Recerca de l'eficiència en aplicacions SIG per Android.

Rafael Bermúdez Guijo

20 de Gener de 2011

1) Resum

Degut a l'increment del número de consumidors de dades geogràfiques, sobretot de dispositius mòbils, els sistemes d'informació geogràfica es troben davant el repte de reduir el coll d'ampolla que suposa l'arquitectura client-servidor clàssica.

Una de les solucions a aquest problema és la translació de part de la intel·ligència al node client, així com la creació de xarxes descentralitzades (peer to peer),

L'objectiu d'aquesta recerca és demostrar la viabilitat d'una infraestructura client mòbil-servidor, on el client no és únicament un consumidor més, si no que es torna un node intel·ligent.

Per fer aquest estudi s'ha desenvolupat una aplicació pel sistema operatiu Android que consumeix dades d'OpenStreetMap. Aquesta aplicació utilitza tècniques de tessel·lat, catching i descàrrega de dades en background segons la posició de l'usuari, per facilitar el consum i reduir el flux de dades intercanviades entre el client i el servidor. També s'ha creat un servei web intern al dispositiu mòbil per a la creació de xarxes peer to peer, les quals permetin un intercanvi de dades entre els terminals mòbils.

En aquest treball s'ha demostrat la viabilitat de la infraestructura a través del impacte de l'ús de les tècniques comentades anteriorment sobre el dispositiu client. Per mesurar l'impacte s'ha tingut en compte la càrrega de la CPU (la qual repercuteix al consum de la bateria) i el temps de resposta del sistema.

Els resultats de les proves realitzades indiquen que aquestes tècniques redueixen el temps de cerca de punts d'interès d'una manera dràstica, però que també tenen un alt impacte a la CPU i en el temps de càrrega, sobretot en dispositius amb menys capacitats.

Paraules clau: SIG, TMS, WMS, OSM, P2P, Android

2) Introducció

En els darrers anys podem veure dues tendències en el món dels sistemes d'informació geogràfica [3], la primera és l'augment de les infraestructures de dades espacials (IDE) que donen serveis web geoespacials (serveis OGC) i la segona és l'augment i la popularització dels SIG mòbils, els quals exerceixen de clients dels IDE.

En l'actualitat existeixen un llarg llistat d'aplicacions SIG mòbils desenvolupades pels diferents sistemes operatius [7], especialment per Windows Mobile, però el creixement d'aplicacions per Android està sent exponencial.

De les aplicacions SIG mòbils d'Android existents s'ha de destacar gvSIG mini [8] la qual té les següents característiques:

- Navegació per mapa: Ens permet visualitzar mapes dels WMS més coneguts, com OpenStreetMap.
- Gestió de mapes: Visualització de capes del mapa segons desig de l'usuari.
- Gestió de punts de ruta: Afegir o eliminar punts en una ruta definida
- Localització per GPS: Si el dispositiu on s'executa té receptor de GPS permet localitzar al usuari en el mapa
- Client Name finder: Permet buscar en OpenStreetMap per nom del recurs sol·licitat.
- Client de rutes: Càlcul de rutes mitjançant la definició del punt d'inici i punt final

Tenint en compte aquest augment d'aplicacions, els sistemes d'informació geogràfica es troben davant el repte de reduir el coll d'ampolla que suposa el consum d'aquests serveis per un gran número de clients mòbils [6] i d'augmentar l'eficiència de l'arquitectura per no haver d'incrementar la quantitat de servidors d'una manera lineal en relació al número de clients.

Dia a dia els dispositius mòbils van evolucionant, millorant, entre d'altres aspectes, les seves capacitats de processament, però les aplicacions SIG mantenen les bases de funcionament inalterables, una arquitectura client-servidor clàssica, on el client es tractat com un dispositiu amb poques capacitats de càlcul que només es dedica a la representació gràfica de les dades rebudes.

Amb aquesta arquitectura, la potencia del client està totalment desaprofitada, a més de necessitar més potencia en el servidor i augmentar el volum de trànsit de dades entre el client i el servidor, degut a que qualsevol petició ha de ser enviada pel client, rebuda pel servidor, processada i retornada cap al client.

Una de les tècniques més utilitzades per reduir el flux

de dades és l'emmagatzemament de dades en la memòria cau (tècnica de catching), que en fan ús els clients de servidors de mapes (WMS) del tipus WMS-C [9], els quals emmagatzemen les tessel·les en la memòria del dispositiu. A part de reduir la quantitat de dades transmesa i el número de peticions que haurà de processar el servidor, s'aconsegueix millorar la fluïdesa de la visualització dels mapes.

