ISIS2503 ARQUITECTURA Y DISEÑO DE SOFTWARE

Joan David Torres Pinzon
Luis Andres Mesa
Juan Pablo Gonzalez
Felipe Martinez

# Pre-experimentación

## Problemática

OilCol S.A desea adquirir un software para el procesamiento y monitoreo automatizado de datos sobre sus pozos, campos y regiones de explotación de petróleo. Mediante la experimentación se explorarán las diferentes opciones de arquitecturas que satisfagan los atributos de calidad deseados, específicamente para este experimento se trabajara la arquitectura basada en Play-Framework.

## Decisiones de Arquitectura:

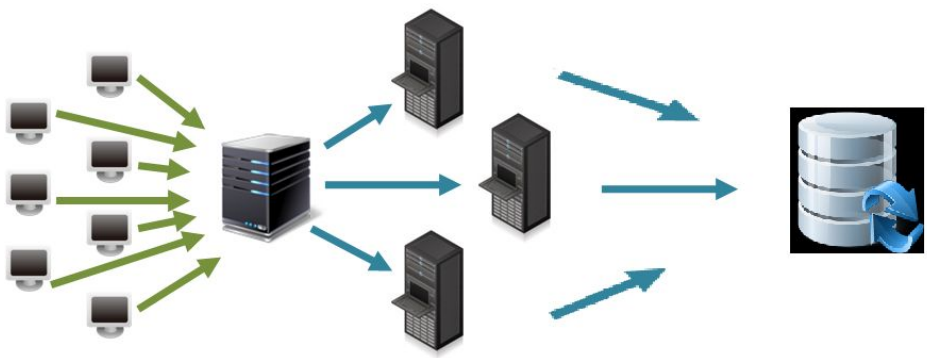
A lo largo de este experimento se centró el trabajo en cumplir con los atributos de disponibilidad, seguridad y desempeño. Con el fin de dar solución a los requerimientos de disponibilidad se implementó un balanceador de carga en Nginx, se definieron 3 servidores para responder a las solicitudes que se generan y uno de backup. Con respecto al método para distribuir las solicitudes se implementó una técnica de Least Conn, ya que, reduce la sobrecarga de los servidores y por consiguiente permite un mejor uso de los recursos. Por otro lado, se decidió configurar un peso, que indica un mayor número de solicitudes recibidas, sobre el servidor alojado en el mismo lugar de la base de datos, ya que por sus recursos físicos (mas RAM) y su menor tiempo de transmisión, hacia y desde la base de datos, es capaz de resolver las peticiones en menor tiempo. Finalmente, se implementó un mecanismo de Health Check que identifica cuando uno de los servidores está saturado y fija un tiempo para que esté disponible antes de recibir nuevas solicitudes. A continuación se muestra la configuración mencionada:

```
upstream oilcol {
    least_conn;
      server 172.24.42.138:9000 backup;
      server 172.24.42.139:9000 weight=2;
      server 172.24.42.111:9000 max_fails=1 fail_timeout=5s;
      server 172.24.42.107:9000 max_fails=1 fail_timeout=5s;
}
    server {
        listen       80;
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            proxy_pass http://oilcol;
        }
```

Con respecto al atributo de desempeño, al implementar el balanceador de carga con la configuración anterior, se realiza un mejor manejo de los recursos y por consiguiente se aumenta el desempeño de la aplicación.

Finalmente, para el atributo de seguridad, se realizaron avances en mecanismos de autenticación implementados en la librería de Play Secure. En esta entrega se tiene un solo usuario el cual tiene que autenticarse para poder acceder a los servicios que tienen la aplicación.

**Esquema Disponibilidad de la Aplicación**



La imagen anterior muestra el esquema utilizado para las pruebas y cambios arquitecturales que responden al requerimiento de disponibilidad y desempeño. En este se puede observar varias peticiones que entran a un balanceador de carga el cual reparte las mismas en los diferentes servidores desplegados en nuestro caso en las máquinas virtuales. Finalmente estas se comunican con la base de datos y guardan o hacen los cambios pertinentes sobre la información que maneja.

