

hdiv

1.0

Rendimiento



1.	<u>INTRODUCCION</u>	3
2.	<u>CONFIGURACIÓN</u>	4
	2.1 SOFTWARE	4
	2.2 MÁQUINAS	4
	2.3 HDIV	4
3.	<u>RESULTADOS</u>	5
	3.1 TIEMPO MEDIO DE RESPUESTA	5
	3.1.1 TOMCAT 5.0.28	6
	3.1.2 WEBSHERE 6.0.1	6
	3.2 USO DE CPU.....	7
	3.2.1 TOMCAT 5.0.28	7
	3.2.2 WEBSHERE 6.0.1	7
	3.3 CONSUMO DE MEMORIA	8
	3.3.1 TOMCAT 5.0.28	8
	3.3.2 WEBSHERE 6.0.1	8
	3.4 ANÁLISIS DE LOS RESULTADOS	9
	3.4.1 ESTADO EN CLIENTE VS. ESTADO EN SERVIDOR	9
	3.4.1.1 ESTADO EN CLIENTE	9
	3.4.1.2 ESTADO EN SERVIDOR.....	10
4.	<u>CONCLUSIONES</u>	11
5.	<u>REFERENCIAS</u>	12

1. INTRODUCCION

Con el objeto de solucionar la mayoría de vulnerabilidades web se ha creado HDIV, añadiendo funcionalidades de seguridad a aplicaciones desarrolladas en Struts de forma transparente al programador y sin añadir mayor complejidad en el desarrollo de aplicaciones.

El objetivo de este artículo es evaluar el rendimiento de HDIV aplicado a diferentes aplicaciones web sobre diferentes servidores.

Las diferencias entre las distintas propuestas que ofrece HDIV implican la necesidad de estudiarlas y compararlas, posibilitando la elección de la mejor alternativa o estrategia para cada caso en particular. Para ello, este artículo hace un análisis de las tres estrategias posibles en HDIV: Memoria, Cifrado y Hash; comparando los tiempos con aplicaciones sin ninguna solución para la integridad de los datos.

Se han utilizado las mediciones de la línea base para calcular las mediciones superiores correspondientes a cada solución.

El documento finaliza mostrando las conclusiones que se han obtenido de este estudio.

2. CONFIGURACIÓN

Las pruebas se han llevado a cabo en un entorno de pruebas automatizado con el software y hardware que a continuación se detalla.

Se configuró el hardware y software en el entorno de pruebas para que la carga del servidor simulara un entorno real de producción.

2.1 Software

- JMeter 2.2 con J2SDK 1.4.2_04 y configuración por defecto de la JVM [1]
- Jprofiler, versión 4.2.1 [2]
- Servidores Web (configuración por defecto de la JVM):
 - Tomcat 5.0.28 y JRE de SUN, versión 1.4.2_04
 - Websphere 6.0.1 y JRE de Websphere v6

2.2 Máquinas

- Interfaz gráfica de JMeter sobre un Intel Pentium 4 CPU 3.2 GHz, 1 GB DDR
- Tomcat sobre un Intel Pentium 4 CPU 3.2 GHz, 1 GB DDR
- LAN 100 Mb/s

2.3 HDIV

Se han realizado pruebas de las tres estrategias disponibles en HDIV: Memoria, Cifrado y Hash.

3. RESULTADOS

Los resultados que a continuación se presentan han sido obtenidos tras haber repetido la configuración de cada prueba 1000 veces para garantizar que los resultados se repiten, y tras haber conseguido estabilizar el *tiempo medio de respuesta*, *uso de CPU* y *consumo de memoria* obteniendo así unos resultados fiables y con garantía.

Las mediciones obtenidas en el entorno sin ninguna solución de integridad de datos instalada se utilizaron como mediciones “de línea base” para calcular las mediciones de rendimiento superiores a ella. Las mediciones superiores de *tiempo de respuesta* y *uso de la CPU* se obtuvieron restando el valor “de línea base” de la medición de la solución correspondiente y expresando el resultado como un porcentaje de los resultados de línea base.

