Benchmarking de implementaciones WMS-C de software libre.

R. García Martín[(1)], J.P. de Castro Fernández [(1)]

[(1)] Laboratorio de Infraestructuras de Datos Espaciales (IDELab), Escuela Técnica Superior de Ingenieros de Telecomunicación, Campus Miguel Delibes, Universidad de Valladolid, Camino del Cementerio s/n, 47011 Valladolid, {ricgar,juacas}@tel.uva.es.

RESUMEN

La propuesta WMS Cached (WMS-C) surge ante la necesidad de disponer de una solución más escalable al actual servicio web de mapas (WMS). Mediante la limitación de los parámetros de las peticiones a un conjunto discreto de valores, el servidor de mapas puede servir imágenes pregeneradas (teselas) a gran velocidad. En este documento se recogen las medidas de rendimiento realizadas sobre las implementaciones WMS-C: Tilecache, GeoWebcache y WMSCWrapper, todas ellas basadas en software libre.
Los resultados de los benchmarks reflejan que, con todas las peticiones cacheadas y bajo las mismas condiciones, la caché WMSCWrapper ofrece un mejor rendimiento que las otras implementaciones.
La difusión de estos resultados será de utilidad para ayudar a diversas entidades en la elección de una de estas soluciones libres en función de su objetivo de optimización.

Palabras clave: *WMS-C, benchmarking, GeoWebCache, TileCache, WMSCWrapper, software libre, JMeter.*

ABSTRACT

The WMS Cached proposal (WMS-C for short) is motivated by the need of more scalable approach to the current Web Map Service (WMS). By limiting the rendering parameters in the request to a discreet set of values, the WMS is able to serve pre-generated images (called tiles) very fast. This paper collects performance measurements realized over the main OpenSource WMS-C implementations: Tilecache, GeoWebcache and WMSCWrapper.
Benchmark results demonstrate that, with a fully loaded caché and under the same conditions, WMSCWrapper proxy offers a better performance in all scenarios under test.
The disclosure of these results may be of interest to some entities in choosing between these OpenSource solutions based on their particular optimization objectives.

Key words: *WMS-C, benchmarking, GeoWebCache, TileCache, WMSCWrapper, OpenSource, JMeter.*

INTRODUCCIÓN

La especificación WMS (*Web Map Service*) fue desarrollada para servir mapas cartográficos de forma interoperable, y uno de sus principales objetivos fue la flexibilidad. Sin embargo, debido a sus numerosos grados de libertad, los mapas necesitan generarse al vuelo. El dibujado de un mapa es una tarea de gran carga computacional que dificulta que el servicio pueda responder adecuadamente a un número elevado de peticiones simultáneas. En general los servicios WMS tienen serios problema de escalabilidad.

Para afrontar este problema se pueden reducir los dominios continuos de los parámetros a un conjunto discreto de valores. La zona geográfica queda dividida en una rejilla compuesta por elementos de geometría predefinida (denominadas teselas) e identificables mediante índices enteros. De esta forma es posible la actuación de mecanismos de caché entre el cliente y el servidor WMS (ver Figura 1) o incluso la prestación del servicio mediante una colección de imágenes pregeneradas.

En este último caso se reduce la carga computacional al mínimo mejorando el tiempo de respuesta y la escalabilidad.

Para tal fin se han desarrollado las especificaciones WMS Tile Caching (conocida como WMS-C) [1], Tile Map Service de OSGeo [2], y Web Map Tiling Service [3] del OGC, de las cuales la primera es la más extendida.

De entre las implementaciones de WMS-C del lado del servidor existentes destacan Tilecache[1], GeoWebcache[2] y WMSCWrapper [4].

Existen múltiples estudios comparativos acerca del rendimiento de servidores de mapas [5-7]. Sin embargo no se han constatado estudios de rendimiento comparativo entre sistemas de caché de mapas.

En el presente documento se realiza un estudio comparativo del rendimiento ofrecido por los tres sistemas de caché de mapas comentados, bajo diversos escenarios. Asimismo se presenta el procedimiento automatizado empleado para la realización de las pruebas, publicado para libre uso del mismo.

