WMSCWrapper. Implementación WMS-C OpenSource para servicios WMS teselados.

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, plopesc@gmail.com {ricgar,juacas}@tel.uva.es.

RESUMEN

En este documento se presenta el proyecto Open Source WMSCWrapper: un innovador sistema de caché de teselas geográficas. Su arquitectura permite la inclusión de componentes y sondas experimentales, resultando idóneo para experimentación con nuevas estrategias de caché.
El sistema está implementado en Java como un conjunto de servlets que exponen diversos interfaces de servicio como la recomendación WMS del OGC y el perfil WMS-C, así como el acceso por medio de interfaces REST, utilizados por Google Earth, Google Maps y Microsot Bing Maps.
Cada petición es analizada en busca del tipo de cliente y de los parámetros obligatorios u opcionales y después transferida a una serie de componentes intercambiables que pueden preprocesar o postprocesar la información según las necesidades.
A diferencia de otras implementaciones de WMS-C, se implementan técnicas de gestión de la cache que aplican heurísticas definidas para un dominio de aplicación. De esta manera, se intenta maximizar la probabilidad de acierto, manteniendo el consumo de recursos dentro de unos rangos definidos.
Para ello, la actividad de la cache se monitoriza permanentemente almacenando los resultados en un índice espacial en memoria. Este proyecto ofrece un banco de pruebas con el que experimentar con diversas implementaciones de este índice y los indicadores que contienen, así como distintas políticas de reemplazo.

Palabras clave: *WMS, WMS-C, WMSCWrapper, caché, software libre.*

INTRODUCCIÓN

Las diversas especificaciones WMS del OGC [1] ofrecen una gran flexibilidad en el servicio. Los parámetros espaciales de las peticiones no están restringidos, lo que hace que cada petición de mapa deba ser atendida en tiempo real mediante un procedimiento, generalmente costoso, que implica acceso a datos de origen, aplicación de estilos, composición de capas y codificación de la imagen comprimida.

Este procedimiento se ha demostrado ineficaz para satisfacer la demanda de algunas aplicaciones de difusión masiva como se expone en [2] tras la experiencia del servidor OnEarth de la NASA. Por este motivo, generalmente los servicios comerciales más populares se prestan con servidores no OGC en los que el espacio geográfico está teselado de acuerdo a una rejilla predefinida y cuyo contenido está frecuentemente pregenerado [3,4].

Cuando un servicio OGC se va a utilizar en un escenario exigente con una información poco parametrizable y estacionaria, se puede utilizar el patrón *proxy web cache* para conseguir una mejoría en la calidad de servicio. El *proxy* es un dispositivo que se interpone de forma preferiblemente *transparente* entre el cliente y el servicio final, como se ve en la Figura 1.

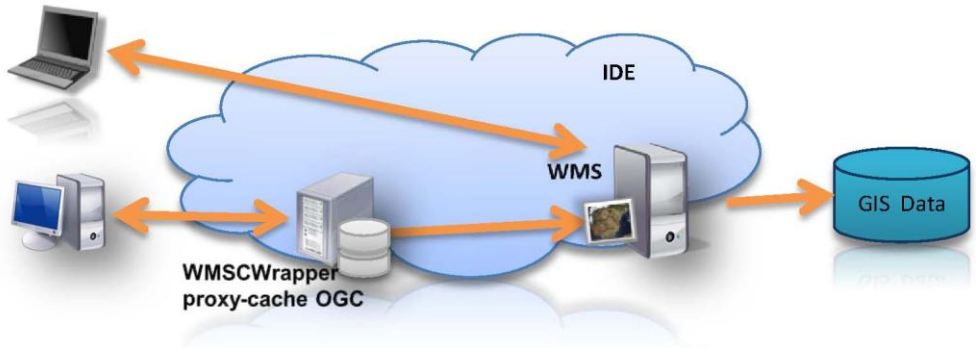

Figura 1: Proxy WMS-C en la IDE.

La popularidad y necesidad de esta estrategia ha provocado la aparición de algunos mecanismos de acceso no estándar [5][6][7], y otras recomendaciones más abiertas como la recomendación WMS de OSGeo [8] o las especificaciones en desarrollo del propio OGC [9][10].

Las características particulares de la información geográfica de una *cache* espacial permite suponer que los algoritmos de reemplazo y carga inicial de la misma deben tener en cuenta las características espacio temporales del comportamiento de los usuarios y la correlación multidimensional de las teselas.

Las *cachés* estudiadas [11][12] utilizan algoritmos muy básicos (LRU) de gestión de sus *caches* que, en su adopción en aplicaciones no geográficas, han demostrado no ser los más eficaces cuando los objetos no son homogéneos[13].

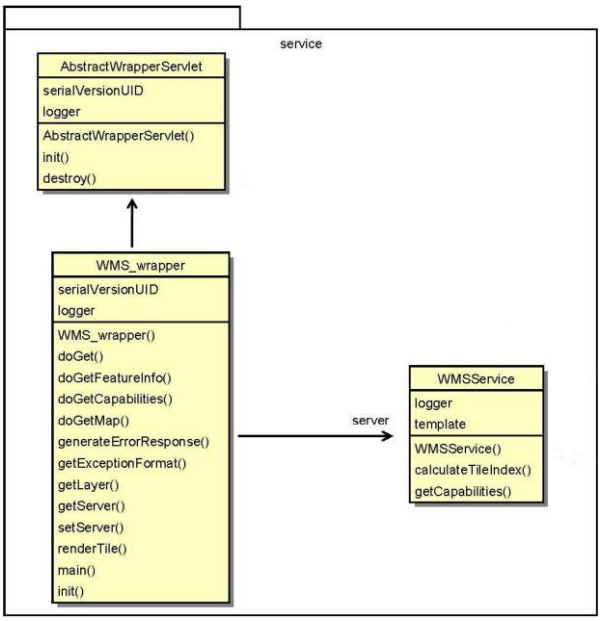

Figura 2: Interfaz de servicio para el proxy cache implementado mediante un servlet y configurado a partir de un fichero XML.