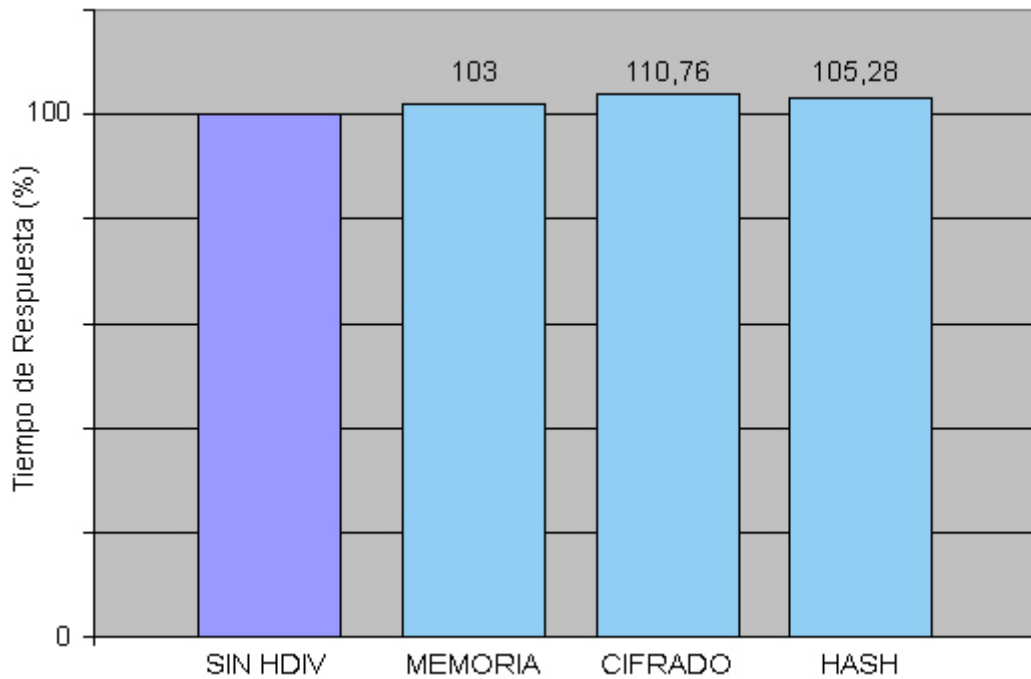
Medición del Rendimiento	Definición de la medición	Método de cálculo
Tiempo Medio Respuesta (%)	Se calculó el tiempo medio de respuesta en cada repetición. La media de las repeticiones representa la medición general del rendimiento.	Calculado automáticamente por <i>JMeter</i> .
Uso de la CPU (%)	Uso del procesador (como porcentaje de la potencia de procesamiento total de la CPU) en intervalos de 5 segundos. Se calculó la media de estos datos para crear el valor en cada repetición. La media de las repeticiones representa la medición general del rendimiento.	Recogido automáticamente por <i>JProfiler</i> .
Consumo de Memoria (MB)	Se calculó el uso de memoria en cada repetición. La media de las repeticiones representa la medición general del rendimiento.	Recogido automáticamente por <i>JProfiler</i> .

