Mejoras en el rendimiento de la IDE

J. J. Rodrigo[1], Deepak Daswani[1],J. Rosales[2].

[1]Dpto. Ingeniería
Cartográfica de Canarias S.A. GRAFCAN
{jrodrigo,ddaswani}@grafcan.com

[2]Director Técnico
Cartográfica de Canarias S.A. GRAFCAN
jrosales@grafcan.com

Resumen

Este documento muestra las estrategias de optimización y mejoras de rendimiento desarrolladas durante la fase de implementación de la Infraestructura de Datos Espaciales de Canarias (IDECanarias).

Palabras clave: Rendimiento,optimización, mapserver, IDECanarias.

1 Introducción

Uno de los aspectos fundamentales en el diseño e implantación de la IDE de Canarias (IDECanarias[1]) ha sido el rendimiento del sistema, medido desde el punto de vista del usuario, es decir, el tiempo de respuesta que el usuario final percibe de los servicios de IDECanarias. Creemos que este ha sido históricamente un punto débil de las IDE´s y de los servicios OGC´s ya que se encuentran pocos ejemplos de sistemas funcionando con buenos rendimientos.

Desde IDECanarias creemos que los pilares fundamentales sobre los que debe sustentarse el diseño de una IDE para que sea útil y para atraer al mayor número de usuarios son :

- Rendimiento de los servicios (tiempo de respuesta percibido por el usuario)
- Disponibilidad de los servicios.
- Fiabilidad de los contenidos.

- Legibilidad (Presentación de los datos)

Por supuesto entendemos que la interoperabilidad y la interconectividad vienen dadas por la propia arquitectura de la IDE y por el cumplimiento de las normas y estándares del OGC.

Las principales estrategias seguidas para mejorar los rendimientos de la IDE las podemos subdividir en los siguientes grupos :

- Capa intermedia de cache
- Optimizaciones del servidor de mapas
- Preparación de los datos para publicación

2 Capa Intermedia de cache

Con la llegada de los navegadores basados en teselas como Google Maps o OpenLayers se ha abierto la posibilidad de cachear la información de los servidores de mapas siempre y cuando se cumplan una serie de requisitos básicos.

En su concepción inicial el estándar WMS[4] no contempla la posibilidad de cachear la información (actualmente existen recomendaciones).

Beneficios para el cliente :

- Velocidad de respuesta del sistema : Los datos ya están pregenerados y únicamente hay que enviarlos al cliente.

Beneficios para el servidor :

- Se reduce mucho el consumo de CPU ya que los mapas solo se generan una vez.

La cache no esta recomendada para aquellos servicios que presentan una alta frecuencia de actualización, por ejemplo servicios que se actualicen diariamente.
Los requisitos para poder implementar un motor de cache son:

- Tamaño de peticiones fijo (ancho y alto) En nuestro caso es de 256x256 pixels.

- Las peticiones se deben regir por un distribuidor.

Si cumplimos estos requisitos podremos aprovechar también el esquema de cache local con el que cuentan los actuales navegadores Web, de forma que realmente vamos a tener una estrategia con dos niveles de cache : cache local y cache en servidor.

2.1 Motor de cache

El motor de cache ha sido concebido y desarrollado como una capa intermedia y transparente entre el cliente y el servidor de mapas y puede ser habilitada o no para un determinado servicio. El motor de cache debe por tanto poder entender y parsear correctamente los parámetros de las cadenas WMS.

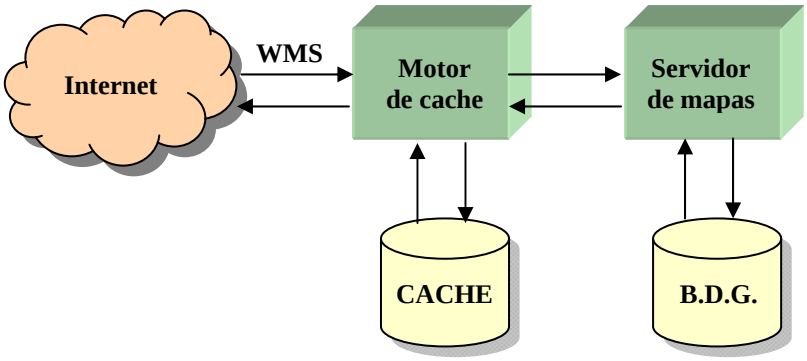

Figura 1 : Esquema del motor de cache

Funcionamiento esquemático del motor de cache :

Se intercepta la petición WMS del cliente.

