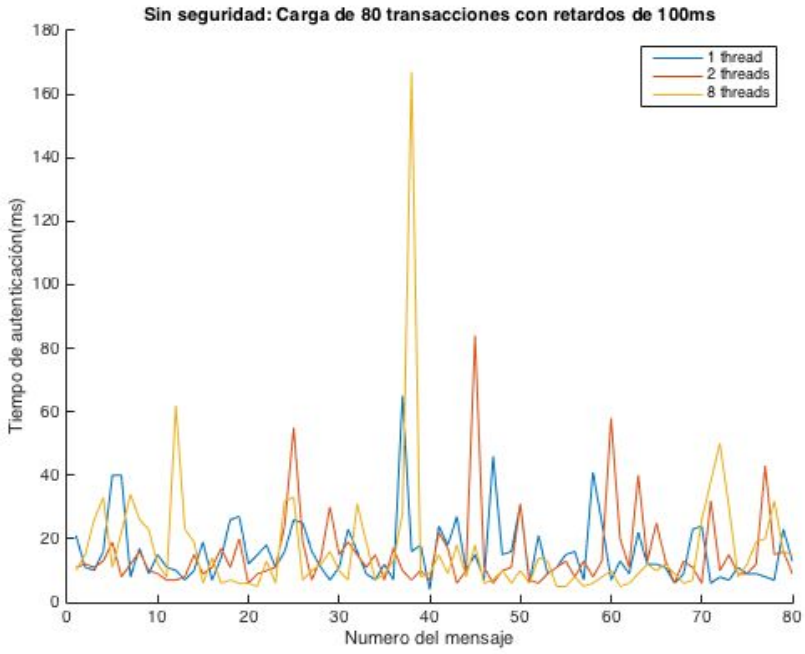

Con seguridad: Carga de 80 transacciones con retardos de 100ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
50
100
150
200
250
300
0
10
20
30
40
50
60
70
80
90
100

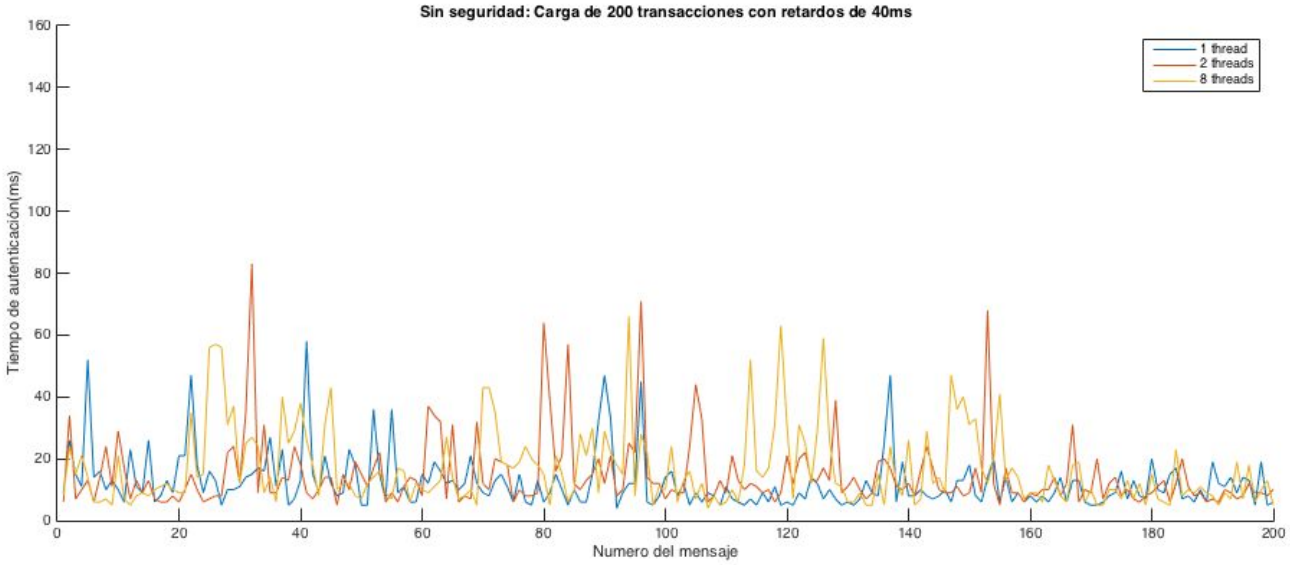

Con seguridad: Carga de 400 transacciones con retardos de 20ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
20
40
60
80
100
120
140
160
180
0
10
20
30
40
50
60
70
80
90
100