## Experimentación

Se realizarán para el atributo de calidad Desempeño los siguientes escenarios.

| Identificador | Tipo | Prioridad |
|---|---|---|
| 1 | Desempeño | Alta |
| Fuente | | |
| Sensores | | |
| Estímulo | | |
| Carga de datos sobre el sistema | | |
| Ambiente | | |
| Normal | | |
| Medida Esperada | | |
| Se recibe información desde los 4800 sensores en una ventana de tiempo de 1 segundo | | |

| Identificador | Tipo | Prioridad |
|---|---|---|
| 2 | Desempeño | Alta |
| Fuente | | |
| Sensores | | |
| Estímulo | | |
| Carga de datos sobre el sistema | | |
| Ambiente | | |
| Normal | | |

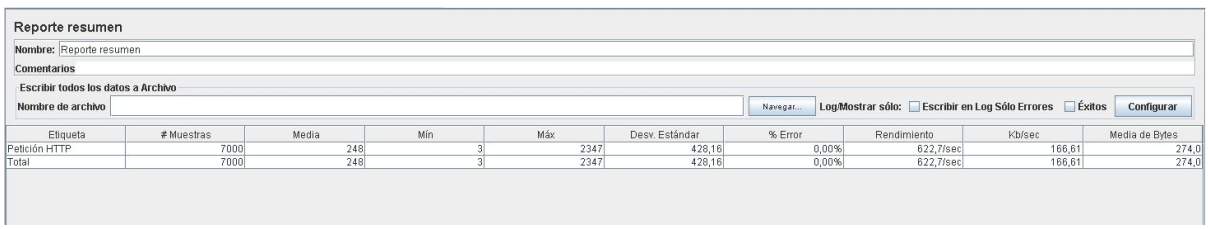
| Medida Esperada |
| --- |
| El sistema recibe la información desde 1600 sensores en una ventana de 1 segundo. |

| Identificador | Tipo | Prioridad |
| --- | --- | --- |
| 3 | Disponibilidad | Alta |
| Fuente | | |
| Sensores | | |
| Estímulo | | |
| Carga de datos sobre el sistema | | |
| Ambiente | | |
| Anormal | | |
| Medida Esperada | | |
| El sistema es capaz de resolver las solicitudes cuando uno de los servidores se ha caído. | | |

**Resultados de pruebas:**

Con el fin de probar la eficiencia de la nueva configuración, se realizaron las pruebas para los métodos de Post y Get, con un ramp up de 1 segundo y variando el número de peticiones. La nueva configuración permite obtener grandes mejoras frente a los experimentos pasados.

Post:




| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Petición HTTP | 7000 | 248 | 3 | 2347 | 428,16 | 0,00% | 622,7/sec | 166,61 | 274,0 |
| Total | 7000 | 248 | 3 | 2347 | 428,16 | 0,00% | 622,7/sec | 166,61 | 274,0 |

Get:

| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Petición HTTP | 3000 | 174 | 6 | 910 | 185,63 | 0,00% | 684,3/sec | 165,06 | 247,0 |
| Total | 3000 | 174 | 6 | 910 | 185,63 | 0,00% | 684,3/sec | 165,06 | 247,0 |

Se logró obtener menor latencia y 0% de error, lo cual satisface los requerimientos de desempeño establecidos al principio del desarrollo. Se alcanzaron 10000 peticiones Post y 7000 peticiones GET antes de obtener una latencia y porcentaje de error mayor al permitido.

Post:

Reporte resumen

Nombre: Reporte resumen

Comentarios

Escribir todos los datos a Archivo

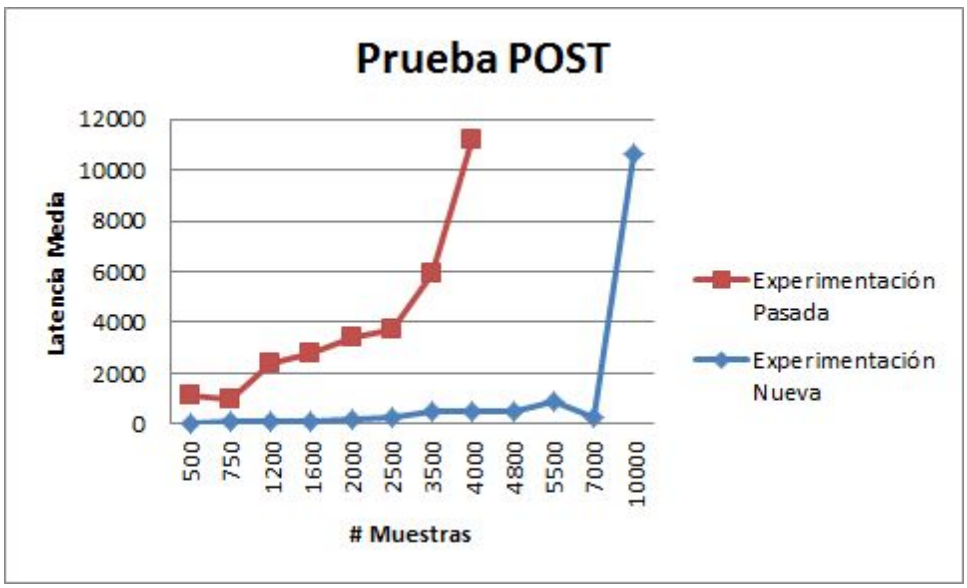
Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