Descripción

En el seno del IDELab (**Lab**oratorio de **I**nfraestructuras de **D**atos **E**spaciales) de la Universidad de Valladolid se ha desarrollado WMSCWrapper, un *proxy cache* para WMS-C. Se trata de un sistema implementado en Java siguiendo la especificación J2EE. Consta de un servlet principal que encapsula las operaciones definidas en la especificación WMS (ver).

El servidor puede gestionar simultáneamente diversas cachés de teselas, denominadas *Tilesets*, con configuraciones inicialmente independientes. Estas *Tilesets* se pueden exportar a través de distintos interfaces de acceso: actualmente como WMS *proxy*, pasarela KML para Google Earth[1], así como soporte para Google Maps[2] y Microsoft Bing Maps[3]. Las características de los servicios y de los conjuntos de datos se definen por medio de un fichero de configuración XML, un ejemplo del cuál se muestra en el Listado 1.

Para dar servicio a peticiones WMS, el *proxy* recibe todas las peticiones a través del componente *WMS_wrapper* que actúa como controlador para decidir el *WMSService* encargado de procesar la petición, comprobar la validez de los parámetros y desencadenar el proceso de obtención de una tesela.

En el mismo fragmento XML donde se especifica cada *TileSet* se definen las políticas de gestión de la *caché* asociada. Las opciones de este importante componente son:

- El tipo de gestor de caché enlazado en tiempo de ejecución: Lo que permite utilizar diversos algoritmos mediante *plug-ins*.
- El tamaño máximo a ocupar en el disco: es un parámetro común a todos los gestores de caché mediante el que se limita el consumo de recursos.
- Las condiciones de purgado y cargado automático de teselas: En la implementación actual el parámetro es el tiempo de vida de las teselas. Estas condiciones pueden especificarse por zonas geográficas, permitiendo el establecimiento (manual todavía) de zonas prioritarias.
- La creación de *"demonios"* para el mantenimiento específico de las teselas. El sistema lanza tareas que utilizan las estadísticas recopiladas para la ejecución de las heurísticas y algoritmos de gestión.

A continuación se reproduce, a modo de ejemplo, un fichero de configuración en el que se define completamente el servicio de dos conjuntos de teselas (IDE y STATS) a través de interfaces WMS y REST para *Google Earth*.

```
<?xml version="1.0" encoding="UTF-8"?>
<WMSCacheConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <KMLService>
    <ServiceType>KML</ServiceType>
    <Name>KML gateway IDELab</Name>
    <Title>Pasarela KML</Title>
    <Abstract>Descripcion del servicio experimental</Abstract>
    <OnlineResource>http://itastserver.tel.uva.es/WMS_C_wrapper</OnlineResource>
  </KMLService>
  <WMSCService>
```