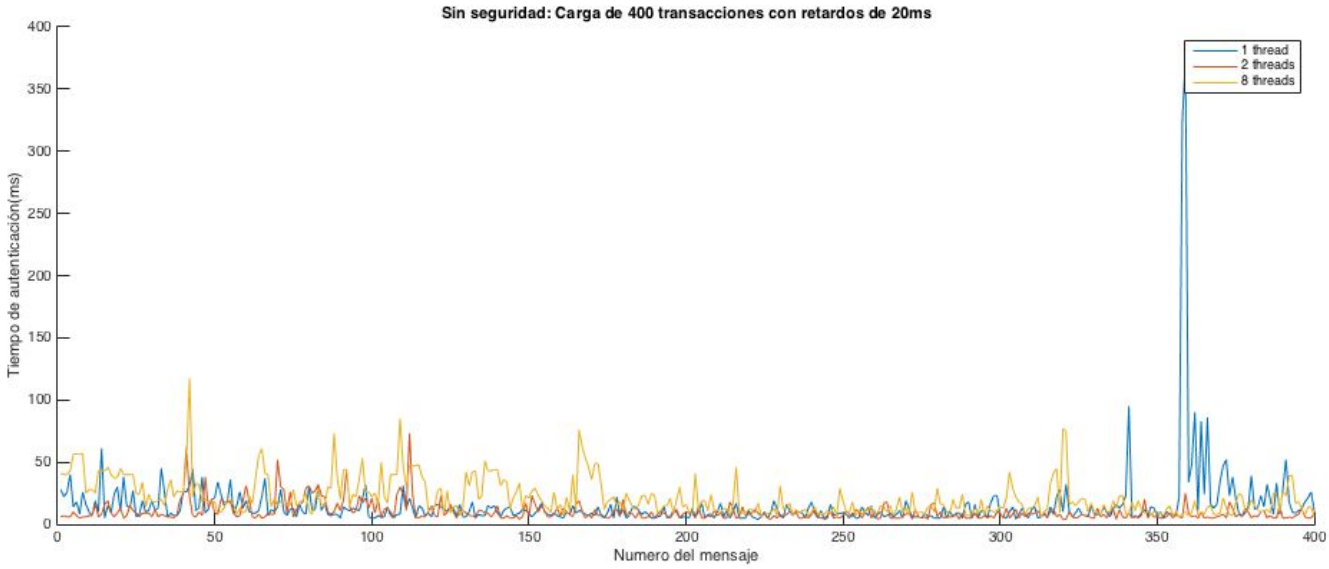
4. Comportamiento de la aplicación ante diferentes niveles de seguridad

4.1 Gráficas

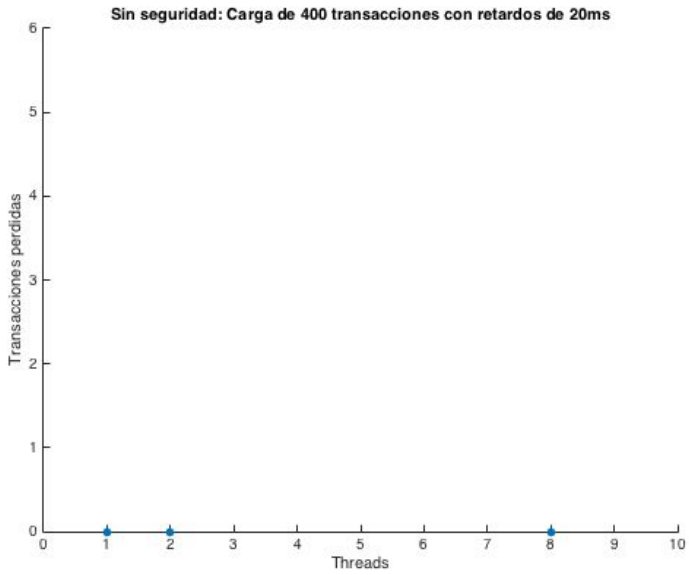
a. Habiendo determinado en el punto 3 cómo representar de manera más adecuada las gráficas de tiempo de autenticación vs número de threads, se repite el procedimiento para el sistema sin seguridad