3.1 Tiempo Medio de Respuesta

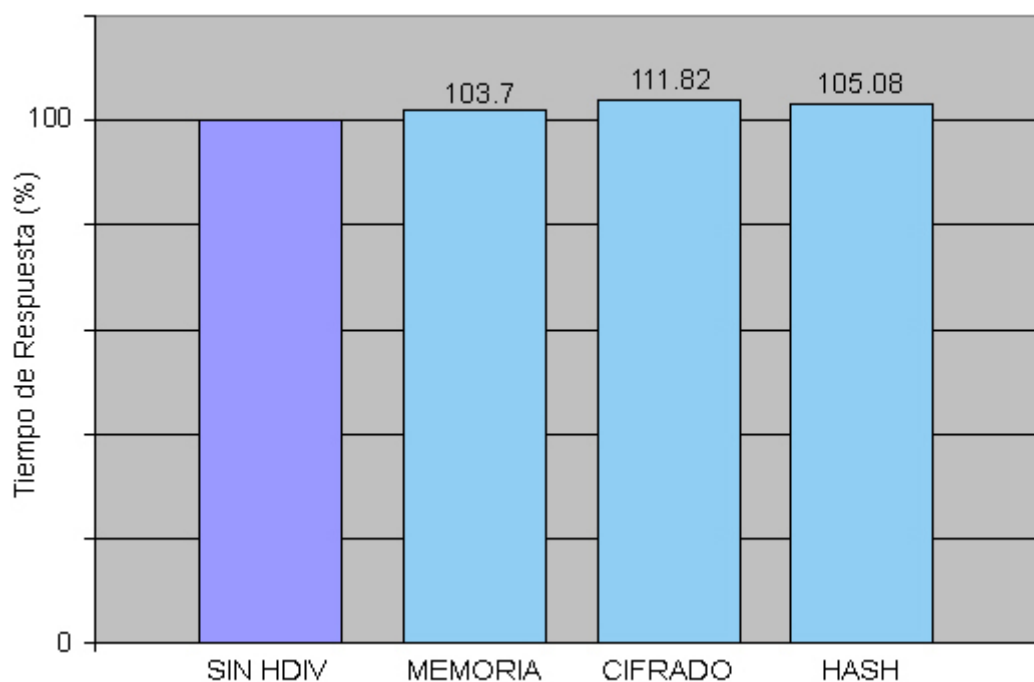
El tiempo medio de respuesta (en milisegundos) y el tiempo medio de respuesta superior consumido por HDIV (diferencia expresada en porcentaje respecto a la medición de la línea base sin ninguna solución para la integridad de los datos) se obtuvo de la media de los resultados de las repeticiones para cada estrategia o solución en HDIV.

Se muestra gráficamente el tiempo medio de respuesta para cada estrategia disponible en HDIV siendo 100 el porcentaje de tiempo consumido por la aplicación de línea base (sin haber aplicado HDIV).