WMS-C és un exemple d'emmagatzemament d'imatges per un ús posterior, però si es desitja guardar dades més complexes i una administració d'aquestes més avançada s'utilitzarien bases de dades locals (en el propi dispositiu). Degut a la impossibilitat i a la ineficiència de tenir totes les dades guardades en un dispositiu mòbil, [11] proposa l'ús de bases de dades dinàmiques que continguin les dades geogràfiques locals que s'actualitzen segons la posició de l'usuari. La implementació d'aquesta solució aconsegueix reduir el temps de resposta a un 30% respecte al consum puntual cap al servidor.

Les tècniques d'emmagatzemament en memòria cau augmenten el número de clients que pot tenir un servidor, però no eviten el coll d'ampolla, ja que continua sent una infraestructura centralitzada, Per donar escalabilitat als serveis de les infraestructures de dades espacials es proposa l'ús d'infraestructures distribuïdes com xarxes peer to peer (P2P) [4][9][10], en les quals el client també exerceix de servidor.

Degut a la complexitat de compartició de dades geogràfiques, és necessari que els terminals mòbils realitzin un processat de la petició i donin una resposta adequada, per realitzar aquesta feina és necessari un servidor web contingut al terminal. L'augment de la potencia dels terminals mòbils existents al mercat i l'aparició de sistemes operatius amb capacitats avançades han fet que sigui possible contenir un servidor web en un terminal mòbil, però aquest suposa un cost computacional alt per al dispositiu [5], encara que pot ser reduït amb l'ús d'interfícies de comunicació de tipus REST en comptes de SOAP [1].

La naturalesa dels dispositius mòbils, els quals canvien de xarxa constantment, fa que el descobriment dels servidors web mòbils existents en la xarxa P2P sigui especialment complicat [12], per aconseguir que es puguin intercanviar dades entre terminals mòbils és necessari un node, conegut com RendezVous, el qual s'encarregarà de mantenir actualitzada les direccions IP dels serveis web mòbils [13].

Un node RendezVous en una xarxa P2P és una màquina amb visibilitat pública i IP estàtica (o registrada en un servidor de DNS) on es pot trobar el llistat de peers (usuaris) que estan connectats a la xarxa, d'aquesta forma es torna en el punt d'entrada.
El node RendezVous s'encarrega de mantenir actualitzat el llistat dels usuaris i en cas de canvi enviar-la als usuaris que estiguin connectats.
Els terminals mòbils han evolucionat fins un punt de poder-se qualificar com ordinadors de butxaca, però el parc de dispositius és molt heterogeni, tant a nivell de capacitat de processament, com de resolució de pantalla, qualitat de connexió, etc. Per aquest motiu [3] proposa l'ús d'una capa intermèdia que adapti les dades entre el servidor i el client mòbil on seran consumides.

Android

En aquest projecte ens centrarem en el sistema operatiu mòbil Android [2], a continuació es fa un petit anàlisis de les capacitats que ens ofereix Android pel desenvolupament d'aplicacions SIG.

Les aplicacions Android es desenvolupen amb el llenguatge de programació Java, aquest fet fa que els desenvolupadors puguin fer servir llibreries existents per aquest llenguatge en les seves aplicacions, però sempre s'ha de tenir en compte que es tracta d'un dispositiu amb capacitats de procés i emmagatzematge limitades i que per aprofitar el potencial d'Android s'ha d'intentar aprofitar al màxim les llibreries natives del sistema operatiu.

Les llibreries d'Android varien i s'amplien segons la versió del S.O. en la qual es desenvolupi, per tant a l'hora de crear l'aplicació s'ha de tenir en compte el parc de dispositius existent i els S.O. predominant.

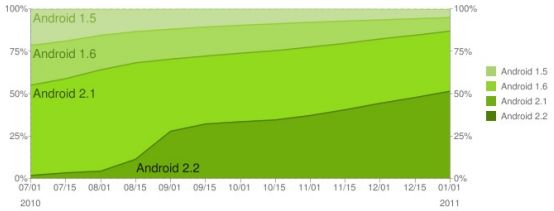

Figura 1. Tendència versió Android (font: www.Android.com)

En la gràfica anterior es posa de relleu que la versió més utilitzada d'Android a dia d'avui és la 2.2, seguida de la 2.1, conjuntament cobreixen més del 80% del mercat. Amb aquesta premissa, i tenint en compte la seva característica de retrocompatibilitat, sabem que si desenvolupem amb la versió 2.1 arribarem a un gran ventall d'usuaris.

Gràfics a Android

Android disposa de llibreries per pintar els principals formats d'imatges de tipus raster (GIF, JPG, PNG ...), però no conté cap llibreria de rasterització de SVG ni cap lector de format vectorial.

La llibreria de gràfics (android.graphics), a més, permet el pintat de figures primitives, com punts, ratlles, esferes, quadrats, entre d'altres, en les aplicacions Android.