[1] http://earth.google.es/
[2] http://maps.google.es/
[3] http://www.bing.com/maps/

```
  <ServiceType>WMS-C</ServiceType>
  <Name>WMS-C IDELab</Name>
  <Title>Titulo del servidor</Title>
  <Abstract>Descripcion del servicio experimental</Abstract>
  <OnlineResource>http://itastserver.tel.uva.es/WMS_C_wrapper/wms</OnlineResource>
 </WMSCService>
 <TileSet name="STATS" type="Stat">                                              (1)
  <LayersetRef>IDE</LayersetRef>
  <SRS>EPSG:4326</SRS>
  <BoundingBox SRS="EPSG:4326" minx="-23.0132795" miny="27.636523" maxx="9.1810205" maxy="43.793673" />

  <profile type="specific" levels="25" maxResolution="1.40625"> <!-- global|specific -->
   <BoundingBox SRS="EPSG:4326" minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0" />
   <Resolutions levels="20">0.703125 0.494384765625 0.3515625 0.17578125 0.087890625 0.0439453125
                            0.0219726563 0.010986328125 0.0054931640625</Resolutions>
   <Width>256</Width>
   <Height>256</Height>
  </profile>
  <Format>png</Format>
  <Layers>stats</Layers>
  <Styles />
  <cacheconfig method="dummy" debug="false"> <!-- simplecache or prefech or LRUcache -->
   <MaxCacheSize>260M</MaxCacheSize>
   <CachePath>c:/temp/sys/cache/cartociudad_stats</CachePath>
   <MinimumTimeout>10</MinimumTimeout>
  </cacheconfig>
  <loader type="buffered">
   <prefetchBuffer>0</prefetchBuffer> <!-- number of tiles to render by WMS around asked tile-->
  </loader>
 </TileSet>

 <TileSet name="IDE" type="WMS">
  <url><![CDATA[http://www.idee.es/wms/IDEE-Base/IDEE-Base?]]></url>
  <SRS>EPSG:4326</SRS>
  <BoundingBox SRS="EPSG:4326" minx="-37.474148" miny="2.7077813" maxx="22.74257" maxy="57.752808" />
  <profile type="global" levels="20" maxResolution="0.703125"> <!-- global|specific -->
   <BoundingBox SRS="EPSG:4326" minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0" />
   <Resolutions levels="20"> 0.3515625 0.17578125 0.087890625 0.0439453125 0.0219726563
0.010986328125</Resolutions>
   <Width>256</Width>
   <Height>256</Height>
  </profile>
  <Format>png</Format>
  <Layers>Todas</Layers>
  <Styles />
  <cacheconfig method="SpatialCache" debug="false" implementationClass="org.jp.cache.HeuristicalSpatialCacheManager">
                                                                                 (2)
   <MaxCacheSize>260M</MaxCacheSize>
   <CachePath>/var/cache/cartociudad</CachePath>
   <MinimumTimeout>-1</MinimumTimeout>
   <MinimumTimeout resIndex="10,11,12,13,14,15,16,17,18,19,20,21,22,23">9000000</MinimumTimeout>
   <MinimumTimeout SRS="EPSG:4326" minx="-4.85287" miny="41.568" maxx="-4.61426" maxy="41.72710"
                                                                                 (3)
    resIndex="1,2,3,4,5,6,7,8,9">60000</MinimumTimeout>
   <task name="CacheCleaner IDEE" implementationClass="org.jp.task.tasks.FullCacheCleaner">
    <CacheCleaner />
   </task>                                                                       (4)
```

```
    <task name="Seeder for IDEE" implementationClass="org.jp.task.tasks.SeederJob">
     <Seeder>
      <SeederBoundingBox resIndex="5,6,7,8,9,10,11,12,13,14" SRS="EPSG:4326" minx="-4.85287"
                         miny="41.568" maxx="-4.61426" maxy="41.72710">
       <ServerLoad>2</ServerLoad>
      </SeederBoundingBox>
      <SeederBoundingBox resIndex="0,1,2,3,4" SRS="EPSG:4326" minx="-180" miny="-90" maxx="-180" maxy="90">
       <ServerLoad>2</ServerLoad>
      </SeederBoundingBox>
      <ServerLoad>2</ServerLoad>
     </Seeder>
    </task>
   </cacheconfig>
   <loader type="buffered">
    <prefetchBuffer>1</prefetchBuffer> <!-- number of tiles to render by WMS around asked tile-->
   </loader>
  </TileSet>
</WMSCacheConfig>
```

(5)

Listado 1: Fichero de configuración de WMSCWrapper

En el documento anterior cabe destacar los siguientes fragmentos:

En (2) se define el gestor de *caché* con índice espacial de recopilación de estadísticas.

En (3) se especifican los tiempos de vida preferidos para diversas combinaciones de *bounding box* y escalas. Ésta es una manera básica de regular la QoS (*quality of service*) del servicio utilizando la intuición del gestor para definir las heurísticas de purga y carga de teselas.

Mediante (4) se crea un proceso en segundo plano (*demonio*) que realiza el mantenimiento de la *caché* incluso en ausencia de peticiones de usuarios. Las otras implementaciones conocidas realizan las operaciones de mantenimiento en el mismo instante en que se realiza la petición de la tesela. Con esta aproximación se pretende aumentar la QoS general de la *caché* durante las horas de servicio con menor carga.

En (1) se define un *Tileset virtual* que se genera al vuelo a partir de las estadísticas recopiladas por la caché de la capa. Aunque en el fondo es una herramienta para la depuración del funcionamiento de los algoritmos, demuestra una de las aplicaciones de generar teselas en tiempo real a partir de la información ya existente en la *caché*. En la Figura 5: Capa de teselas virtuales representando una de las estadísticas recopiladas junto con información de depuración.Figura 5 se muestra una captura en la que se visualiza esta capa.

Sobre esta infraestructura se han desarrollado los módulos de experimentación que se presentan a continuación en la Figura 3.

COMPONENTES

Cache con índice espacial

Se ha implementado un componente de caché espacial (ver *HeuristicalSpatialCacheManager* en la Figura 3) cuyo objetivo principal es llevar un registro espacial en tiempo real de la actividad de la caché. Toda la actividad de ésta queda registrada en un índice *QuadTree* mediante el cual se pueden realizar

búsquedas espaciales de gran eficacia, especialmente los recorridos en profundidad. Dada la naturaleza exponencial de la estructura de la pirámide, resulta impráctico albergar un índice del mismo tamaño que la *caché* que queremos representar, por lo que se ha implementado un *Quadtree* truncado en un nivel concreto que seleccionamos mediante un parámetro. Es evidente que ajustando este parámetro se puede transformar el índice para que cubra completamente la pirámide de teselas.