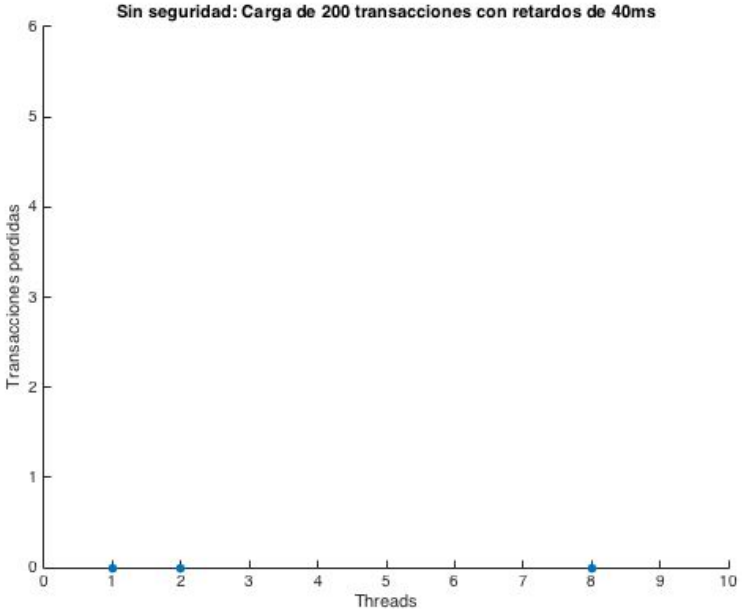


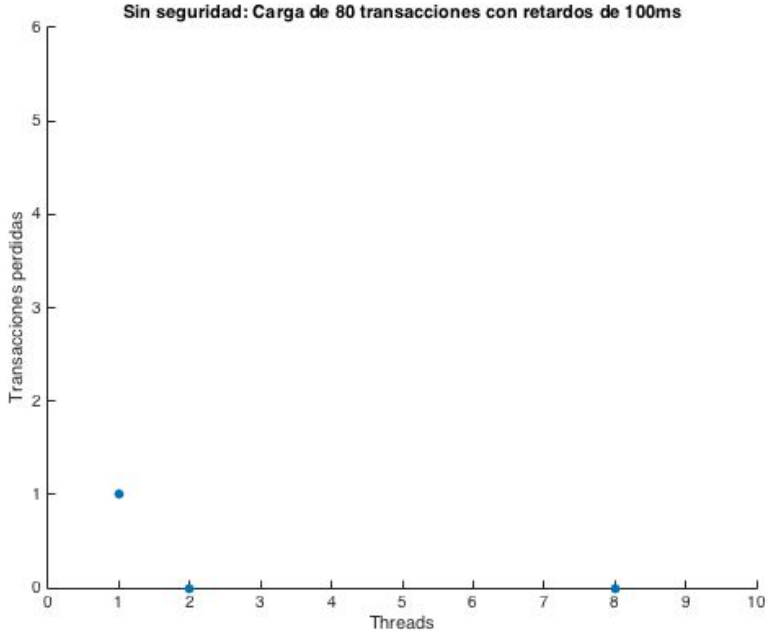




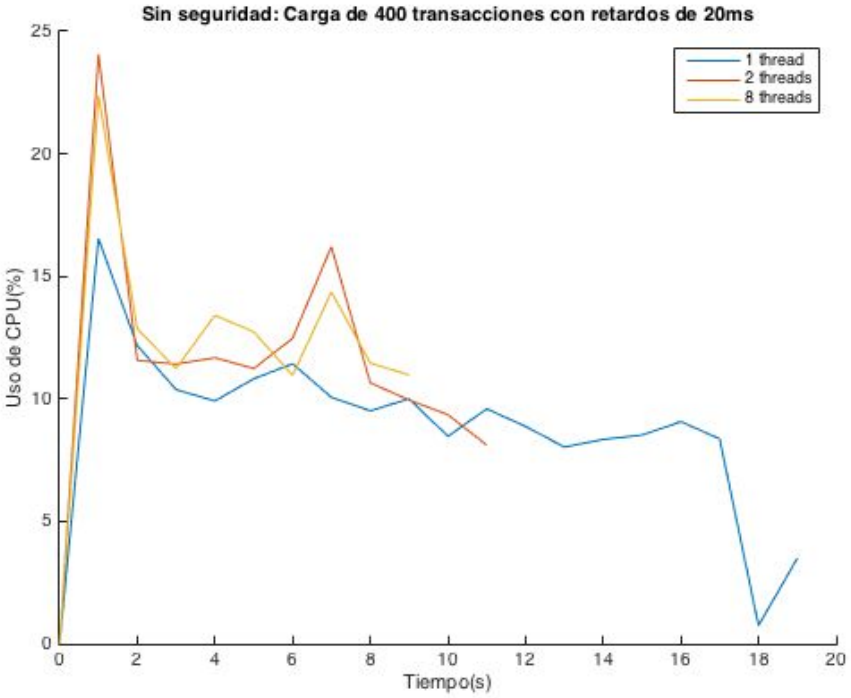
b. Desafortunadamente, al igual que cuando se corrió el sistema con seguridad, al correrlo sin seguridad no se obtuvieron fallas.