Emmagatzematge de dades

Des de la API d'Android es pot llegir i escriure de forma permanent dades tant en la memòria interna del dispositiu com a l'externa (si en té). A més, es

permet la creació, consulta i escriptura de bases de dades SQLite.

Mapes en Android
Android permet als desenvolupadors utilitzar una vista de mapes (eix de coordinades) sobre la qual es sol renderitzar Google Maps, però pot ser utilitzada per renderitzar altres gràfics, com imatges o dades vectorials.

La llibreria de Google Maps descarrega, renderitza i emmagatzema en la memòria cau les tessel·les del mapa de Google.

La API d'Android aporta algunes funcionalitats geogràfiques, com l'adquisició de la posició i la velocitat del dispositiu (a partir de GPS o triangulació d'antenes terrestres) i el càlcul de distància entre punts.

Formats d'intercanvi de dades
Android conté llibreries tant per la lectura de XML (DOM i SAX) com JSON.

Components d'una aplicació Android
Existeixen quatre tipus de components d'una aplicació:

- *Activitats*: Component visual de l'aplicació, equivalent a una plana en una aplicació web, una aplicació pot estar formada per una o varies activitats.
- *Serveis*: Component que corre en background per un temps indefinit, dedicat a tasques que no tenen part visual, com pot ser la reproducció de música o càlculs complexos.
- *Broadcast Receiver*: Component dedicat a rebre canvis en l'estat del telèfon, com l'estat de la bateria, canvi de zona horària, entre d'altres.
- *Content Provider*: Component que posa a disposició d'altres aplicacions dades, com els contactes existents al telèfon.

3) Aplicació pilot

S'ha desenvolupat una aplicació anomenada “OSM as you go” que descarrega les dades geogràfiques que envolten a l'usuari del servidor de OpenStreetMap i permet que aquest les consumeixi mitjançant una interfície que es recolza en la vista de mapes d'Android i la llibreria de Google Maps, heretant la seva facilitat d'ús.

Les característiques de l'aplicació són:

- Visualització de mapes servits per WMS o TMS.
- Visualització de dades vectorials de OSM.
- Descàrrega de mapes de WMS o TMS de l'àrea d'interès de l'usuari segons camina.
- Descàrrega de dades de OSM de l'àrea d'interès de l'usuari en format XML o JSON.
- Llistat de les dades geogràfiques de l'àrea d'interès.
- Cercador per nom de les dades geogràfiques de l'àrea d'interès.

Visualització de mapes servits per WMS o TMS
Es visualitzaran les tessel·les del servidor de mapes que hagi triat l'usuari.

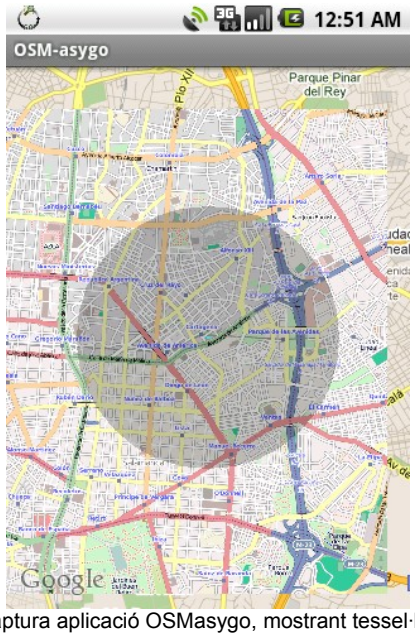

Figura 2.Captura aplicació OSMasygo, mostrant tessel·les de OSM i area d'interés (font: pròpia)

Visualització de dades vectorials de OSM
Mitjançant l'ús de les figures primitives (punts i línies), es pinten les dades rebudes de OSM.

Figura 3. Captura aplicació OSMasygo, mostrant carreteres i àrea d'interès (font: pròpia)

Descàrrega de mapes de WMS o TMS de l'àrea d'interès de l'usuari
Segons el dispositiu canviï de posició, es descarreguen i s'emmagatzemen en la targeta de memòria les tessel·les que formen la zona d'interès als diferents nivells de zoom.

Descàrrega de dades de OSM de l'àrea d'interès de l'usuari en format XML o JSON
L'aplicació descarregarà les dades del servidor OSM en format XML o d'un proxy JSON i les emmagatzemarà en una base de dades SQLite.

Llistat de les dades geogràfiques de l'àrea d'interès
S'endrecen i es llisten totes les entitats geogràfiques que formen part del radi d'interès de l'usuari.

