“Sistema de Seguimiento Cardiaco”

Juan Sebastián Barragán Jerónimo, Sebastián Sánchez Galiano, Santiago Rojas Herrera

Experimento 1 – Entrega Parcial 1

Bogotá, Colombia

{s.rojas19, js.barragan796, s.sanchez59}@uniandes.edu.co

Fecha de presentación: 15 de febrero de 2017

1 Experimentación 2

1.1 Recepción de Información de Sensores 2

1.1.1 Entrega Parcial 2

1.1.2 Entrega Definitiva 3

1.2 Recepción de Emergencia 4

2 Arquitectura 4

### 

# Experimentación

## Recepción de Información de Sensores

Se realizaron unas pruebas de carga sobre la aplicación. En estas pruebas se comprobó que el sistema soporta la recepción de información desde los 3.000 sensores (3 sensores x 1.000 pacientes potenciales) en una ventana de tiempo de un segundo. Para la configuración de las pruebas se utilizaron solo 1.000 threads considerando que una decisión de arquitectura fue utilizar una solo solicitud para enviar la información de los tres atributos: frecuencia cardiaca, presión sanguínea y nivel de estrés. Está decisión es explicada en la sección de arquitectura de este documento.

### Entrega Parcial



Figure 1. Tiempo de Respuesta Medio para Crear una Medición (Entrega Parcial)

En la gráfica anterior se puede comparar el tiempo de respuesta medio dependiendo del número de threads lanzados. Se puede ver que para números de threads menores e iguales a 1.000 el tiempo de respuesta medio es menor a 1 segundo. También se puede ver que el tiempo de respuesta medio aumenta drásticamente cuando hay más de 600 threads. El tiempo de respuesta medio para 1.000 threads es 700ms.

A continuación, se muestra el pantallazo de JMeter para la prueba realizada con 1.000 threads.

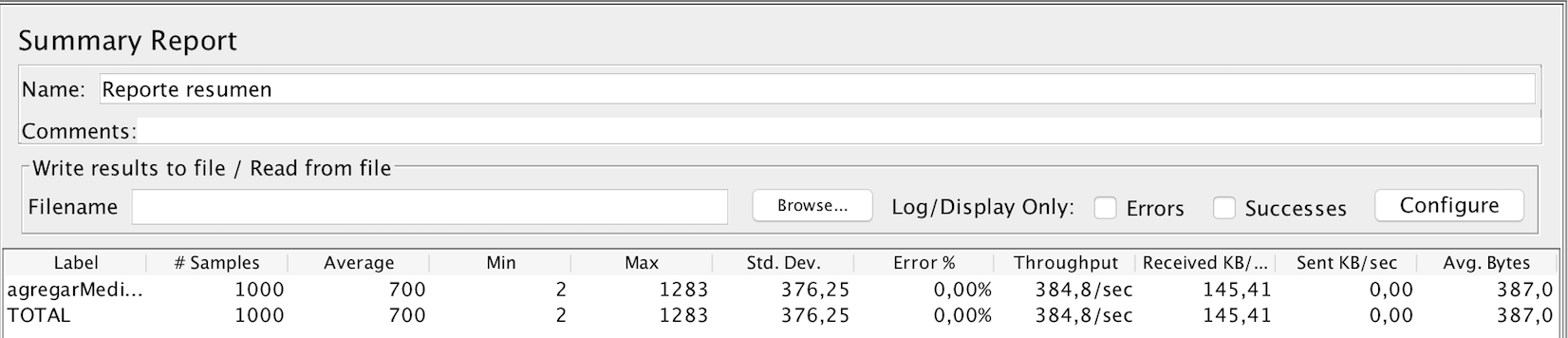


Figure 2. Reporte Resumen de JMeter (Entrega Parcial)

### Entrega Definitiva



Figure 3. Tiempo de Respuesta Medio para Crear una Medición (Entrega Definitiva)

En la gráfica anterior se puede comparar el tiempo de respuesta medio dependiendo del número de threads lanzados. Se puede ver que para números de threads menores e iguales a 1.000 el tiempo de respuesta medio es menor a 1 segundo. Se puede ver que a medida que aumenta el número de threads aumenta el tiempo de respuesta medio. El tiempo de respuesta medio para 1.000 threads es 662 ms.

A continuación, se muestra el pantallazo de JMeter para la prueba realizada con 1.000 threads.