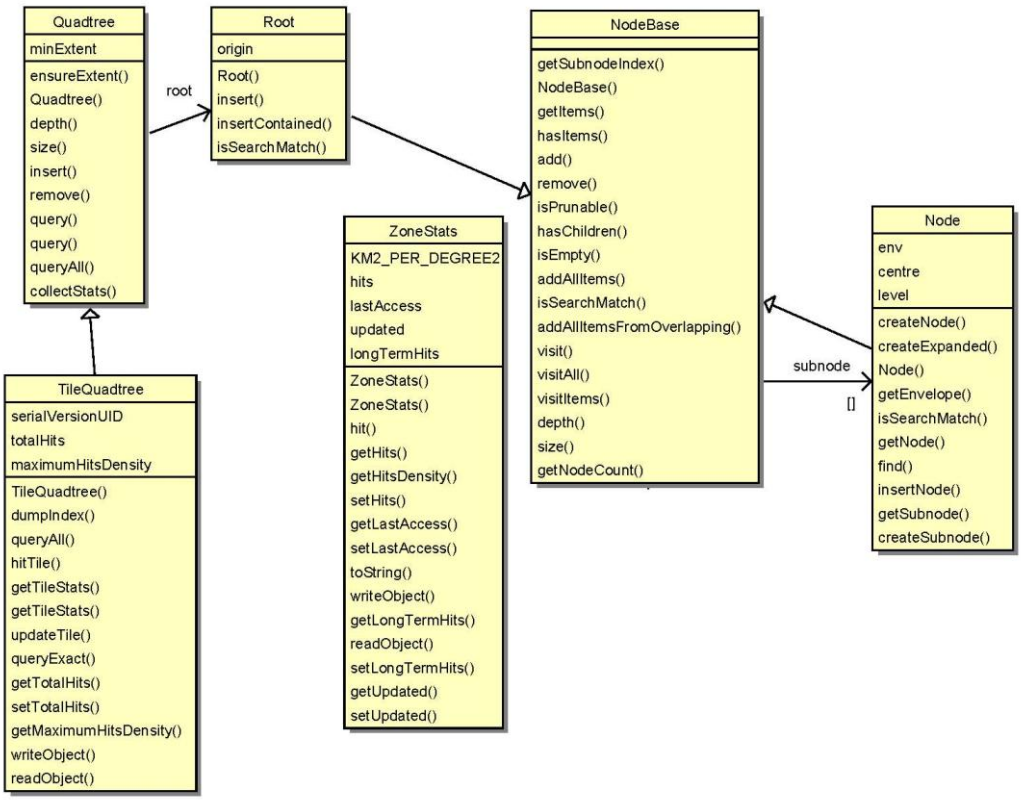

Figura 3: Diseño del sistema *caché*

Este índice puede utilizarse para almacenar cualquier característica primaria o derivada que generen nuestros algoritmos o necesiten las heurísticas de nuestros experimentos.

En esta versión se ha implementado una estrategia básica inicial para obtener una aproximación a la probabilidad de acceso a las zonas geográficas representadas por las teselas.

Dado que no resulta práctico realizar operaciones a resoluciones altas, es preciso consolidar características espaciales en niveles intermedios de forma que esta información nos permita extrapolar y predecir las características de los niveles inferiores, que consideraremos inaccesibles. Se ha implementado el esquema ilustrado en la Figura 4.

La probabilidad de acceso a una tesela $P_{req}\{T(i,j,n)\}$ es uno de los parámetros más importantes que caracterizan el comportamiento de los usuarios de la *caché* y puede estimarse estadísticamente. Partiendo de la condición previa de que las peticiones están restringidas a la rejilla de referencia, la probabilidad de acceso a una tesela de coordenadas $T(i,j,n)$ puede aproximarse como el cociente entre el es el número $n_{h(i,j,n)}$ de aciertos en caché de la tesela con índice (i,j) en el nivel de resolución n, y es el número total de peticiones realizadas a la caché.

De acuerdo con esta aproximación, el índice espacial registra cada petición de elementos $T(i,j,n)$ mediante dos contadores de "*hits*" que reflejan la historia a medio

y a largo plazo de la zona geográfica cubierta por esa tesela. Cada *hit* en un determinado nivel se anota en las estadísticas de las teselas de niveles superiores. Este comportamiento se ilustra en la **¡Error! No se encuentra el origen de la referencia.**. De esta forma la probabilidad de un *bounding box* geográfico se calcula como el cociente del número de peticiones englobadas por dicho *bounding box* y el total de peticiones del sistema.

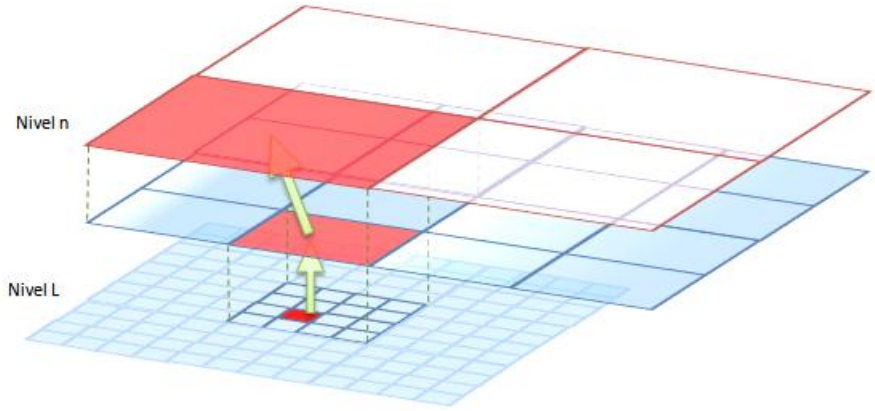

Figura 4: Método para extrapolar características probabilísticas usando el principio de localidad.

Dado que el área geográfica cubierta por cada elemento $T(i,j,n)$ depende exponencialmente de *n*, el número de peticiones recibidas aumenta a medida que *n* disminuye y la probabilidad de acceso deja de ser una buena heurística en los algoritmos que operen en distintas escalas. Por ese motivo las heurísticas implementadas utilizan en lugar de la probabilidad, la *función densidad de probabilidad media* de cada tesela $f_{req}(x,y,n)$. Es inmediato ver que esta función puede entenderse en nuestra implementación como la densidad espacial normalizada de *hits* de *caché*.