DESCRIPCIÓN DEL ENTORNO

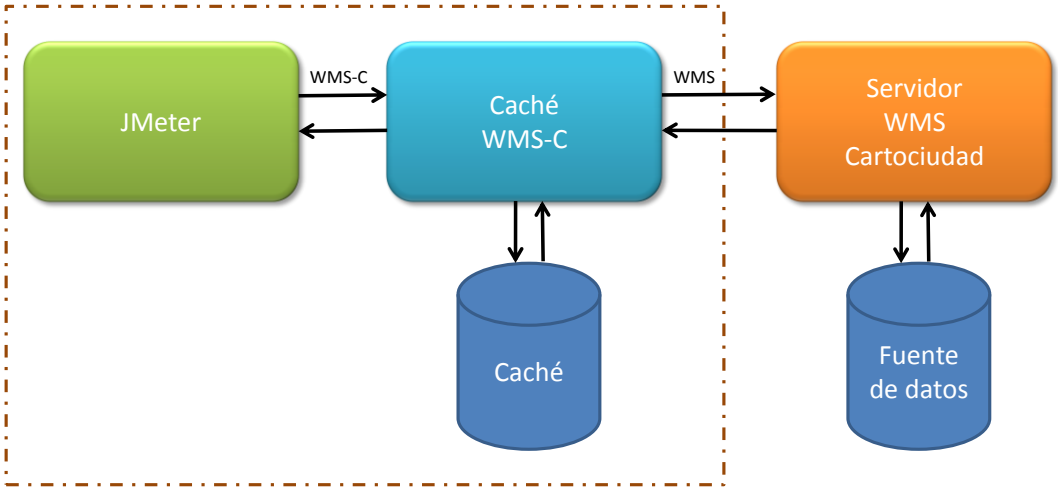

Figura 1: Arquitectura del entorno de pruebas

[1] http://tilecache.org
[2] http://geowebcache.org

El entorno de pruebas para la realización de las medidas de rendimiento de las diversas caché es el mostrado en la Figura 1. Los componentes localizados en el interior de la línea punteada están alojados en un servidor local, mientras que el resto de componentes están alojados en servidores remotos.

Como servidor WMS se ha utilizado el servidor web de mapas de Cartociudad[1].

La herramienta de pruebas JMeter está alojada en una misma máquina junto a las caché de mapas, y se utiliza la interfaz de red *localhost*. De esta forma se independizan los resultados de rendimiento de las caché de la red utilizada, evitando que una posible saturación de la conexión de red invalide los resultados obtenidos.

El servidor local es un Intel Pentium(R) Dual-Core E5300 a 2.60GHz con 4GB de memoria RAM, con Sistema Operativo Ubuntu 9.10, Kernel de Linux 2.6.31-14-generic.

Se ha utilizado TileCache v.2.10 corriendo sobre un servidor Web Apache 2.0 bajo *mod_pyhton* (se ha comprobado que esta instalación de la caché es la que obtiene un mejor rendimiento).

En cuanto a las implementaciones Java, se ha utilizado GeoWebCache v.1.2.1 y WMSCWrapper v.1, desplegadas sobre un servidor de aplicaciones Tomcat 6 con máquina virtual de Java de 64 bit (JDK 1.6.0_15).

Generación de las peticiones

Para la generación de los *bboxes* de las peticiones WMS se ha desarrollado una herramienta en Java que ofrece múltiples posibilidades:

- Se puede restringir el área geográfica de las peticiones, así como especificar las escalas de resolución deseadas.
- Posibilidad de generación “ordenada” de las peticiones (i.e. para simular un escenario de *seeding* de la caché) o genearación aleatoria.
- Permite autoconfigurarse a partir de un documento *capabilities* conforme a la especificación WMS-C de OSGeo, especificando la URL de la petición WMS getCapabilities.
- Utilización de una base de datos como fuente de peticiones. Pueden almacenarse en base de datos los registros de un servicio de mapas y posteriormente utilizar estos registros para la reproducción de un escenario real.

Tras la ejecución de la herramienta desarrollada se obtiene un fichero *.csv* que contiene los *bboxes* de las peticiones, que puede ser directamente interpretado por JMeter como fuente de peticiones.

