

Experimento 2

Arquitectura de software

**Integrantes**

Karen Lorena Osorio Henao 201126600

Johan Velazques

Juan Sebastian Arciniegas 201325828

Santiago Robayo 201216389

**Índice**

1. **Pre-Experimentacion y post-experimentacionResultados**
2. **Implementacion balanceo de cargas**
3. **Comparación de resultados**
4. **Conclusiones**

**Definición del tipo de balanceador de carga**

Para nuestro balanceador de carga, se definió un Least-connection. Ya que este tipo de balanceador de carga optimiza mejor los recursos y puede a llegar a trabajar como el Round-Robin si todos los servidores reciben en la misma proporción las conexiones. Adicional, puede ofrecer mejor una respuesta si un servidor llega a fallar lo cual es muy importante en nuestro contexto. Gracias a la alta demanda de disponibilidad, por su necesidad de respuesta frente a los diferentes sucesos de advertencia o emergencia, se requiere servicio altamente disponible y veloz.

**Pre-experimentación**

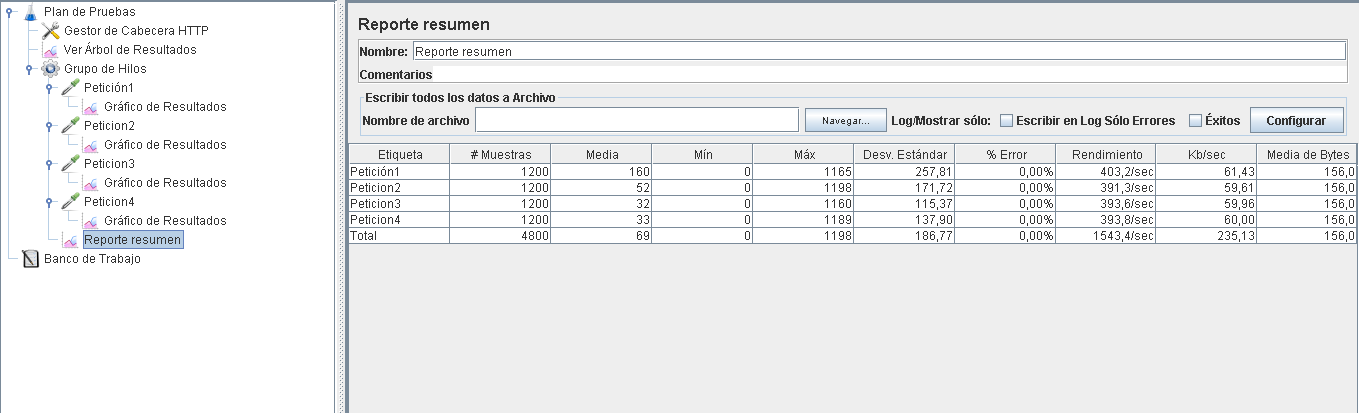
Para verificar el proyecto y el cumplimiento de los requerimientos no funcionales propuestos, ahora con la adición del requerimiento no funcional de disponibilidad, se realizaron varias pruebas en las que midieron los tiempos de respuesta al someter el sistema a diferentes cargas de peticiones. La problemática consiste en poner a prueba el funcionamiento del sistema OIL COL en cuanto al procesamiento y monitoreo de campos, pozos y zonas geograficas. El objetivo de los experimentos a realizar es revisar el comportamiento del sistema frente a los escenarios de calidad 001, 002 y 003 implementado diferentes servidores para garantizar la disponibilidad. Para realizar los experimentos, se realizarán varias iteraciones consumiendo los servicios de la aplicación que tienen que ver con los escenarios de calidad incrementando gradualmente el número de threads y registrando el tiempo de respuesta medio (ms), % de error y rendimiento (threads/sec). A partir de los datos y los resultados obtenidos, se concluirá acerca del cumplimiento de los escenarios de calidad teniendo en cuenta las medidas de respuesta de los mismos. Para llevar a cabo las pruebas, fue necesaria la creación de 4000 sensores que posteriormente se utilizarán para la actualización de los mismos y la verificación del escenario de calidad 003. Además, para la creación y actualización de cada uno de ellos se usó un script en el JSon del request para modificar los datos de dicho JSon cada vez que se realiza una petición. Adicionalmente, para garantizar la disponibilidad y simular la utilización de cinco servidores que procesan solicitudes en el sistema, fue necesaria la creación de cuatro proyectos más en Heroku en donde se clonó el proyecto original. Esto se hizo debido a que al utilizar la cuenta gratuita de Heroku, este proveedor de cloud provee un solo dyno (servidor). Al tener cinco proyectos estaríamos utilizando cinco dynos, lo que se traduce en cinco servidores. Finalmente, para agregar las funcionalidades de balanceo de carga, se utilizó Nginx. Para incluir los cinco dynos proveídos por Heroku, fue necesario agregar las direcciones IP de cada uno de estos en el archivo de configuraciones de Nginx. De esta manera, se distribuyen las peticiones hechas al sistema a cada uno de los servidores de Heroku. Finalmente, en el archivo de configuraciones de Nginx, se utilizó el método de balanceo de carga Round Robin. A partir de este, se manda la misma cantidad de peticiones a cada servidor sin tener en cuenta que tan saturado este cada uno de ellos. Consideramos que por medio de este método de balanceo de carga se consiguen mejores resultados en cuando a tiempos de respuesta y rendimiento porque no se tiene que consultar el número de conexiones activas a cada servidor cada vez que se haga una petición. Esto se constituye como un beneficio frente a Least Conn debido a que este método si hace verificación del número de conexiones que tiene cada servidor y envía la solicitud al servidor menos saturado.