3.1.1 Tomcat 5.0.28



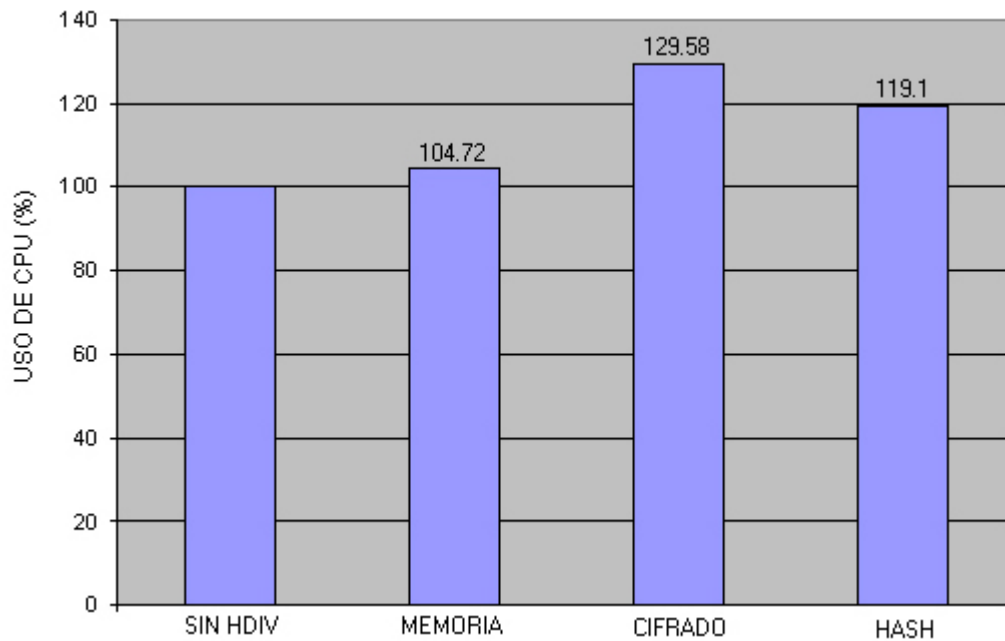
3.1.2 Websphere 6.0.1



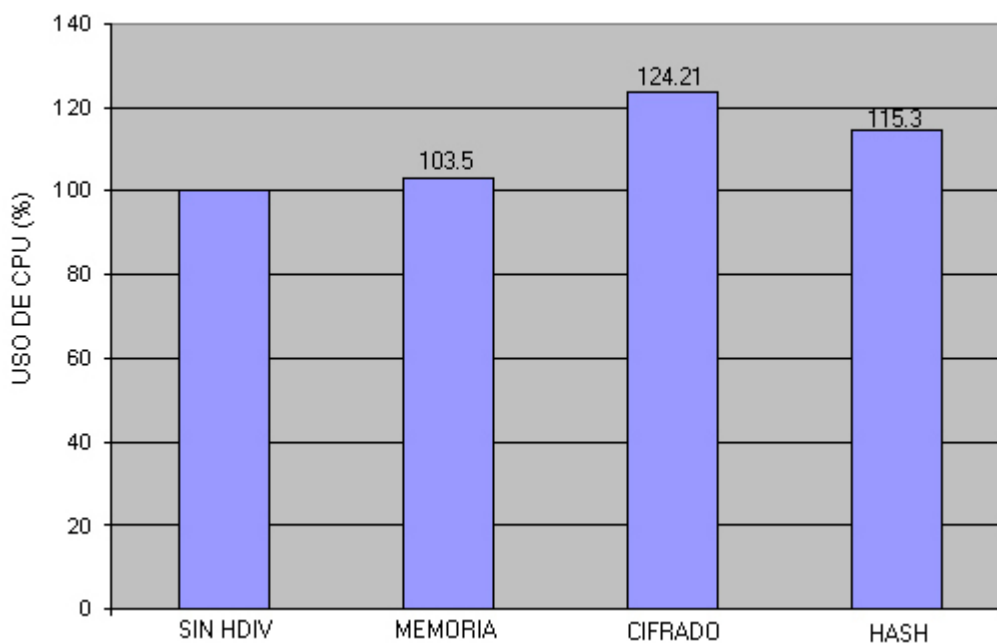
3.2 Uso de CPU

Se muestra gráficamente el uso medio de CPU que se ha obtenido en las repeticiones de las pruebas en cada estrategia de HDIV siendo 100 el uso de CPU consumido por la aplicación sin HDIV aplicado.