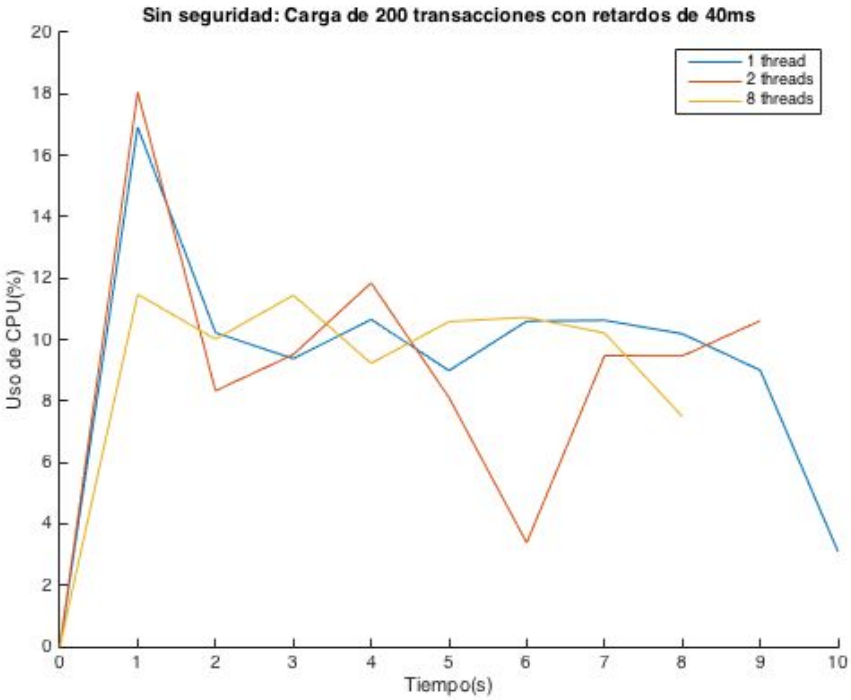


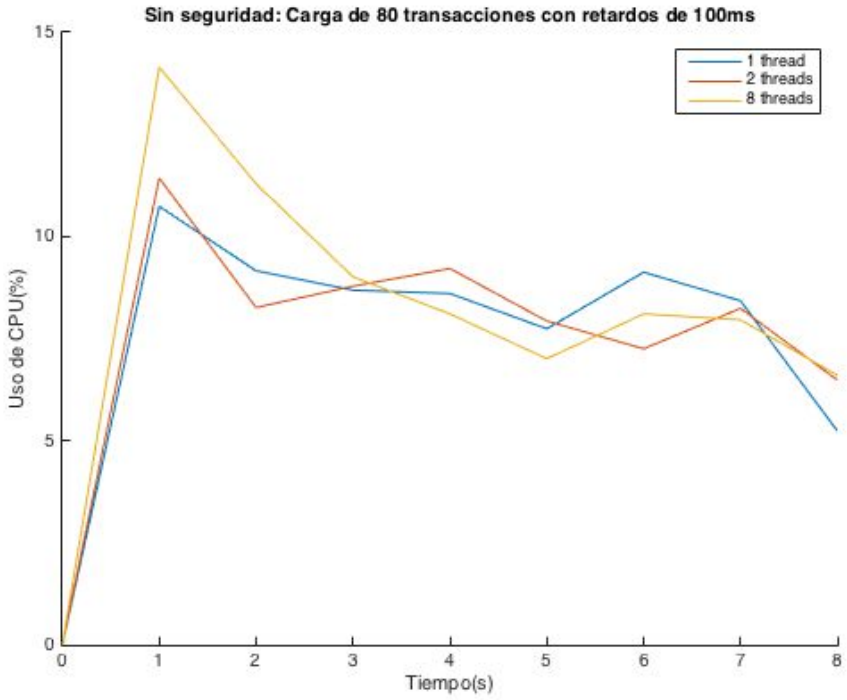

Sin seguridad: Carga de 200 transacciones con retardos de 40ms
Transacciones perdidas
Threads
0
1
2
3
4
5
6
7
8
9
10


Sin seguridad: Carga de 80 transacciones con retardos de 100ms
Transacciones perdidas
Threads
0
1
2
3
4
5
6
7