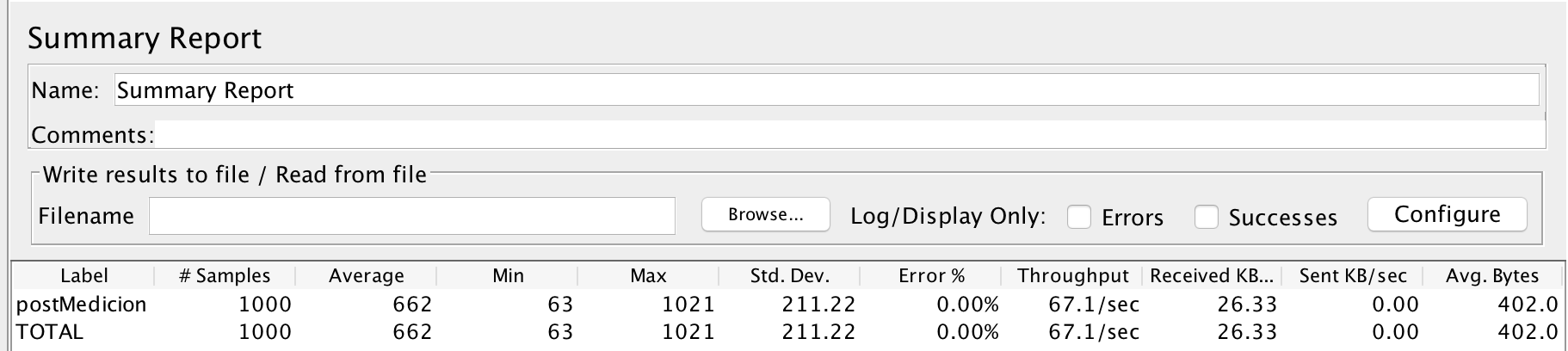
****

Figure 4. Reporte Resumen de JMeter (Entrega Definitiva)

Se puede hacer una comparación de los resultados obtenidos en la entrega parcial y en la entrega definitiva. Se puede notar que para la entrega definitiva el tiempo de respuesta medio aumento de manera más rápida que en la entrega parcial. Esto puede deberse a que para la entrega parcial las pruebas se realizaron en un computador con procesador i7 con cuatro nucleos mientras que para la entrega definitiva se hicieron en un computador con procesador Intel Core i5 con dos nucleos.

## Recepción de Emergencia

Se realizaron unas pruebas de carga sobre la aplicación. En estas pruebas se comprobó que el sistema soporta la recepción de 1.000 emergencias en una ventana de tiempo de un segundo. Para la configuración de las pruebas se utilizaron solo 1.000 threads.

# Arquitectura

Para favorecer el cumplimiento de los escenarios de calidad se tomaron unas decisiones en cuanto a la arquitectura a implementar.

En primer lugar, se tomó la decisión de implementar una arquitectura por capas, más específicamente se eligió implementar una arquitectura JAX-RS. Implementar una arquitectura por capas permite que haya bajo acoplamiento y alta cohesión. El hecho de que la solución tenga estas características permite que el software tenga robustez, confiabilidad, reusabilidad y facilita su entendimiento. La justificación para utilizar una arquitectura de capas es que se realizó un experimento en el que se comparó la arquitectura JAX-RS con la arquitectura Play y se encontró que JAX-RS es mejor que Play en lo que se refiere al atributo de calidad de escalabilidad. JAX-RS pudo manejar mejor el crecimiento continuo de trabajo de manera fluida y pudo hacerse más grande sin perder calidad en los servicios ofrecidos. La elección de una arquitectura por capaz es justificada teniendo en cuenta que se quiere ofrecer una solución de software que atienda a un número de usuarios considerable.

En segundo lugar, se tomó la decisión de implementar una solución mediante servicios web RESTful. Implementar esta arquitectura permite cumplir con los escenarios de calidad para el atributo de desempeño. Esto se debe a que hay un protocolo cliente/servidor sin estado y como resultado ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Esto se traduce en un mejoramiento en las métricas de latencia y un bajo uso de los recursos computacionales. Otra decisión que se tomó fue implementar beans sin estado. La justificación para hacer esto es similar a la anterior y se traduce a que la aplicación tiene un mejor desempeño.

Otra decisión que se tomó está relacionada a la manera en que el sistema recibe la información de los sensores de frecuencia cardiaca, presión sanguínea y nivel de estrés. Se eligió utilizar solo una petición para mandar la información de estos tres atributos. Si se mandará la información de cada uno esto supondría un aumento en el overhead computacional lo que no favorecería el desempeño de la aplicación.