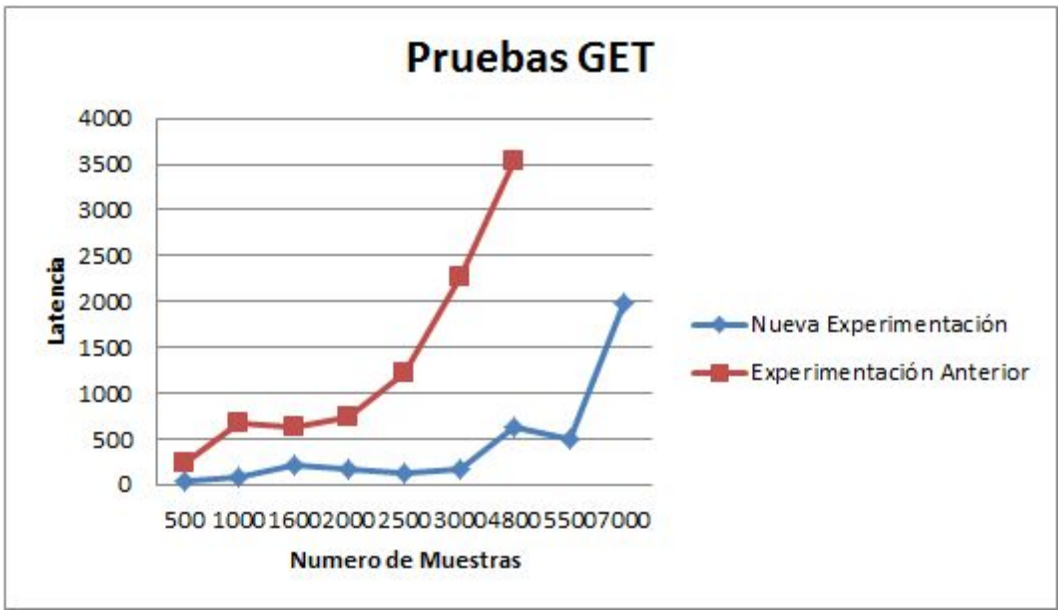
| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
|---|---|---|---|---|---|---|---|---|---|
| Petición HTTP | 10000 | 10661 | 4 | 23933 | 7479,14 | 1,15% | 368,7/sec | 105,45 | 292,8 |
| Total | 10000 | 10661 | 4 | 23933 | 7479,14 | 1,15% | 368,7/sec | 105,45 | 292,8 |

Get:

| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
|---|---|---|---|---|---|---|---|---|---|
| Petición HTTP | 7000 | 1983 | 2 | 6873 | 1423,87 | 14,80% | 589,9/sec | 282,02 | 489,6 |
| Total | 7000 | 1983 | 2 | 6873 | 1423,87 | 14,80% | 589,9/sec | 282,02 | 489,6 |

Para visualizar de mejor manera los avances alcanzados en esta nueva etapa del desarrollo, se graficaron los resultados de las pruebas actuales contra los resultados obtenidos en experimentos anteriores. Es evidente que se presentó una gran mejoría con respecto al atributo de desempeño.