8
9
10

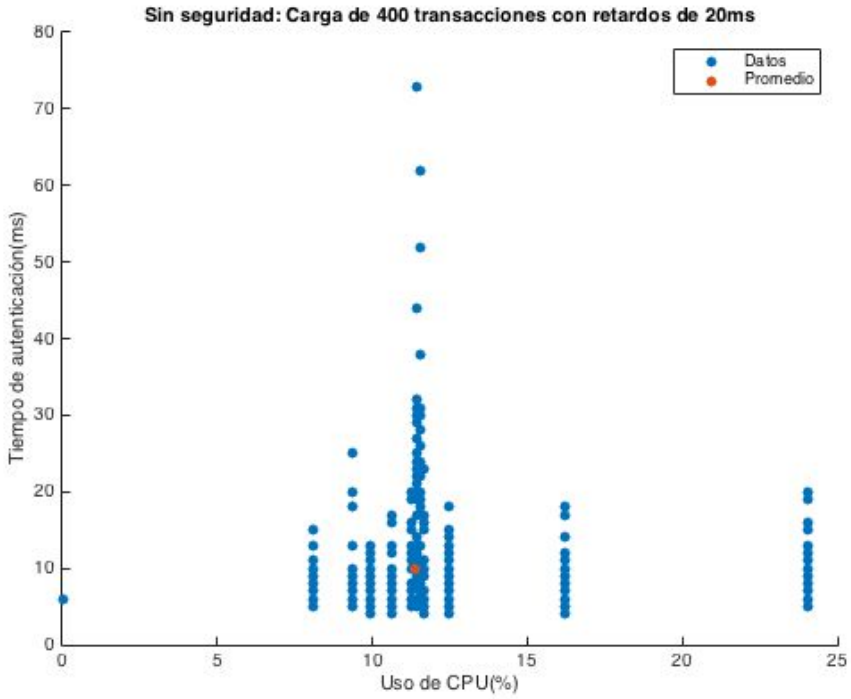
C. Al igual que en el sistema con seguridad, a mayor cantidad de threads, menor tiempo de ejecución y mayor porcentaje de uso del CPU. En este caso sin embargo como los valores son considerablemente más pequeños resulta un poco más difícil identificar esta relación