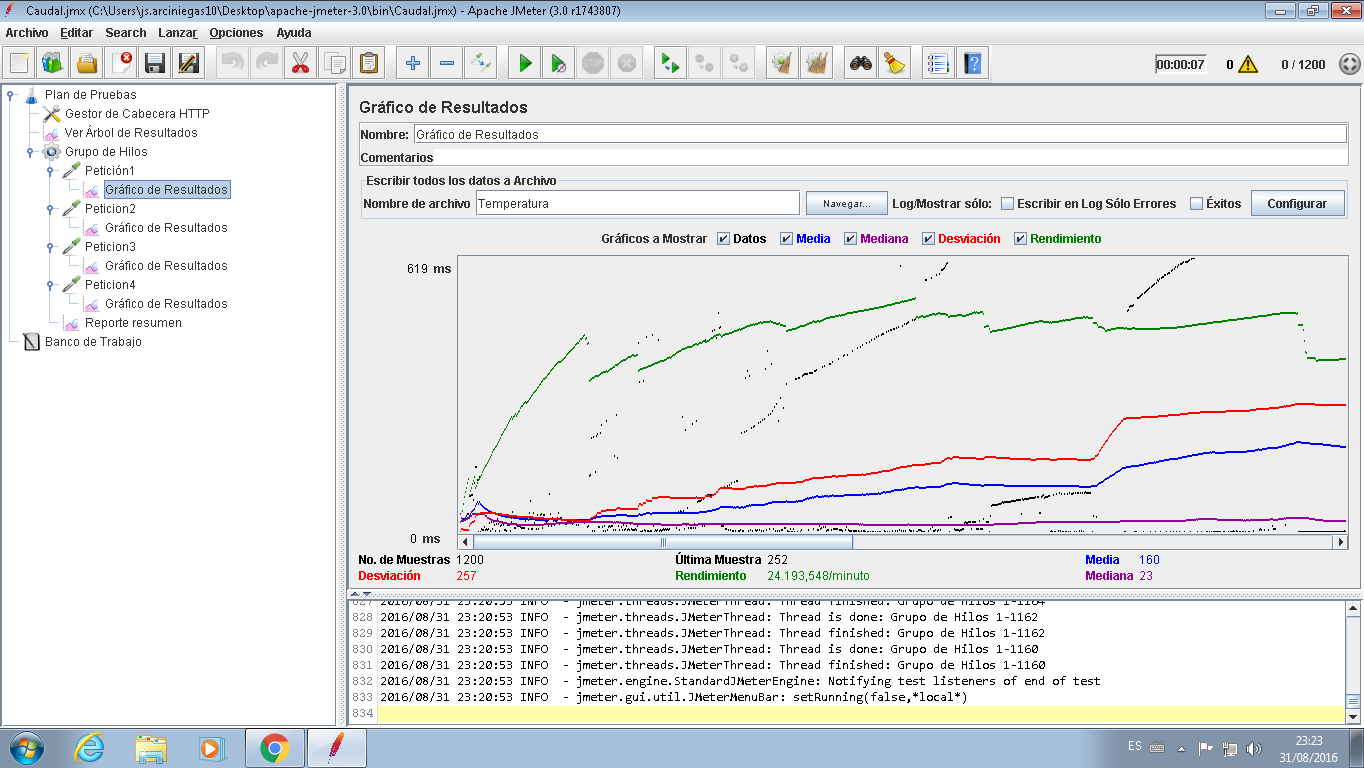
Para realizar los experimentos se utilizó la herramienta JMeter, el cual es un software en el que se pueden hacer pruebas variando la cantidad de threads, el tiempo de ramp-up, y algunos otros parámetros. La característica más importante de esta herramienta es que presenta la posibilidad de utilizar scripts adentro del request JSon de manera que se varíen automáticamente algunos de los parámetros del mismo, presentando más posibilidades de experimentación.