Con este sistema de estimación de $f_{req}(x,y,n)$ se pueden implementar algoritmos de delimitación de zonas homogéneas (Z_i) haciendo equivalente esta función a la densidad espacial normalizada de peticiones. En la Figura 5 se muestra el resultado de la actividad de este índice espacial en un nivel de la *cache*.

Prototipo de generación predictiva de teselas

Otra línea de investigación en progreso es la mejora, mediante estrategias de racionalización de la carga ejecutadas en el proxy, del tiempo incurrido en construir un nuevo objeto en la caché a partir de los servicios originales.

En el proceso de generación de teselas hay diversos cuellos de botella que se pueden mejorar. Una de las partes del proceso de dibujado es el acceso a los almacenes externos de información geográfica. Ya que las consultas suelen estar optimizadas mediante índices espaciales, cabe la posibilidad de que al reducir el número de consultas ampliando el *bounding box* de la zona a generar, se obtenga una mejoría.

Otro beneficio obtenido indirectamente es la reducción del problema del etiquetado fraccionado o redundante provocado por el teselado de las peticiones.

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 procesa para aprovechar la información disponible y generar nuevas teselas.

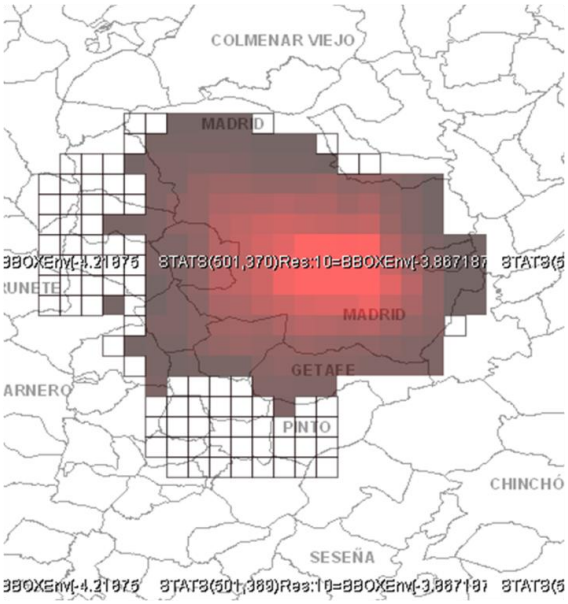

Figura 5: Capa de teselas virtuales representando una de las estadísticas recopiladas junto con información de depuración.

Las implementaciones investigadas permiten especificar el número adicional de teselas alrededor de la realmente solicitada (denominémoslo *buffer de N teselas*) que se van a pedir en una sola petición al servidor WMS. De esta manera se le pide al WMS una *metatile* de tamaño $(2N+1)^2$ teselas centrada en el elemento realmente solicitado. En un escenario de *cache* no completa (pero no vacía) esta elección del área a generar no es muy eficiente puesto que es muy probable que algunas de las teselas próximas a la solicitada ya estén disponibles.

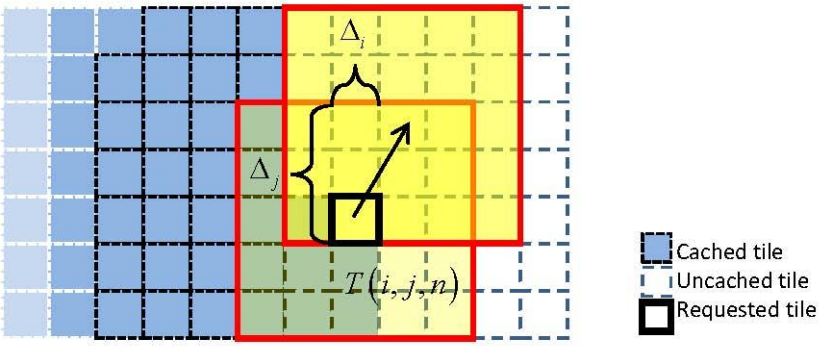

Figura 6: Estrategia de mínima correlación con la cache para la generación de *metatiles*

Partiendo de la suposición de que la zona de la tesela solicitada no está homogéneamente cargada, se ha implementado un algoritmo para la elección óptima de las *metatiles* a generar. El procedimiento, ilustrado en la Figura 6, busca obtener en función del estado de la *caché* la *metatile* que, conteniendo la tesela solicitada, minimice la correlación espacial:

$$R_n(\Delta i, \Delta j) = \sum_{l=i-N}^{j+N} \sum_{m=j-N}^{j+N} h[l+\Delta i, m+\Delta j] \qquad (1)$$

Interpretando la correlación en (1) como una medida de la semejanza de la información contenida en ambos objetos (*metatile* y *caché*), parece evidente que la *metatile* que tenga una menor correlación espacial con el estado de la caché es

aquella que proporciona la mayor información al sistema, puesto que la información que contiene es complementaria en mayor grado a la ya disponible. En la Figura 6 se ilustra la configuración con la que se consigue un mínimo en la redundancia o en la información mutua.