Plan de pruebas en JMeter

Cada test en JMeter va asociado a un plan de pruebas. El plan de pruebas creado está completamente parametrizado a través de lo que se conoce como propiedades locales en JMeter. Estas propiedades pueden sobrescribirse directamente desde la línea de comandos, lo que facilita la automatización de las pruebas mediante su lanzamiento a través de *scripts*.

Para sobrescribir una propiedad hay que especificar como argumento `-J[prop_name]=[value]` en la llamada a JMeter. I.e. `jmeter –JHost=127.0.0.1`.

[1] http://www.cartociudad.es/portal/1024/index.htm

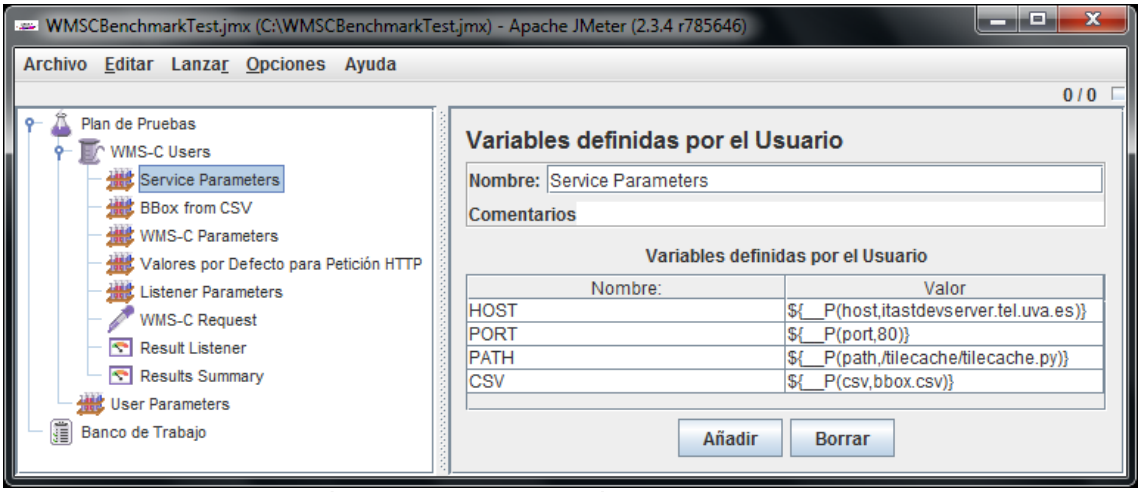

Figura 2: Parámetros de configuración del servicio en Jmeter

En el elemento WMS-C Users (ver Figura 2) se especifica el número de *threads* (usuarios simultáneos) que van a lanzar peticiones a las cachés, así como el tiempo de subida en segundos que tardan en iniciarse todos los procesos (en los *benchmarks* realizados se lanzan todos a la vez).

Los elementos con el icono corresponden a elementos de configuración. En ellos se especifican los parámetros para el acceso al servicio, como la dirección del servidor, el puerto, y el *path* de cada una de las caché (ver Figura 2), el fichero csv con los *bboxes*, los parámetros de la petición (ver Figura 3) conforme a la especificación WMS-C[1], y la configuración de los *listeners*.