Debido a que la base de datos y el servidor están en la misma ubicación geográfica se espera que las peticiones, ya sea de POST evento sísmico y PUT sensores, presenten poca latencia. Además, gracias al script mencionado anteriormente, se logra actualizar un sensor diferente por thread, logrando que no se encolen peticiones por recursos ocupados.

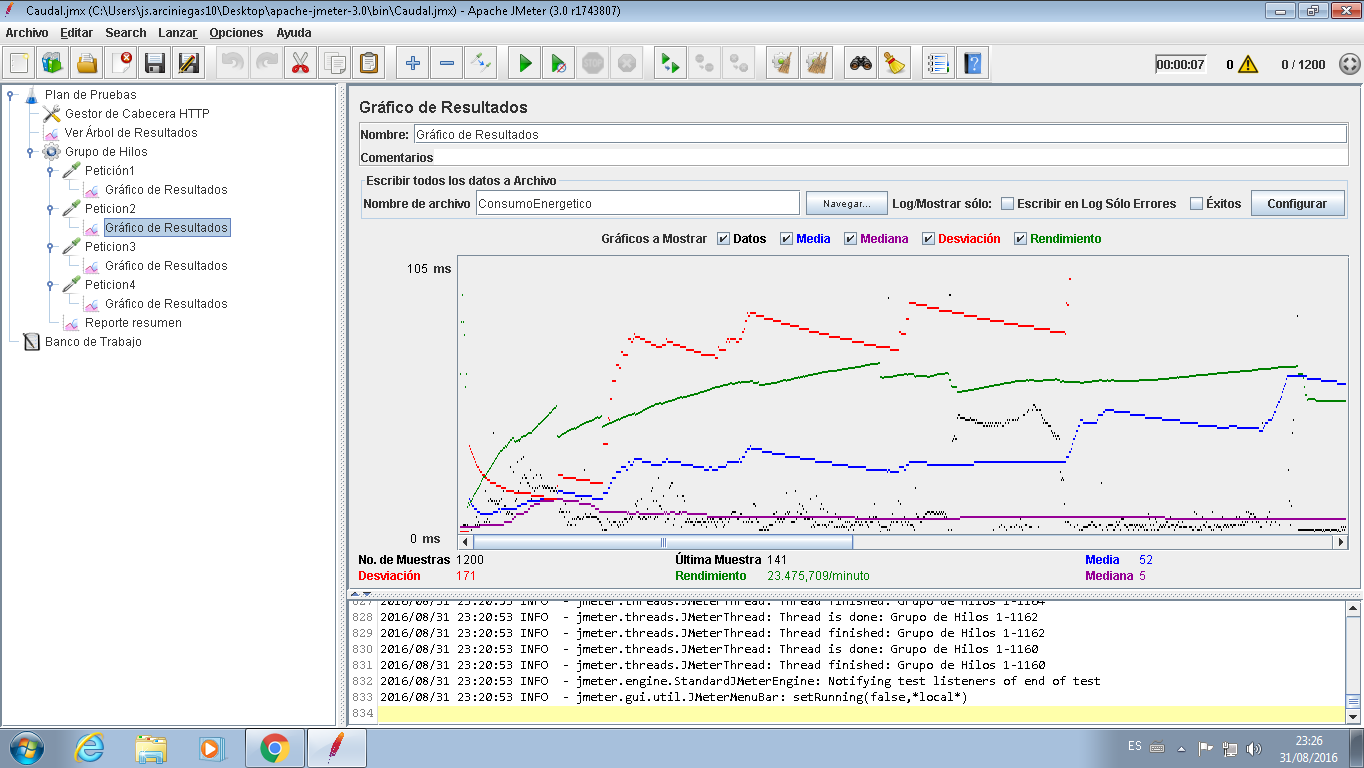
**Resultado Pruebas Experimento 1:**

A continuación, se muestran algunos resultados obtenidos para estas pruebas y sus respectivos análisis:

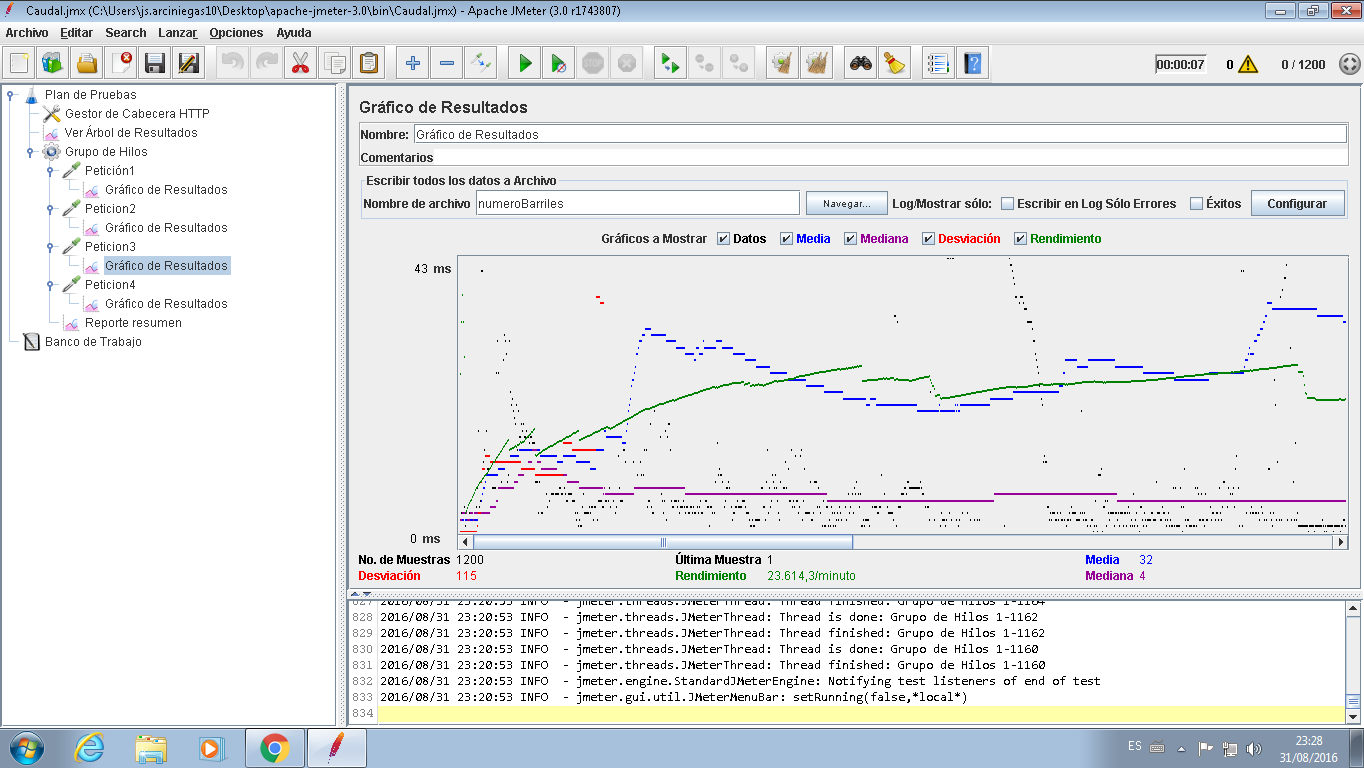


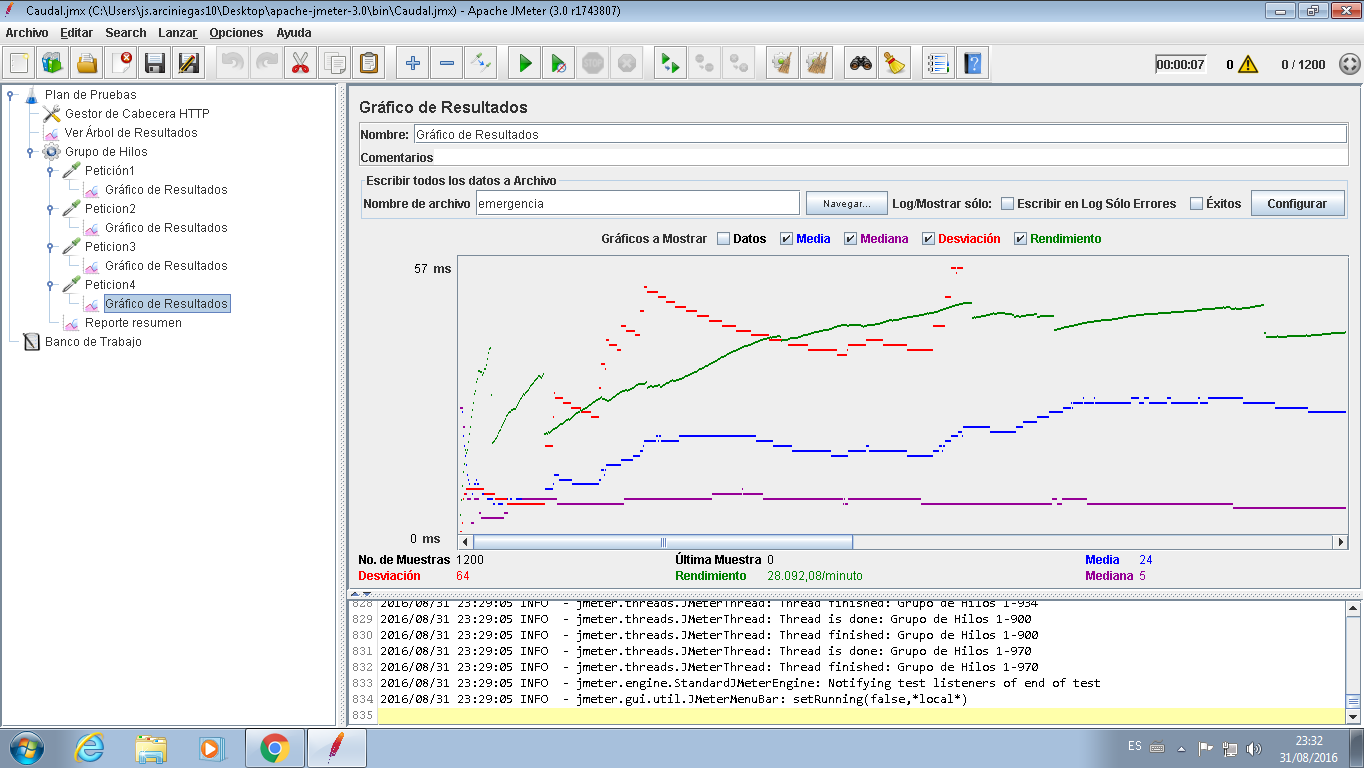


En la primera gráfica se puede apreciar que en la prueba de latencia para que generar un reporte de temperatura, con 1200, se obtiene una media de 160 ms de respuesta por cada ThreadGroup. Lo que garantiza que latencia en la generación de un reporte es baja. Así mismo, la segunda gráfica lo constata lo mismo con el consumo energético.