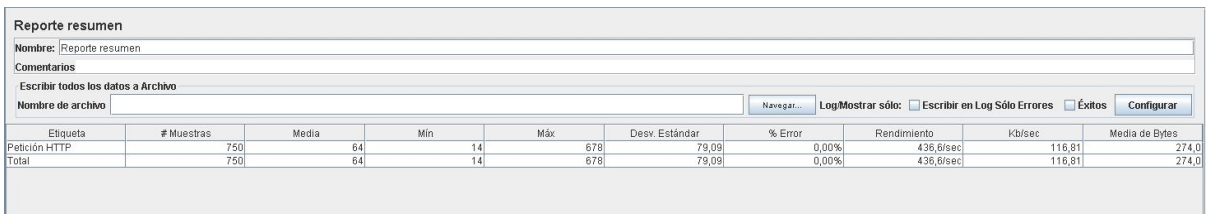



Los resultados obtenidos en las gráficas corroboran las hipótesis planteadas antes de iniciar la experimentación y muestran que al implementar el balanceador de carga se logró mejorar los resultados obtenidos.

**Disponibilidad:**

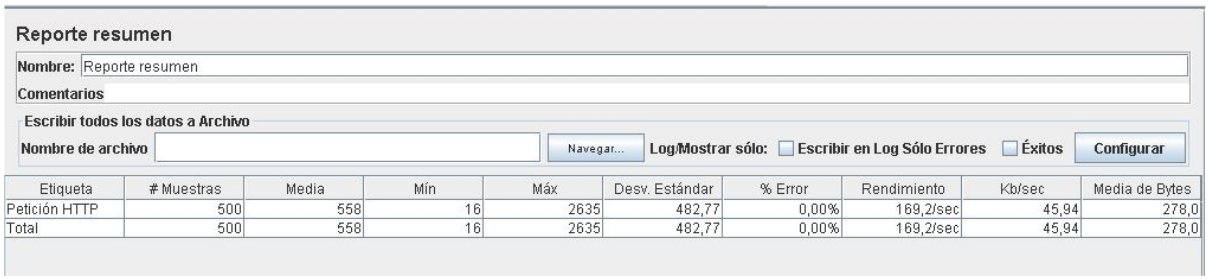
Para probar la disponibilidad se decidió realizar una serie de peticiones mediante Jmeter y durante la ejecución se desconecto uno de los servidores para analizar la reacción del balanceador de carga.

Resultados en estado normal:




| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
|---|---|---|---|---|---|---|---|---|---|
| Petición HTTP | 750 | 64 | 14 | 678 | 79,09 | 0,00% | 436,6/sec | 116,81 | 274,0 |
| Total | 750 | 64 | 14 | 678 | 79,09 | 0,00% | 436,6/sec | 116,81 | 274,0 |

Resultados desconectando servidor:




| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Media de Bytes |
|---|---|---|---|---|---|---|---|---|---|
| Petición HTTP | 500 | 558 | 16 | 2635 | 482,77 | 0,00% | 169,2/sec | 45,94 | 278,0 |
| Total | 500 | 558 | 16 | 2635 | 482,77 | 0,00% | 169,2/sec | 45,94 | 278,0 |

En estos resultados se puede apreciar que al desconectar el servidor, el balanceador reajusta los redireccionamientos, y logra completar las solicitudes. No obstante, se puede apreciar que el desempeño se ve afectado. Debe tenerse en cuenta que estas pruebas se repitieron más de una vez para poder así obtener una margen de error.

## Conclusiones

Después de analizar los resultados obtenidos a lo largo de la experimentación, se llegó a la conclusión que mediante la implementación del balanceador de carga se logró mejorar el cumplimiento del atributo de desempeño. Los tiempos que se obtuvieron con el balanceador reflejan un mejor manejo de recursos y por consiguiente una menor latencia. Por otro lado, se logró introducir un mecanismo para cumplir el requerimiento de disponibilidad, el cual fue un balanceador de carga implementado en Nginx. No obstante, se plantea la posibilidad de que se esté generando un cuello de botella al acceder a la base de datos, ya que se está utilizando una base de datos centralizada mediante Postgres. Asimismo, se evalúa la posibilidad de replicar los datos en distintas máquinas e implementar un mecanismo para sincronizarlos. Finalmente se comenzó a agregar la parte de seguridad donde se implementaron mecanismos de autenticación usando la librería de Play Secure. Esto es de gran importancia ya que ahora solo usuarios autenticados pueden hacer uso de los servicios que provee la aplicación. Esto nos permitió por otro lado, darnos cuenta que a pesar de que se implementó esta parte de seguridad, todavía nos hace falta la creación de permisos para diferentes grupos de usuario.

## Repositorio de la aplicación

*https://github.com/f-martinez11/Oilcol*