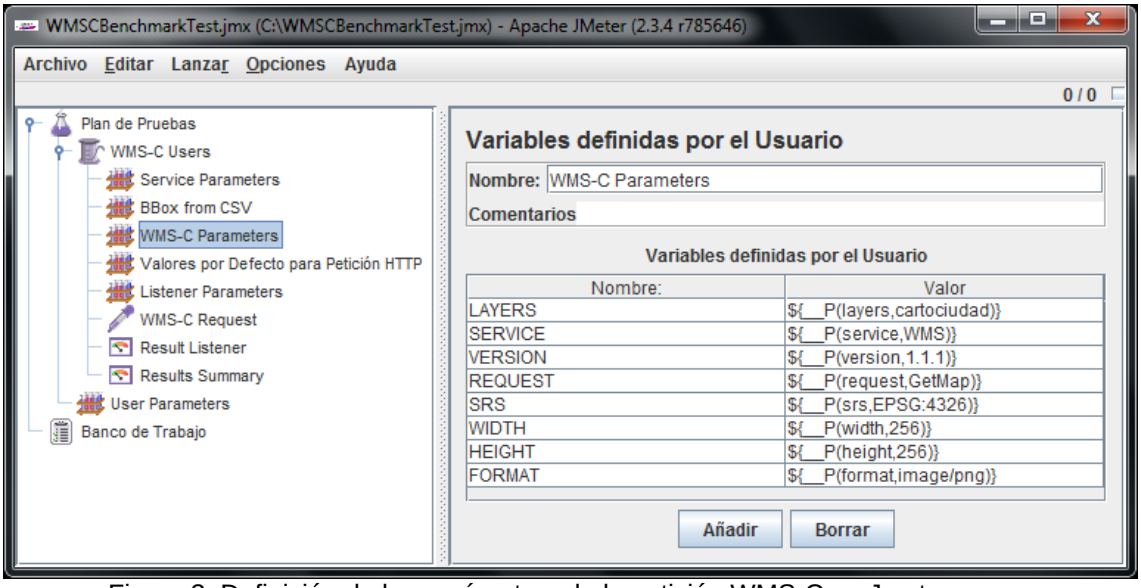

Figura 3: Definición de los parámetros de la petición WMS-C en Jmeter

El elemento WMS-C Request es el encargado de realizar la petición HTTP GET con los parámetros antes configurados.

Por último, se incluye un elemento "*listener*" que vuelca la información de las peticiones en un fichero csv, con los campos que se desean monitorizar (ver Figura 4). A pesar de que existen en JMeter *listeners* para el reporte gráfico de los resultados, esta opción se ha desestimado, dadas las limitaciones en la información a

representar, así como a la imposibilidad de agrupar los resultados de múltiples ejecuciones, esencial para realizar una comparativa de distintos servicios.

Es por ello que se ha optado por la interpretación de los resultados obtenidos por una aplicación externa a JMeter, tal y como se comenta a continuación.

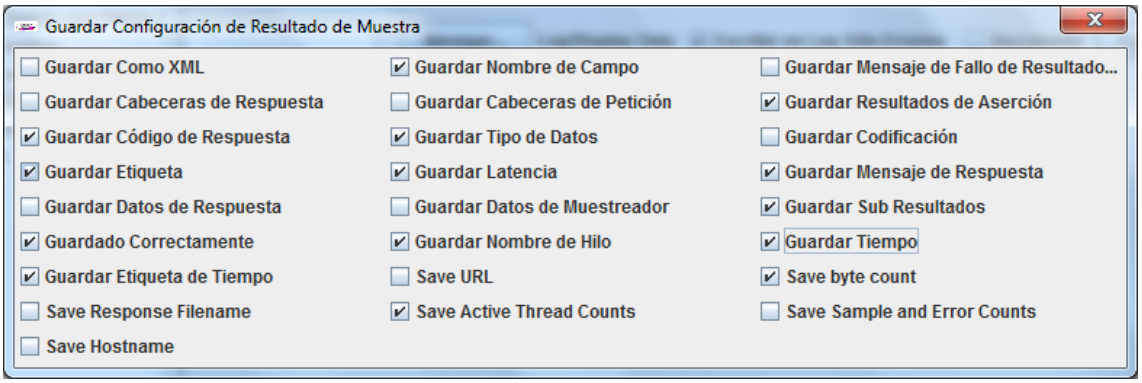

Figura 4: Configuración del listener en Jmeter

Reporte gráfico de resultados

Dadas las limitaciones ya comentadas en cuanto a la representación gráfica de resultados en JMeter, se ha optado por el desarro de una herramienta *"ad-hoc"* que suple dichas carencias.

La herramienta está desarrollada en Java, y utiliza el API POI de Apache[1] para la generación de unas hojas de cálculo, en formato .xls de Microsoft Excel, que aglutinan los resultados de la ejecución de varios planes de pruebas, y generan reportes gráficos conjuntos de todos ellos.

Esta herramienta facilita enormemente la tarea de representación de los resultados obtenidos, y evita posibles errores humanos en la recogida de los datos.