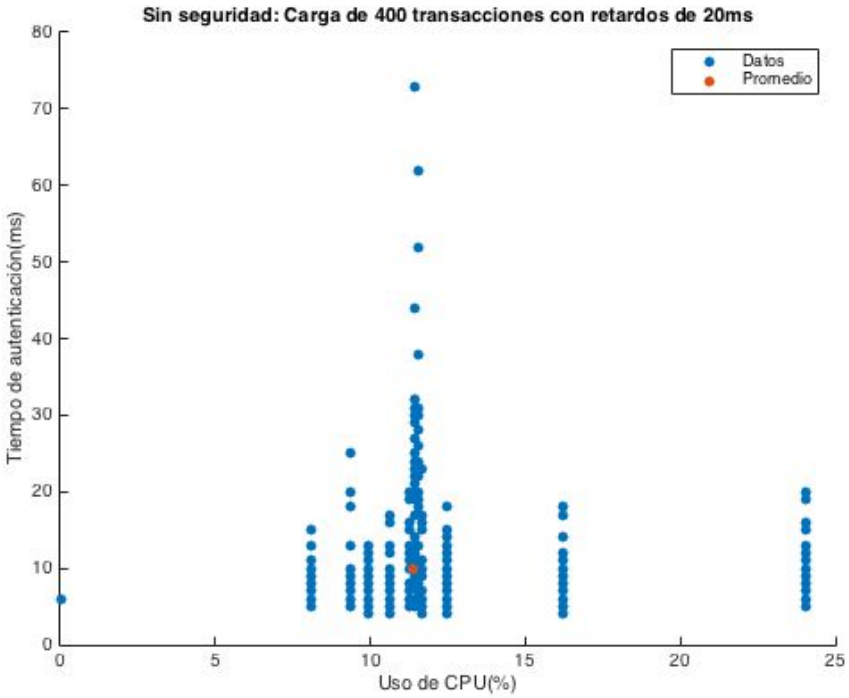


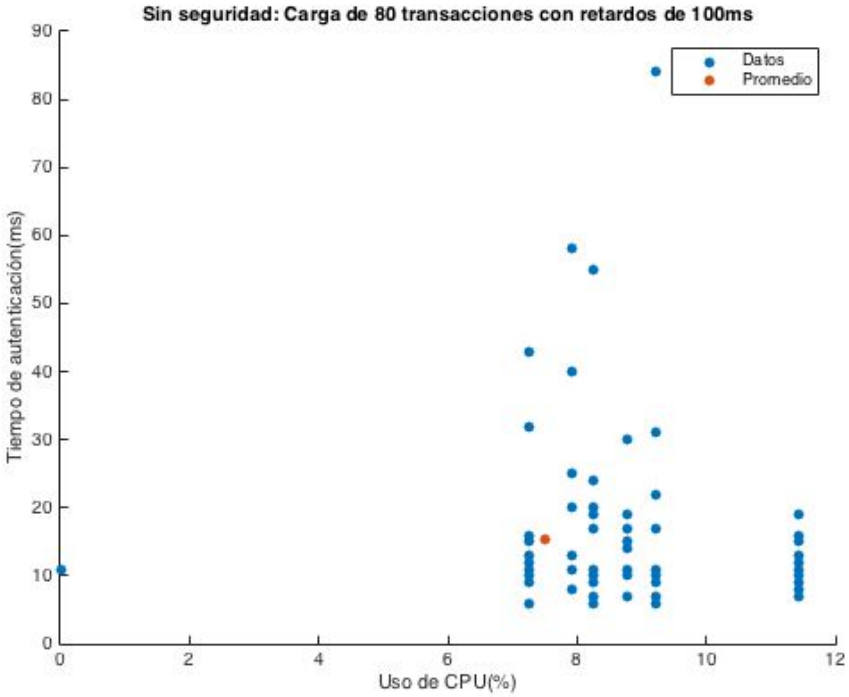




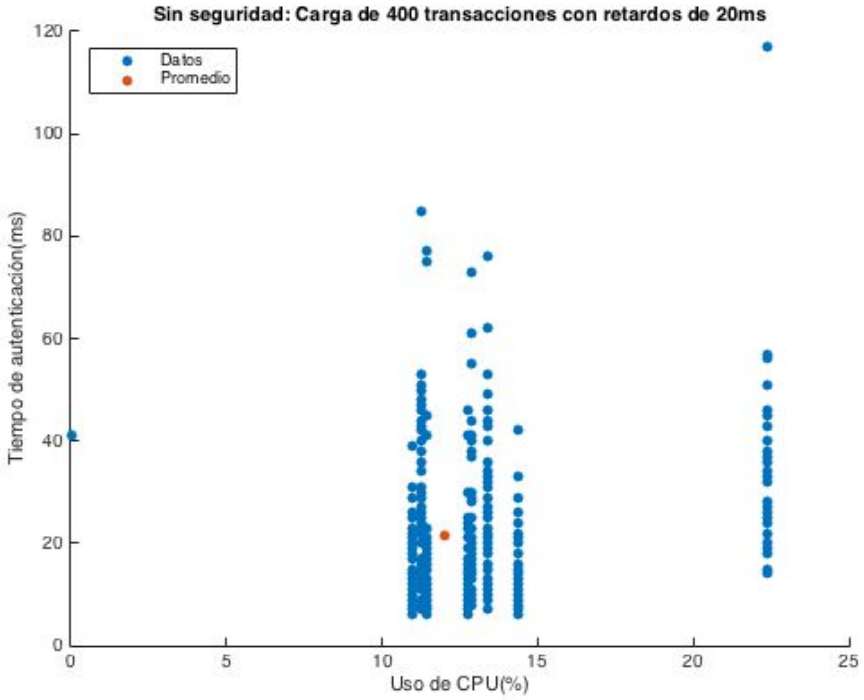
d. Se tienen en cuenta las mismas consideraciones que se mencionaron en el literal (d) del punto 3.