Se comprueba si corresponde con una tesela (se verifica ancho y alto y resto de parámetros WMS).

Si no corresponde con una tesela reenviamos petición al servidor de mapas

Si corresponde con una tesela se busca si ya existe esa tesela en la cache

Si existe en la cache se lee el fichero y se envia al cliente

Si no existe solicitamos el fichero al servidor de mapas, se guarda en la cache y se envía al cliente

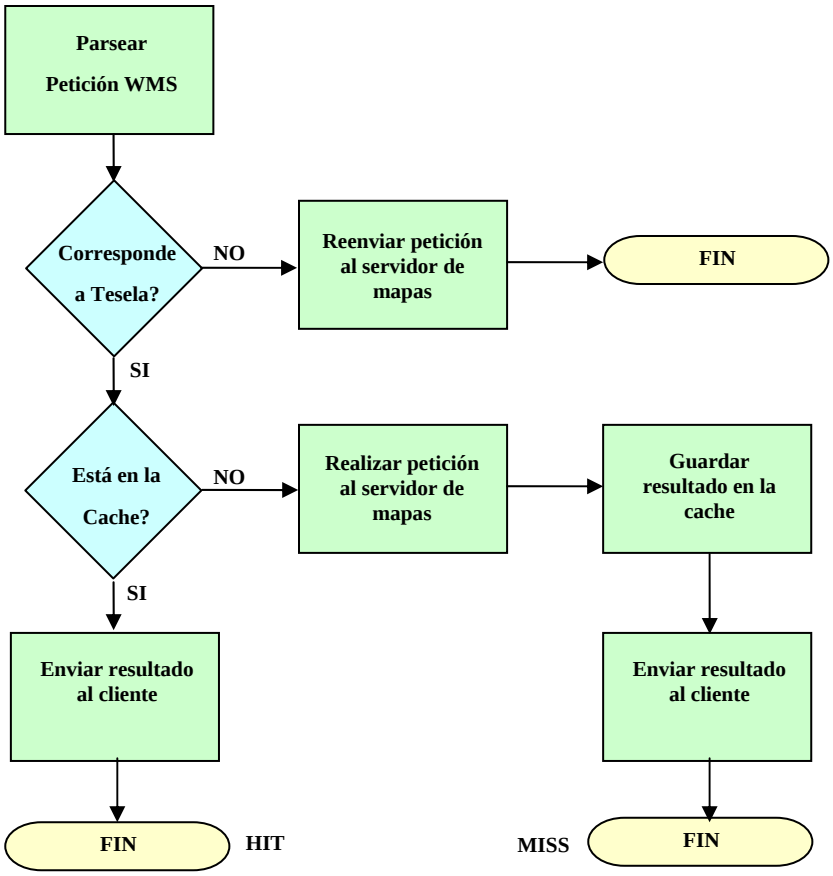

Figura 2 : Diagrama de flujo del motor de cache

El motor de cache también permite ir registrando el funcionamiento de la cache de forma que en todo momento podemos saber el porcentaje de éxito en la cache (HITS) que son aquellas peticiones que ya estaban en la cache y se han servido directamente desde la misma y el porcentaje de fallos (MISS) que son aquellas peticiones que no estaban en la cache y se han tenido que generar llamando al servidor de mapas para después ser enviadas al cliente. Lo deseable es que el porcentaje de HITS se sitúe por encima del 80-85% de las peticiones hechas a la cache para conseguir una mejora notable en el rendimiento del sistema. Las mejoras en los rendimientos las podemos expresar de la siguiente forma :

```
tiempo_medio_cache=(porcentaje_HITS*tm_HIT)+(porcentaje_MISS*tm_MISS)
tm_HIT = tiempo_medio_HIT
tm_MISS = tiempo_medio_MISS
SpeedUp_cache = (tiempo_servidor_mapas)/(tiempo_medio_cache)
```

Información de Acceso	
Número de Accesos:	9.333.606
Hits en Caché:	8.315.634
Miss en Caché:	1.677.316
Redirecciones en Caché:	61.174
Tiempo Medio de Hits:	0,08640 sg.
Tiempo Medio de Miss:	0,55320 sg.
Aciertos en Caché:	82.7086874998 %
Tráfico generado:	84.955,61 Mbytes. (82,96 Gb.)