Orquestación de las pruebas

La orquestación de todos los componentes anteriores se realiza a través de una serie de *shell scripts* cuya funcionalidad se describe en el siguiente pseudocódigo:

```
foreach thread in threads {
  set global parameters;
  foreach cache in [tilecache, geowebcache, wmscwrapper] {
    set cache specific parameters
    launch jmeter
  }
  generate Excel report
  delete temporal files
}
```

RESULTADOS

Los *benchmarks* utilizados tienen como objetivo mostrar una comparativa del rendimiento ofrecido por las tres cachés consideradas. Se han considerado el tiempo

[1] http://poi.apache.org/

de respuesta (seg/petición) y la tasa de transmisión (peticiones/seg) como los indicadores más idóneos para la medida de rendimiento.

El análisis cuantitativo realizado debe interpretarse prestando atención a las medidas relativas, pues el objetivo principal es el de llevar a cabo una comparativa. Los valores absolutos de rendimiento son variables y pueden mejorarse empleando un *hardware* con mejores prestaciones, incrementando el ancho de banda, elección del formato de las imágenes generadas, etc., como se analiza en [7].

La cache, al recibir una petición de mapa, consulta si tiene la tesela cacheada en disco. De ser así, lo cual ocurre si se ha recibido la misma petición con anterioridad, se lee de disco y se devuelve al cliente. En caso contrario se realiza la petición al servidor WMS remoto, y una vez recibida la respuesta se almacena en disco y se envía al cliente.

Interesa, por tanto, evaluar el rendimiento en los dos casos anteriores. Para ello se ha planteado un escenario en el que todas las peticiones están cacheadas, y suponen por tanto aciertos en caché (Hot Benchmark), y otro escenario en el que la caché está completamente vacía (*Cold Benchmark*).

Seeding (*Cold Benchmark*)

Frecuentemente se utiliza la técnica de generación de peticiones de mayor tamaño que la tesela a *cachear* (esta "supertesela" se denomina *metatile*) y posteriormente se postprocesa para aprovechar la información disponible y generar nuevas teselas. De esta forma se reduce el número de consultas al WMS remoto.

En las figuras Figura 5 y Figura 6 se muestran los tiempos de respuesta y tasas de transmisión medios para distintos tamaños de *metatile*. Se observa que, a pesar de que aumenta la latencia por la creación y procesamiento de una imagen de mayor tamaño, la latencia media de las peticiones se reduce, pues disminuye el número de peticiones al servidor WMS remoto.

Un factor limitante a la hora de decidir el tamaño de *metatile* a utilizar es la cantidad de memoria que requiere la generación de la imagen. Nótese que para un factor de *metatile* de 5x5, para una petición de tesela de 256x256 *pixels* la imagen generada es de 1280x1280 *pixels*.