La implementación llevada a cabo incluye además un procedimiento en segundo plano para realizar el postproceso de las teselas adicionales. De esta forma se busca minimizar también el tiempo $\tau_h(hit)$ necesario para obtener un objeto de la cache. En el Listado 1(5) se muestra un ejemplo de configuración de *metatiling* con un buffer de una unidad (*metatile 3x3*).

Almacenamiento de las teselas

El almacenamiento de las teselas se realiza con persistencia en disco siguiendo la siguiente estructura de directorios: raíz/layer/resolución/x/y.<extension> (ver Figura 7).

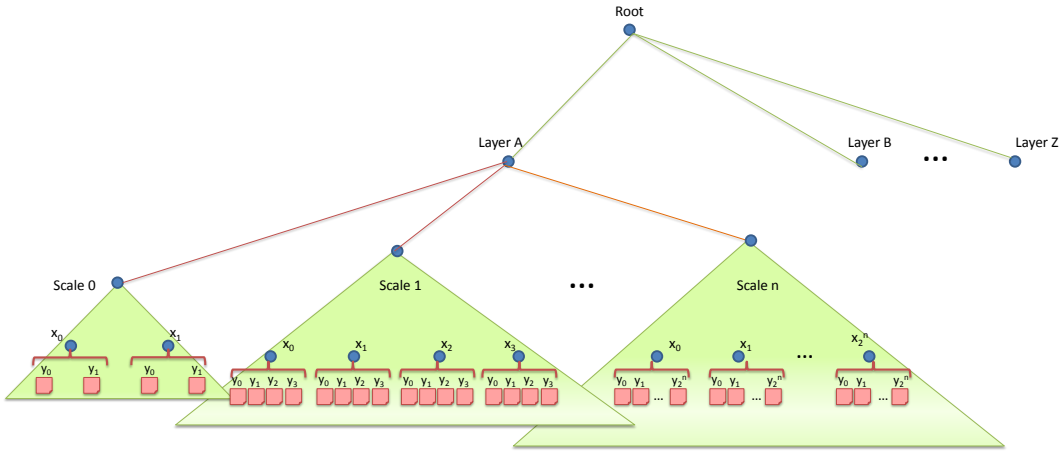

Figura 7: Estructura de directorios para el almacenamiento persistente de las teselas en WMSCwrapper

El sistema de ficheros NTFS utilizado en Windows permite almacenar hasta $2^{32} - 1$ ficheros en cada directorio, por lo que siguiendo esta estructura de directorios se podrían almacenar hasta 31 niveles de resolución, suficiente para las necesidades actuales.

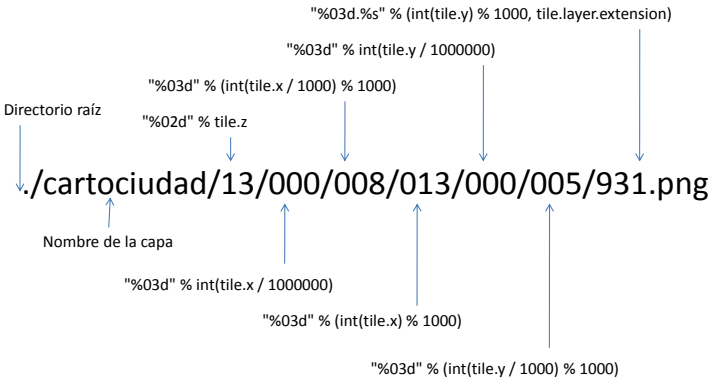

Figura 8: Estructura de directorios para la persistencia de teselas en TileCache

En la se muestra la estructura de directorios utilizada por TileCache. Comparando ambas aproximaciones se observa que TileCache utiliza una profundidad de más del doble de directorios que WMSCwrapper (7 frente a 3), por lo que la primera sufre una mayor sobrecarga por el almacenamiento de un número mayor de directorios.

INTERFACES DE ACCESO

A parte de la interfaz WMS-C especificada en [14], el *proxy* implementado funciona como pasarela KML para Goolge Earth, y soporta el acceso de los clientes Google Maps y Microsoft Bing Maps. En la Figura 10 se muestran capturas de estos clientes visualizando una capa procedente del *proxy* WMSCwrapper.

Pasarela KML para Google Earth

Para la visualización de una capa expuesta por la caché con el cliente Google Earth hay que realizar una petición HTTP como la mostrada a continuación:

```
'http://<host>:<port>/WMS_C_wrapper/<layer_name>.kmz'
```

Como respuesta se devuelve un documento KMZ (KML comprimido) interpretable por el cliente Google Earth. El documento KML contiene un elemento <NetworkLink>, que contiene, a su vez, un elemento <Link> mediante el cual se solicita el contenido que se desea visualizar, indicado a través del elemento <viewFormat>. El contenido se actualiza periódicamente, o cada vez que se detiene la navegación. Al recibir las peticiones de los <Link> se devuelve un fichero KMZ generado dinámicamente, que contiene una matriz de elementos <GroundOverlay>, cada uno de los cuales contiene una única petición conforme al estándar WMS-C. Ver Figura 10(c).

Google Maps

Google Maps utiliza una proyección esférica mercator, por lo que la capa a visualizar debe estar disponible en el sistema de coordenadas denominado EPSG:900913 (muy similar al EPSG:3857). En la Tabla 1 se muestra el código JavaScript necesario para superponer una capa procedente de la caché WMS_C_wrapper[1].