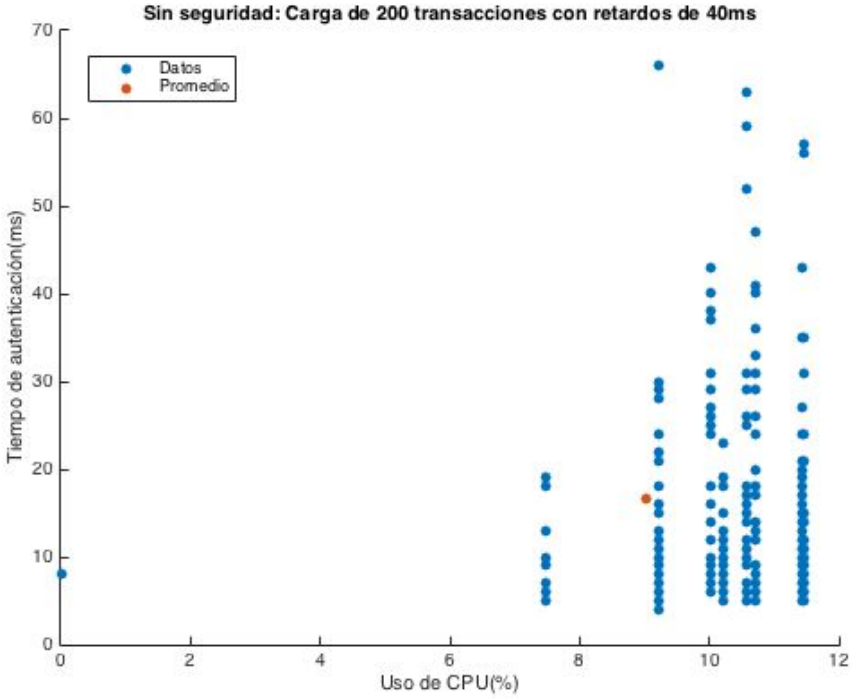


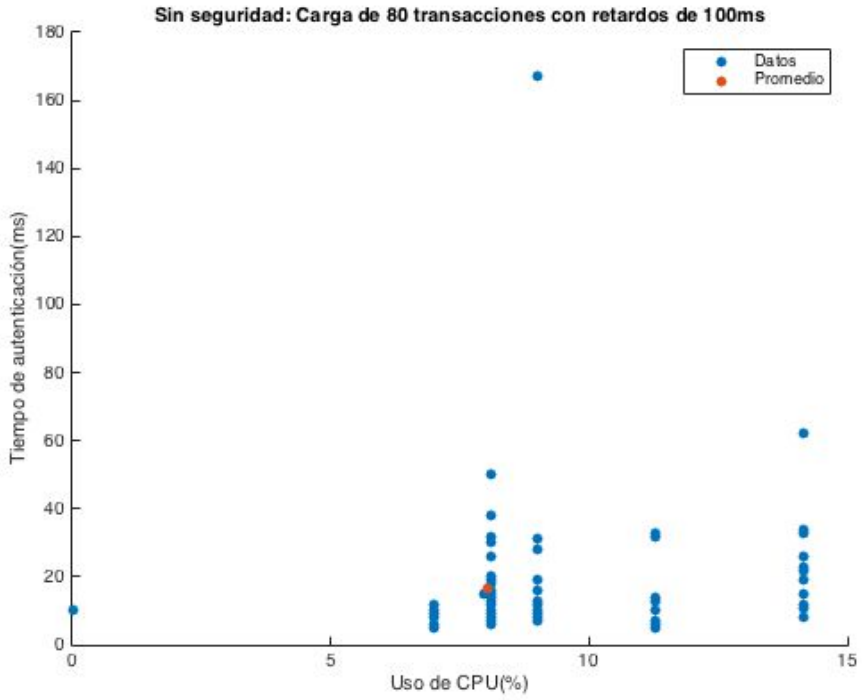

Sin seguridad: Carga de 400 transacciones con retardos de 20ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
10
20
30
40
50
60
70
80
0
5
10
15
20
25


Sin seguridad: Carga de 80 transacciones con retardos de 100ms
Datos
Promedio
Tiempo de autenticación(ms)
Uso de CPU(%)
0
10
20
30
40
50
60
70
80
90
0
2
4
6
8
10
12

e. En este caso, a diferencia del mismo literal del punto anterior (3), los datos se ven más centralizados.







4.2 Resultado esperado

Se espera que al correr el mismo proceso sin seguridad, los tiempos de respuesta se vean reducidos considerablemente. Al incluir seguridad, la complejidad de la consulta aumenta considerablemente debido a todos los algoritmos de cifrado/descifrado que se deben implementar, por lo cual los tiempos obtenidos en el cliente/servidor con seguridad deberían ser mayores que los tiempos obtenidos al correr las contrapartes sin seguridad. Más específicamente, los tiempos de autenticación deberían ser mucho menores sin seguridad que con seguridad, los tiempos de consulta menores, pero no tanto como los de autenticación y también se espera que haya menos fallas. Finalmente el uso de CPU debería ser menor en el caso sin seguridad puesto que el servidor solo necesita responder rápidamente las consultas sin tener que correr algoritmos de cifrado y descifrado.

4.3 Resultado obtenido

Estas gráficas confirman en su mayoría los resultados esperados. Las fallas no llevan a conclusiones debido a que no se presentaron fallas en la gran mayoría de las pruebas, pero las demás métricas sí se pueden comparar.