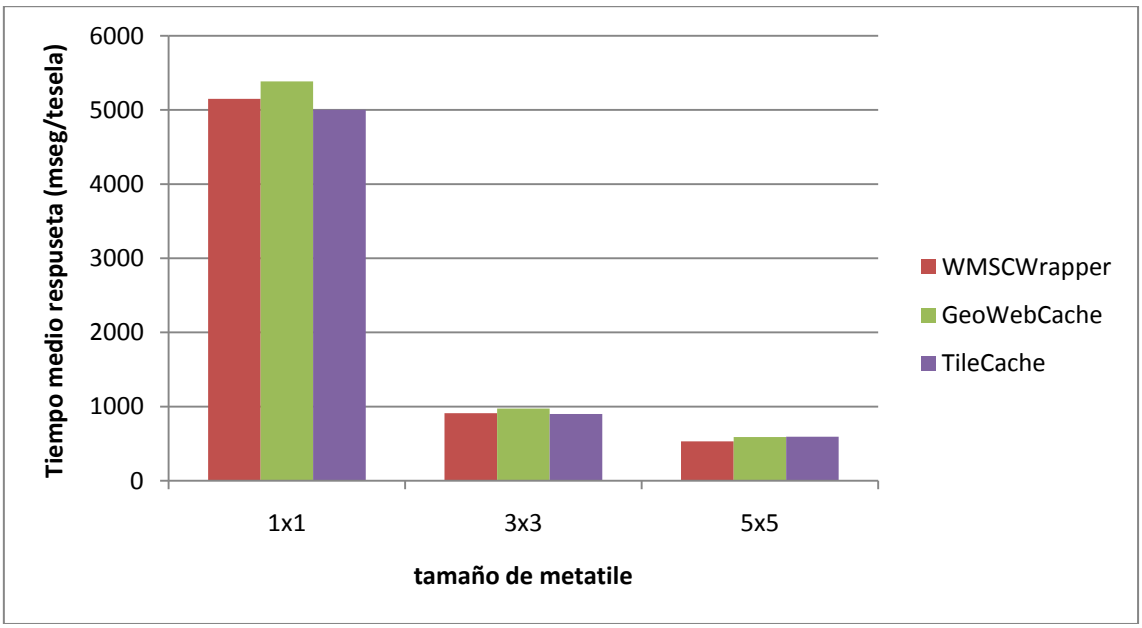

Figura 5: Tiempos de respuesta para distintos tamaños de metatile, con caché inicialmente vacía.

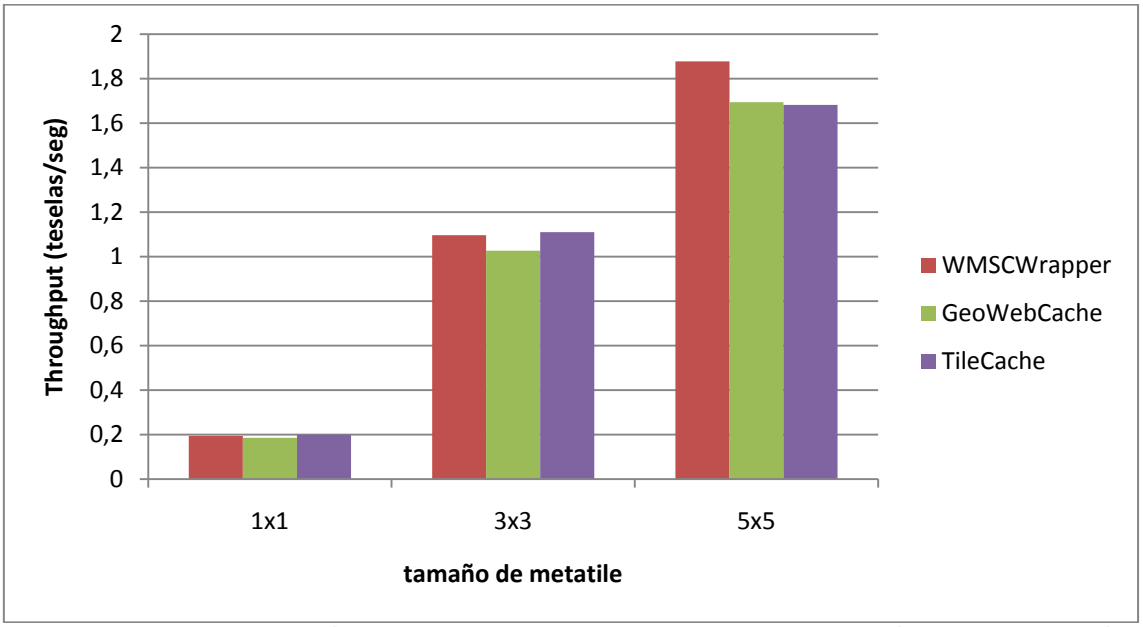

Figura 6: Tasa de transmisión para distintos tamaños de metatile, con caché inicialmente vacía.

Para ilustrar lo anterior considérese, por ejemplo, que en el servidor de mapas Geoserver, se configura a 16MB la cantidad máxima de memoria permitida para una única petición WMS (parámetro maxRequestMemory). Esta cantidad de memoria permite generar una imagen 2048x2048 con 4 bytes/pixel, lo que es equivalente a un metatile de 8x8. Si el SLD (Styled Layer Descriptor) contiene dos elementos FeatureTypeStyle para el dibujado de distintos tipos de línea en una autopista, el tamaño máximo de la imagen se limita a 1448x1448 pixels (metatile máximo de 5x5) .