Figura 3 : Información de la cache recogida en el Monitor de la IDE

También cabe destacar que las respuestas que genera el motor de cache tienen una cabecera HTTP[5] modificada para permitir que el cliente Web perciba los resultados como cacheables y que se pueda también hacer uso del esquema de cache local

2.2 Generador de cache

Como se puede comprobar el motor de cache expuesto corresponde con una estrategia de “cache bajo demanda” es decir, no existe la necesidad de que la cache

este rellena, ya que las propias peticiones de los usuarios van a ir rellenando la misma. Sin embargo se ha desarrollado también un módulo de “precacheo” o “generador de cache” que permite rellenar la cache a priori mejorando así los tiempos de respuesta desde el inicio de la publicación de un nuevo servicio. Este módulo de generador de cache realmente es un modulo de generador de peticiones WMS que aprovechan el propio motor de la cache para ir rellenando los ficheros de la misma.

En la práctica cuando se publica un nuevo servicio en la IDE se emplea una estrategia mixta de “precacheo” y “cache bajo demanda” de forma que se genera la cache de los niveles de zoom más visitados y el resto de la cache la irán rellenando los usuarios cuando realicen peticiones en ámbitos geográficos que aún no hayan sido cacheados.

GRAFCAN dispone de un monitor IDE que permite visualizar de forma gráfica las zonas y los niveles de zoom que son más consumidos por los usuarios. Con esta información se generan los ficheros que definen que zonas y niveles de zoom se van a precachear.

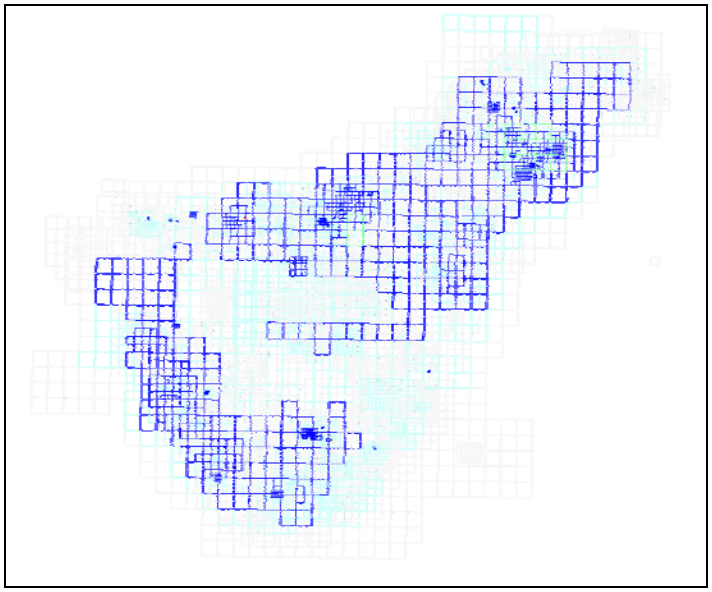

Figura 4 : Estudio de peticiones de los usuarios para selección de zonas de precacheo

3 Optimizaciones de Mapserver

UMN Mapserver[2] es uno de los servidores de mapas de software libre con mayor grado de implantación en el ámbito de las Infraestructuras de Datos Espaciales. Partiendo de la distribución estándar de Mapserver se han realizado modificaciones en su configuración y código fuente para mejorar su rendimiento y adaptarlo a los requisitos funcionales y de arquitectura de sistemas de IDECanarias.

Las principales mejoras introducidas a Mapserver son :

Corrección de errores :

Corrección de un error en el getFeatureInfo (reportado : Ticket 842).