Tabla 1: Código JavaScript para superponer en Google Maps una capa accesible a través de WMSCwrapper

```
var tilelayer = new GTileLayer(<copyrights>, <minResolution>, <maxResolution>,
{
  tileUrlTemplate: 'http://<host>:<port>/WMS_C_wrapper/gm?
                    layers=<layer_name>&zoom={Z}&x={X}&y={Y}&format=<format>',
  isPng:<{true/false}>,
  opacity:<opacity>
} );
```

Los índices X, Y, Z anteriores son sustituidos por el cliente de Google por los valores apropiados de coordenadas latitudinal, longitudinal, y nivel de zoom, respectivamente, según la localización a visualizar en el mapa.

El esquema de teselado utilizado por Google Maps (ver Figura 9Figura 9) tan sólo difiere del utilizado internamente por WMS_C_wrapper en la inversión del índice Y, por lo que la traducción es inmediata: $Y_{WMSCWrapper} = MaxY - Y_{GoogleMaps}$.

[1] Información adicional en la documentación del API de Google Maps: http://code.google.com/intl/es-ES/apis/maps/documentation/reference.html#GTileLayer

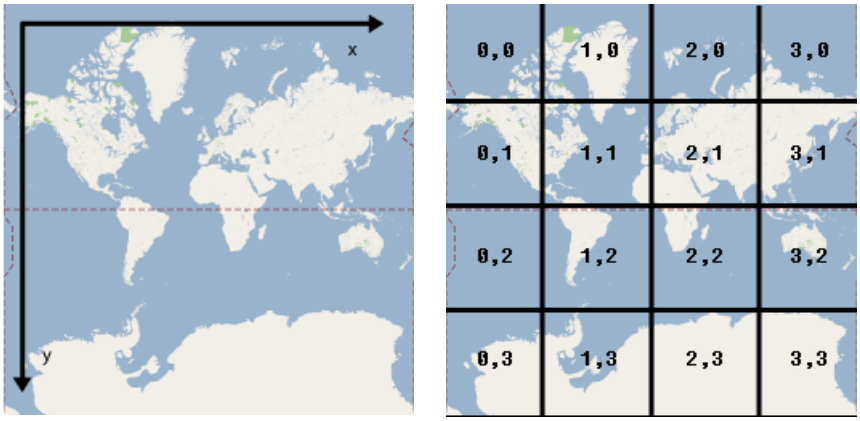

Figura 9: Esquema de teselado empleado en Google Maps

Microsoft Bing Maps

Bing Maps utiliza la misma proyección que Google Maps por lo que se requiere que, al igual que para este último, la capa esté disponible en esta proyección.

Tabla 2: Código JavaScript para superponer en Bing Maps una capa accesible a través de WMSCwrapper

```
var map = new VEMap('myMap');
var tileSourceSpec =
    new VETileSourceSpecification(
      'TITLE_OF_LAYER',

'http://<host>:<port>/WMS_C_wrapper/ve?quadkey=%4&format=image/png&layers=<lay
er_name>'
    );
tileSourceSpec.Opacity = 0.5;
map.AddTileLayer(tileSourceSpec, true);

...

<body onload="GetMap();">
```

Para optimizar el indexado y almacenamiento de las teselas, las coordenadas XY de una tesela se combinan en una cadena uni-dimensional denominada "*quadkey*". Cada *quadkey* identifica una tesela en un determinado nivel de resolución. Para convertir de coordenadas XY a *quadkey* hay que intercalar los *bits* de ambas coordenadas, y el resultado se interpreta como un número expresado en base-4. Por ejemplo, dada una tesela con coordenadas XY de (3, 5) para un nivel de resolución 3, el quadkey correspondiente se obtiene de la forma siguiente:

$X = 3 = 011_2$

$Y = 5 = 101_2$

quadkey = $100111_2 = 213_4$ = "213"

La longitud de un *quadkey* (número de digitos) corresponde al nivel de resolución de la tesela correspondiente. Por otra parte, el *quadkey* de una tesela comienza con aquel correspondiente de la tesela que lo contiene en el nivel de resolución inmediatamente anterior [7].

Al recibir una petición procedente de Bing Maps, el *proxy* WMSCWrapper traduce el *quadkey* recibido en las coordenadas XY correspondientes.