Se observa en las gráficas que tenemos un rendimiento de 23-28 threads por minuto.

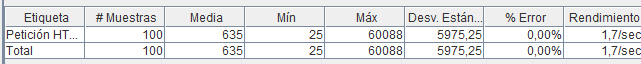




**Resultado Pruebas Experimento 2:**

**GET: 1 Nodo**

100:



200:



400:



800:



1600:



3200:

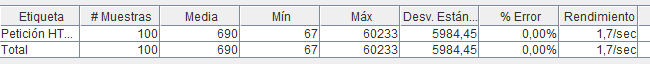


4000:

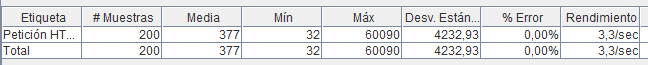


**POST: 1 Nodo**

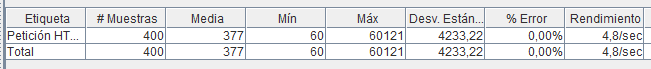
100:



200:



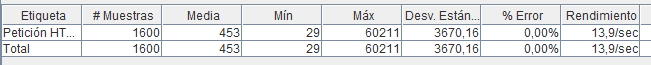
400:



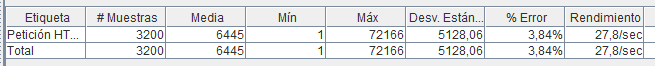
800:



1600:



3200:



4000:



**GET REVIEW:**

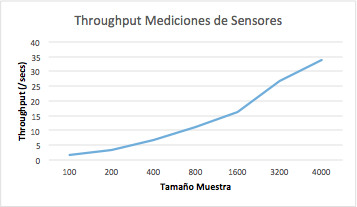


**POST REVIEW:**

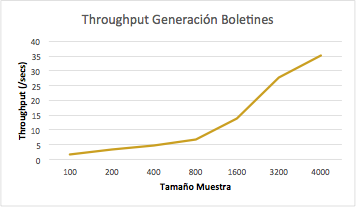


**GRAFICOS:**

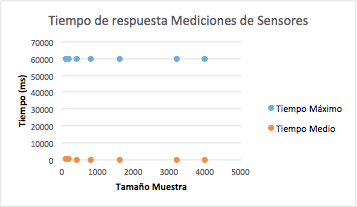
Throughput Sensores GET



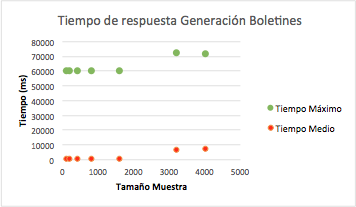
Throughput Sensores POST



Tiempo de respuesta sensores GET



Tiempo de respuesta sensores POST



**GET: 2 Nodos**

100:



200:



400:



800:



1600:



3200:



4000:



**POST: 2 Nodos**

100:



200:



400:



800:



1600:



3200:

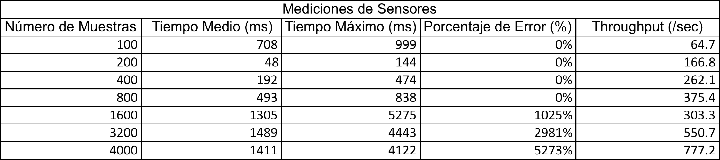


4000:



**REVIEW**

**Mediciones de Sensores GET:**

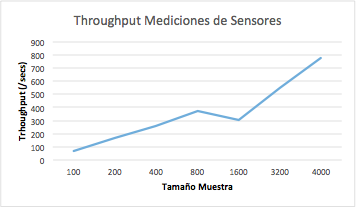


**Mediciones de Sensores POST :**

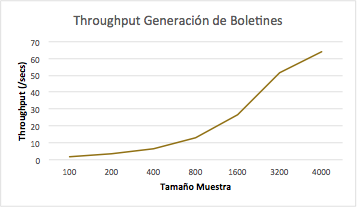


**Graficas**

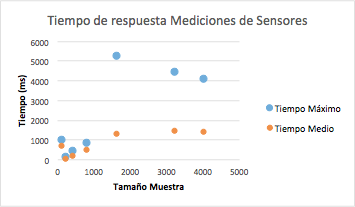
Throughput Sensores Get



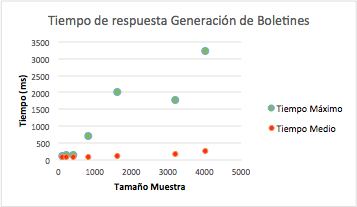
Throughput POST



Tiempo de respuesta sensores GET



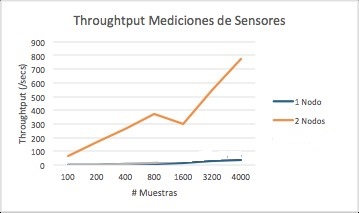
Tiempo de respuesta POST

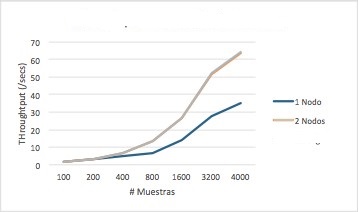


**Comparación de resultados**

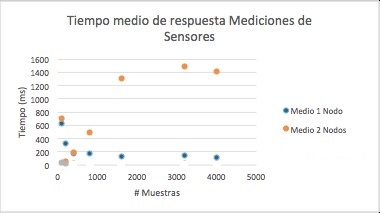
En esta sección se mostrarán en contraste los resultados de ejecutar las pruebas sin la implementación de seguridad, y con implementación de seguridad, utilizando 1 y 2 nodos para el balanceo de las transacciones.

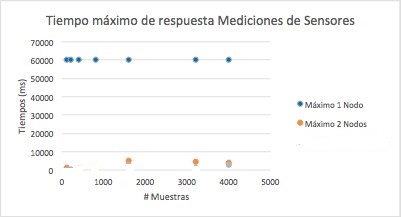
Throughput:





Tiempos de respuesta:





**CONSLUSIONES**

Si se comparan los tiempos de respuesta de la aplicación sin disponibilidad y balanceo de un solo nodo, se puede apreciar la gigantesca diferencia de tiempos, especialmente en los tiempos máximos de respuesta. Esto se puede deducir de comparar el comportamiento de la aplicación con 1 y 2 nodos, en cuanto a la medida de tiempo de respuesta. Se puede apreciar que la aplicación, utilizando 2 nodos para balancear la carga, se comporta casi igual que la aplicació El único contexto en el que no se ve favorecido el tiempo de respuesta para el caso del uso de 2 nodos es para el tiempo de respuesta medio. Esto puede deberse al tiempo necesario que se dispone para la distribución de las transacciones para poder balancear la carga, y esto sólo sucede en el proceso de recibir mediciones de los sensores. En este caso en particular es donde más se puede apreciar el efecto de la implementación de seguridad en la aplicación, y el hecho evidente de que esta tiene un costo.

En cuanto al balanceo de carga en sí, si bien no se tienen datos históricos para dar una medida de disponibilidad verificable, el efecto positivo que éste tiene es evidente en la figura de Throughput para el registro de mediciones de sensores. Como se puede ver, para 2 nodos activos, la cantidad de peticiones efectivas procesadas por unidad de tiempo supera en gran medida al caso de 1 solo nodo, lo cual significa que la capacidad del sistema se escala de tal forma que es capáz de atender la carga de forma creciente con respecto a esta última.