- *Tiempos*: Observando el literal A de los numerales 3 y 4 es posible comparar los tiempos de autenticación obtenidos al correr el sistema con y sin seguridad. Aunque en ambos casos los datos fluctúan bastante, es posible observar que los valores cuando no

se tiene seguridad por lo general se mantienen por debajo de los 20ms, mientras que cuando sí se tiene seguridad estos valores se mantendrán alrededor de los 40ms (ambos datos para la tabla de 80 transacciones separadas por 100ms). Adicionalmente, el mayor pico cuando se tiene seguridad alcanza los 600ms mientras que el mayor pico sin seguridad no supera los 170. No obstante, puede que estos picos alteren los resultados, pero si se ignoraran estos valores y se sacara una media, es evidente que los tiempos sin seguridad serán menores a los tiempos cuando sí se tiene seguridad, aunque se esperaba una diferencia más grande entre los dos, ya que en algunas gráficas la diferencia es difícil de identificar.

- *CPU*: Si se observa el literal C de los numerales 3 y 4 puede fácilmente compararse el uso de CPU cuando se tiene seguridad y cuando no se tiene. Si se comparan las gráficas de 400 transacciones con 20ms de retraso se puede observar claramente que cuando se tienen 8 threads y seguridad el uso de CPU fluctúa entre el 70 y el 90%, lo cual es bastante alto, mientras que cuando no se tiene seguridad, los mismos 8 threads para la misma carga de transacciones toma su valor máximo en 23%. Lo mismo se cumple para las diferentes cantidades de threads y las diferentes configuraciones de carga, por lo cual es seguro asumir que efectivamente se consume más CPU al tener seguridad.