Hot Benchmark

En las figuras Figura 7 y Figura 8 se recogen los resultados obtenidos para un escenario en el que todas las peticiones están en caché (mejor caso), en función del número de usuarios simultáneos.

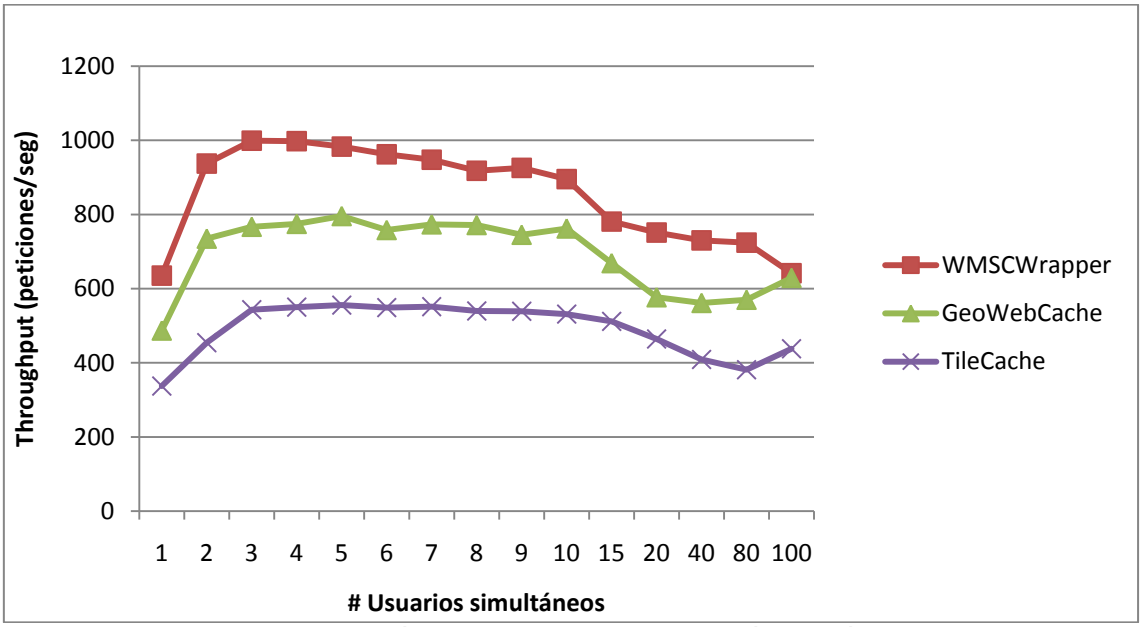

Figura 7: Tasa de transmisión global del sistema en función del número de usuarios simultáneos.

En la Figura 7 se observa que el *rendimiento global* del servicio aumenta al hacerlo el número de usuarios hasta que se llega a los 5 usuarios. En ese punto se alcanza la saturación del servicio y una demanda de servicio mayor provoca una degradación de la calidad de servicio global del sistema.

En la Figura 8 se muestra la tasa de transmisión (peticiones/segundo) del servicio normalizada por el número de usuarios. Esta interpretación de los resultados muestra con mayor claridad la calidad de servicio efectivamente proporcionada a los usuarios del proxy. Se observa cómo al aumentar el número de usuarios aumenta sistemáticamente la latencia percibida por cada uno de ellos.

Se puede apreciar que el *proxy* WMSCWrapper ofrece el mejor rendimiento de las implementaciones probadas.