3.2.1 Tomcat 5.0.28



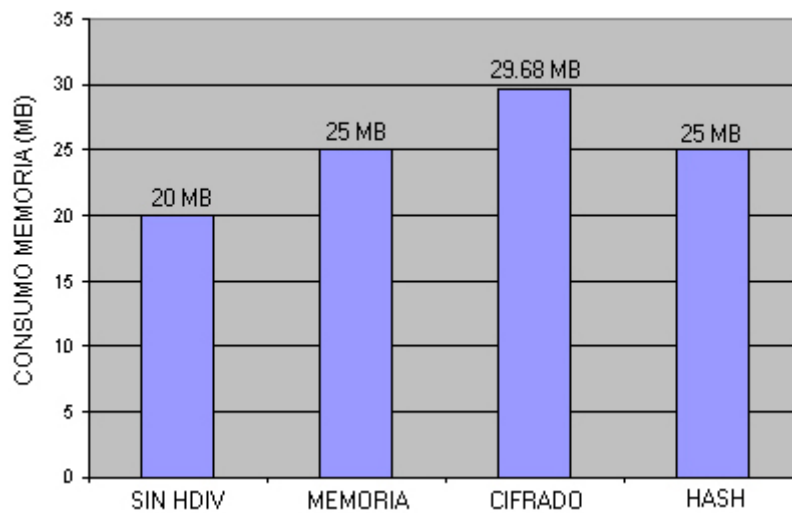
3.2.2 Websphere 6.0.1



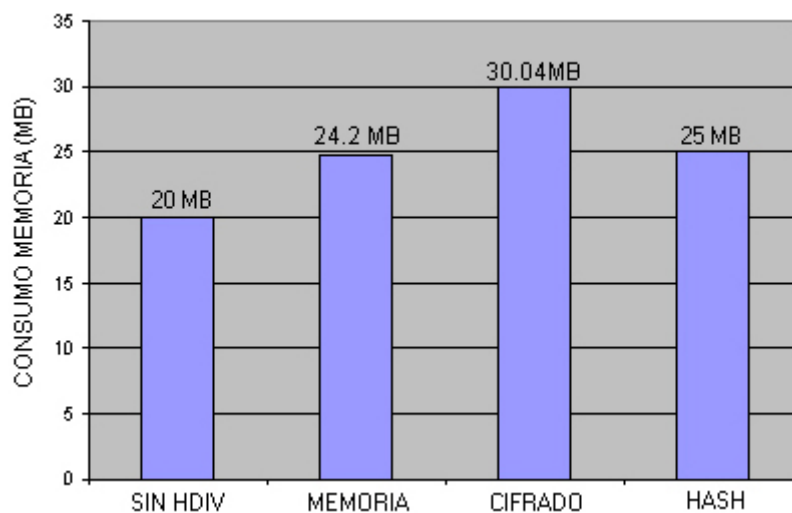
3.3 Consumo de Memoria

Se muestra gráficamente el consumo de memoria medio obtenido en los dos servidores de pruebas.

3.3.1 Tomcat 5.0.28



3.3.2 Websphere 6.0.1



3.4 Análisis de los Resultados

Observamos como en los dos entornos sobre los que se han realizado las pruebas (Tomcat y Websphere) los resultados han sido parecidos para cada estrategia aplicada. Por tanto podemos afirmar que a HDIV no le influye el entorno o servidor sobre el que corre la aplicación web.

La estrategia que mejores resultados ha obtenido es la de Memoria añadiendo a una aplicación sin securizar, en cuanto al tiempo de respuesta se refiere, del orden de un 3% sobre el valor de la línea base, habiendo obtenido resultados más que notables también en los otros dos parámetros analizados: uso de CPU y consumo de memoria. Esta estrategia añade integridad y confidencialidad de los datos.

La segunda mejor opción en lo que a tiempos medios de respuesta, uso de CPU y consumo de memoria se refiere es la estrategia de Hash, la cuál invierte un 5,08% más de tiempo medio en las respuestas en las aplicaciones web. Esta solución añade Integridad de los datos.

Como tercera propuesta tenemos la estrategia de cifrado la cual añade un 10,76% al tiempo medio de respuesta de una aplicación web. Esta solución añade integridad y confidencialidad de los datos.

3.4.1 Estado en Cliente vs. Estado en Servidor

Una vez analizados los resultados obtenidos, veámos a continuación, las ventajas e inconvenientes con respecto al almacenamiento del estado de HDIV en la parte cliente (Estrategia de Cifrado) o en el servidor (Estrategia de Memoria).

La estrategia de Hash combina la funcionalidad de las estrategias de Memoria y Cifrado en lo que al almacenamiento del estado se refiere guardándolo tanto en la parte cliente como en el servidor.

3.4.1.1 Estado en Cliente

Ventajas:

- Consumo de memoria inexistente entre diferentes peticiones
- No hay problemas de concurrencia
- No hay problemas con el botón *atrás* del navegador
- Posibilidad de reinicio
- Mayor escalabilidad

Inconvenientes:

- Mayor consumo de ancho de banda
- Consumo de CPU elevado
- Mayor consumo de memoria en la parte cliente

3.4.1.2 Estado en Servidor

Ventajas:

- Bajo consumo de ancho de banda
- Bajo consumo de CPU

Inconvenientes:

- Consumo de memoria por cada sesión de usuario en el servidor
- Consumo de memoria para soportar el botón *atrás* del navegador
- Menor escalabilidad (debido a la replicación de sesión entre nodos)

4. CONCLUSIONES

Como hemos podido comprobar en este estudio el tiempo que HDIV añade al tiempo medio de respuesta de una aplicación sin securizar, el uso que hace de la CPU y la memoria utilizada depende de la estrategia utilizada.

El tiempo extra que HDIV añade debe ser analizado y evaluado por los administradores de sistemas y aplicaciones con el fin de decidir si el tiempo extra es un tiempo razonable en sus aplicaciones, así como la importancia o necesidad de la confidencialidad de los datos.

Por tanto, podemos concluir afirmando que el consumo que HDIV hace de los recursos con respecto a las aplicaciones sin securizar es un consumo más que aceptable dando la posibilidad a los administradores de aplicaciones evitar las vulnerabilidades web de integridad de datos, así como la opción de ocultar los datos para conseguir la confidencialidad de los mismos, con un costo mínimo.

5. REFERENCIAS

- [1]. Apache JMeter
<http://jakarta.apache.org/jmeter/>

- [2]. ej-technologies Jprofiler
<http://www.jprofiler.com>