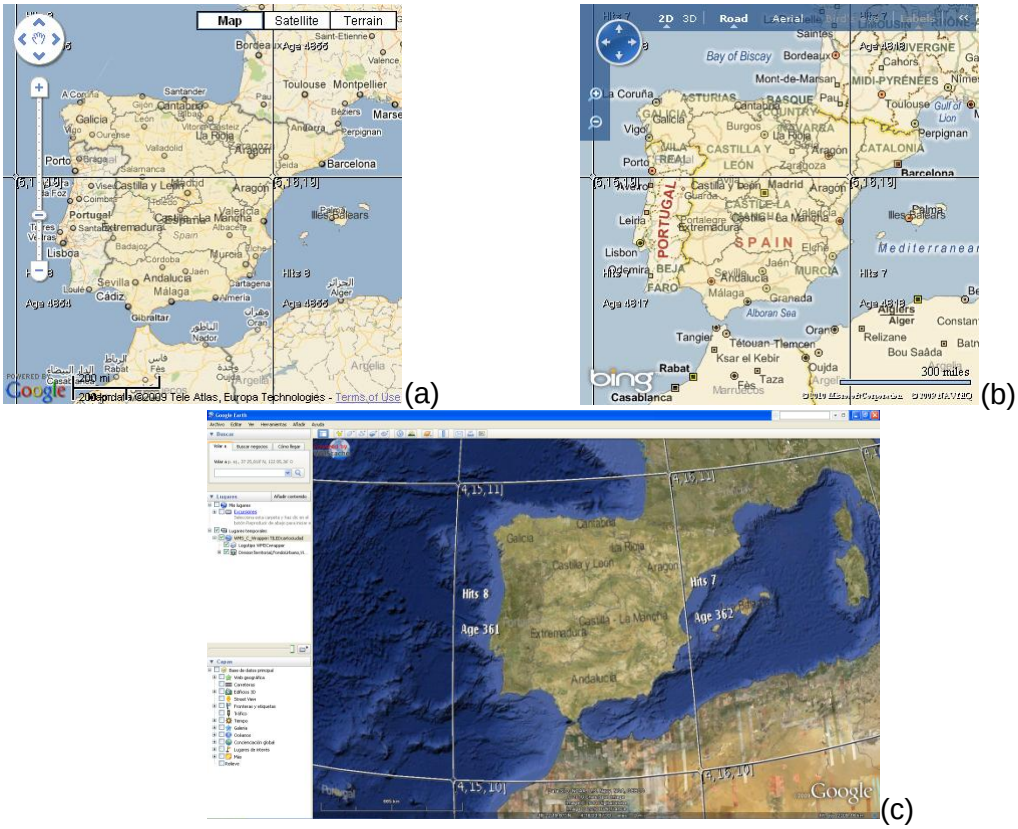

Figura 10: Visualización de una capa a través del *proxy* WMSCwrapper en los clientes (a) Google Maps, (b) Bing Maps, (c) Google Earth

CONCLUSIONES

Se ha presentado un *proxy caché* WMS-C para la aceleración de los servicios de mapas WMS distribuido como software libre. Está implementado íntegramente en Java, y su arquitectura, modular y flexible, permite la inclusión de componentes y sondas experimentales, resultando idóneo para experimentación con nuevas estrategias de caché.

Dispone de un índice espacial en memoria para la recolección de estadísticas, con implementaciones de persistencia de este índice tanto en fichero como en base de datos espacial. Este último caso es especialmente ventajoso para la monitorización continua del servicio y la toma de decisiones para conseguir una mejora en la Calidad de Servicio (QoS) ofrecida.

Ofrece un mecanismo más sofisticado que el de otras implementaciones estudiadas para la generación de *metatiles*, basado en la estrategia de mínima correlación con la cache. Mediante esta estrategia se consigue reducir el número de accesos a los almacenes externos de información geográfica.

El almacenamiento de las teselas de mapas se realiza de forma persistente en disco siguiendo una estructura de directorios con la que se consigue un almacenamiento eficiente (poca sobrecarga de directorios) y un buen tiempo de respuesta en la devolución de los objetos de la caché.

Por último, el sistema ofrece diversos interfaces de servicio, como el perfil WMS-C, pasarela KML para Google Earth, e interfaces REST para Google Maps y Microsoft Bing Maps.

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] OGC, "OpenGIS Web Map Service (WMS) Implementation Specification," *Open Geospatial Consortium* Available: http://www.opengeospatial.org/standards/wms.

[2] Lucian Plesea, "The Design, Implementation and operation of the JPL OnEarth WMS Server," *Geospatial Services and Applications for the Internet*, Sample, J.T., Shaw, K., Tu, S., y Abdelguerfi, M., eds., Berlin: Springer, 2008, págs. 93-109.

[3] Matt Mills, "NASA World Wind Tile Structure" Available: http://www.ceteranet.com/nww-tile-struct.pdf.

[4] NASA, "WorldWind Tile System Schema" Available: http://issues.worldwind.arc.nasa.gov/confluence/download/attachments/394/world+wind+tile+systemt.gif.

[5] K.P. Přidal, "Tiles à la Google Maps: Coordinates, Tile Bounds and Projection - conversion to EPSG:900913 (EPSG:3785) and EPSG:4326 (WGS84)," *Maptiler* Available: http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/.

[6] OSGeo, "Tile Map Service Specification," *Tile Map Service Specification - OSGeo Wiki* Available: http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification.

[7] J. Schwartz, "Bing Maps Tile System," *Microsoft Developer network* Available: http://msdn.microsoft.com/en-us/library/bb259689.aspx.

[8] OSGeo, "WMS Tiling Client Recommendation - OSGeo Wiki" Available: http://wiki.osgeo.org/wiki/WMS_Tiling_Client_Recommendation.

[9] J. Masó, N. Julià, y X. Pons, "Historia y estado actual del futuro estándar Web Map Tiling Service del OGC."

[10] J. Masó y N. Julià, *OpenGIS® Tiled WMS Discussion Paper 0.3*, OpenGIS, 2007.

[11] OpenGeo, "GeoWebCache" Available: http://geowebcache.org/trac.

[12] MetaCarta, "TileCache, from MetaCarta Labs," *TileCache* Available: http://tilecache.org/.

[13] M. Abrams, C.R. Standridge, G. Abdulla, E.A. Fox, y S. Williams, "Removal policies in network caches for World-Wide Web documents," *ACM SIGCOMM Computer Communication Review*, vol. 26, 1996, págs. 293–305.

[14] OsGeo, "WMS Tile Caching," *WMS Tile Caching - OSGeo Wiki* Available: http://wiki.osgeo.org/wiki/WMS_Tile_Caching.