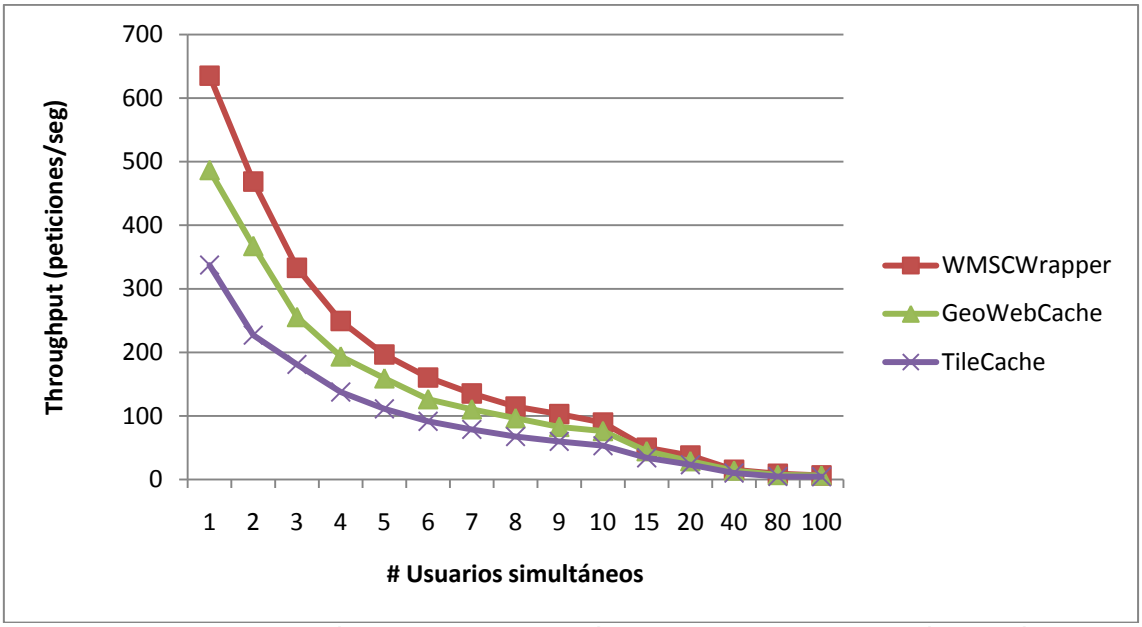

Figura 8: Tasa de transmisión (normalizada por número de usuarios) en función del número de usuarios simultáneos.

CONCLUSIONES

En el presente artículo se han presentado unos resultados preliminares obtenidos a partir de las medidas de rendimiento realizadas sobre los distintos sistemas de caché puestos a prueba, TileCache, GeoWebCache y WMSCwrapper, en distintos escenarios.

En primer lugar las pruebas se han realizado bajo un escenario con las cachés inicialmente vacías, con un único hilo de peticiones, y variando el tamaño de metatile. Los resultados obtenidos indican que el rendimiento conseguido aumenta con el tamaño del metatile. Sin embargo, éste no puede aumentarse indiscriminadamente, dado el elevado consumo de memoria requerido para la generación de una imagen de gran tamaño.

En el segundo escenario, con las cachés inicialmente llenas y para distinto número de usuarios simultáneos, se observa cómo el rendimiento conseguido aumenta monótonamente con el número de usuarios hasta llegar a un determinado número en el que se satura, reduciéndose el rendimiento para un mayor número de usuarios.

AGRADECIMIENTOS

El desarrollo de este trabajo ha sido posible gracias a la financiación por parte del Instituto Geográfico Nacional en el marco del Proyecto Conjunto al amparo del convenio de colaboración entre la dirección general del Instituto Geográfico Nacional y la Universidad de Valladolid en su edición de 2009.

REFERENCIAS

[1] OsGeo, “WMS Tile Caching,” *WMS Tile Caching - OSGeo Wiki*, Mar. 2009.
[2] OSGeo, “Tile Map Service Specification,” *Tile Map Service Specification - OSGeo Wiki*, 2008.

[3] Joan Masó, Keith Pomakis, y Núria Julià, “Web Map Tiling Service Standard,” Feb. 2009.
[4] R. García y J.P. de Castro, “WMSCWrapper. Implementación WMS-C OpenSource para servicios WMS teselados.,” *IV Jornadas de SIG Libre*, Girona: 2010.
[5] D. Deepak P., J. Rodrigo, y J. Rosales, “Medición de rendimientos de servicios WMS con JMeter,” *IDE, aplicaciones al planeamiento y la gestión del territorio*, Tenerife: 2008.
[6] M.Á. Esbrí Palomares y J.V. Higón Valero, “Pruebas benchmark de soluciones cliente/servidor en software libre,” Madrid: 2005.
[7] J.G. Arne Kepp, “Making Maps Fast - How to increase the performance of GeoServer,” Nov. 2009.