Mejoras generales :

Recompilación del código empleando las directivas de optimización de velocidad del compilador tanto en el proyecto Mapserver como en las librerías externas.

Posibilidad de controlar los formatos de salida para cada servicio (para evitar peticiones en formatos muy pesados)

Posibilidad de publicar múltiples capas de información bajo una misma capa del servicio WMS, permitiendo de esta forma realizar generalizaciones por escala transparentes al usuario.

El parámetro STYLE en la llamada WMS se ha dejado como opcional (compatibilidad con determinados clientes WMS).

El Parámetro FORMAT en la llamada GetFeatureInfo del WMS pasa a ser opcional (Compatibilidad con cliente ArcGis).

Mejoras para el consumo por teselas :

Modificación en la renderización de los PIXMAP (símbolos raster) para que no se corten en los bordes de las teselas.

Modificación en el motor de *labeling* para que los textos se desplacen hacia el interior de las teselas permitiendo su renderizado sin tener que perder textos.

Mejoras de configuración :

Adaptación y reducción del fichero de proyecciones para mejorar el rendimiento. (Corregido ya en la nueva versión 5.2 de Mapserver)

Mejoras en la depuración :

Se han ampliado las capacidades de depuración del mapserver mostrando mayor cantidad de información en la misma, permitiendo así realizar unas optimizaciones de los datos más exhaustivas.

Sistema Operativo :

Se realizaron pruebas de rendimiento tanto en entorno Linux como Windows y se concluyó que el servidor de mapas es más eficiente sobre una plataforma Linux. En nuestro caso se ha montado sobre una distribución *Debian Etch 4.0*. Sin embargo la implementación de la librería de ECW dentro del proyecto GDAL para Linux es bastante deficiente por lo que los servicios Raster basados en ficheros ECW se han montado sobre una plataforma *Windows 2003 Server*.

Cada una las mejoras introducidas ha sido testeada usando un conjunto de peticiones predefinido y realizando pruebas de carga sobre el servidor con la herramienta JMeter[3].

4 Preparación de los datos

Otro de los apartados en los que se debe poner especial esfuerzo y que más repercusión tiene en los rendimientos finales es la correcta preparación de los datos para su publicación en la IDE. La preparación de datos resulta especialmente importante en capas vectoriales con gran número de elementos que pueden suponer una sobrecarga importante para los servidores de mapas.

Podemos resumir las estrategias seguidas durante la fase de preparación de datos en los siguientes puntos :

- *Estructuración de las capas de información en el servicio.*
- *Elección correcta de simbologías.*
- *Generación de índices espaciales a los datos.*
- *Generalización de los datos:*
    - Generalización de geometrías (líneas y polígonos).
    - Generalización mediante técnicas raster.
- *Mediciones de tiempos con las opciones de debug del servidor de mapas.*

Cada servicio que se va a publicar en la IDE debe de pasar una etapa de depuración donde se calculan los tiempos de respuesta de una serie de peticiones a diferentes escalas. Si los tiempos no son buenos se comprueba la información de debug de cada una de las capas del servicio para detectar donde está el problema y se procede a su optimización que puede consistir en una revisión del servicio, estructuración de capas o generalización de las mismas. Esto es un proceso iterativo que se repite hasta que el servicio supera los requerimientos de calidad para ser publicado en la IDE.

5 Conclusiones

Este documento muestra de forma resumida las experiencias y estrategias seguidas durante el proceso de optimización de rendimientos en IDECanarias. La implantación de un motor de cache intermedio supone un salto cuantitativo en rendimiento para aquellos usuarios que empleen clientes basados en teselas. Por otro lado las optimizaciones en el servidor de mapas y en la preparación de los datos son imprescindibles para alcanzar un alto grado de calidad en los servicios de la IDE.

Referencias

[1] IDECanarias, http://www.idecan.grafcan.es

[2] Mapserver Project homepage, http://mapserver.gis.umn.edu/

[3] Apache JMeter, The Apache Jakarta Project, http://jakarta.apache.org/jmeter/

[4] Web Map Service (WMS), http://www.opengeospatial.org/standards/wms

[5] HTTP 1.1 Header Definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html