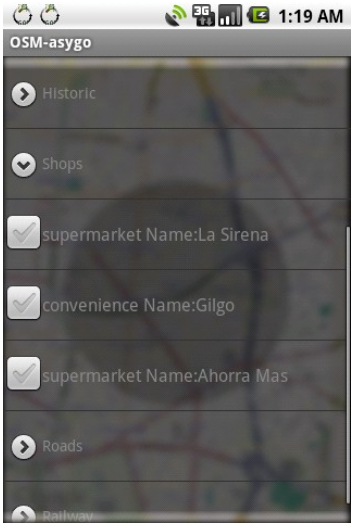

Figura 4.Captura aplicació OSMasygo, mostrant llistat de entitats geogràfiques descarregades de OSM (font: pròpia)

Cercador per nom de les dades geogràfiques de l'àrea d'interès
Permet a l'usuari buscar de manera immediata i amb ajudes SayT (Search-As-You-Type) qualsevol tipus d'entitat que formi part de la zona d'interès.

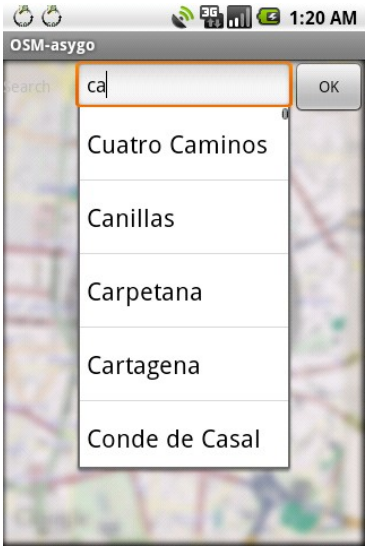

Figura 5.Captura aplicació OSMasygo, pantalla cercador amb valors trobats (font: pròpia)

IMPLEMENTACIÓ

Adquisició de les dades

Una de les claus del desenvolupament és el tessel·lat del món. Aquesta tècnica permet composar el món a partir de petits fitxers, els quals són fàcils d'administrar pel sistema.

L'aplicació de la tècnica de tessel·lat és ben coneguda en la visualització de mapes, on es pot trobar un servei dedicat per aquest fet, anomenat Tile Map Service.

Tile Map Service (TMS) és un estàndard definit per OSGeo (Open Source Geospatial Foundation) per a l'emmagatzematge i l'adquisició de tessel·les de mapes prerenderitzades a diferents nivells de zoom. Aquest estàndard únicament defineix quina ha de ser la divisió del mapa (tessel·lat) i com es posa a disposició de l'usuari (interfície REST).

Tot WMS pot ser transformat en TMS mitjançant l'acompliment de les transformacions de coordinades definides per l'estàndard (figura 6).

Aquesta transformació redueix flexibilitat a l'usuari, ja que nomes deixa a la seva decisió quina tessel·la vol adquirir (depenent del zoom i les coordinades desitjades), la qual ja té una grandària, un sistema de coordenades, unes capes geogràfiques i un format de sortida fixat. Però el fet de tractar-se d'imatges prerenderitzades redueix el cost computacional del servidor, ja que únicament s'encarrega de retornar al client l'arxiu ja generat, i facilita la seva compartició.

Aquesta tècnica facilita la consulta, l'emmagatzematge i compartició d'imatges descarregades d'un servidor de mapes del tipus WMS entre usuaris.

A continuació podem veure la simplificació de TMS respecte WMS.

Exemple de consulta a WMS:

http://www2.demis.nl/wms/wms.asp?Service=WMS&WMS=BlueMarble&Version=1.1.0&Request=GetMap&BBox=0.0,0.0,180.0,85.05112882311167&SRS=EPSG:4326&Width=256&Height=256&Layers=Earth%20Image,Borders,Coastlines&Format=image/gif

On:

- BBox: Marc de coordenades (minX,minY,maxX,maxY).
- SRS: Sistema de coordenades.
- Width: Amplada de la imatge de sortida.
- Height: Altura de la imatge de sortida.
- Layers: Capes geogràfiques que contindrà la imatge de sortida.
- Format: Format d'imatge.

Exemple de consulta a TMS:

http://a.tile.openstreetmap.org/z/x/y

On:

- z: Nivell de zoom.
- x: Posició x de la tessel·la.
- y: Posició y de la tessel·la.

$$lon = (xtile / 2^{zoom} * 360) - 180$$

$$lat = \arctan(\sinh((\pi * [1 - (2 * ytile / 2^{zoom})])) * 180 / \pi$$

$$\text{Xtile} = \left\lfloor ([lon + 180] / 360) * (2^{zoom}) \right\rfloor$$

$$\text{ytile} = \left\lfloor \left(1 - \frac{\ln\left(\tan(lat / 180 * \pi) + \frac{1}{\cos(lat / 180 * \pi)}\right)}{\pi} \right) * 2^{(zoom-1)} \right\rfloor$$

$\lfloor \ \rfloor$ Partie entiere de la valeur

lat = latitude ; lon = longitude

Figura 6.Transformació entre coordenades i tessel·les i viceversa (font:http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames)

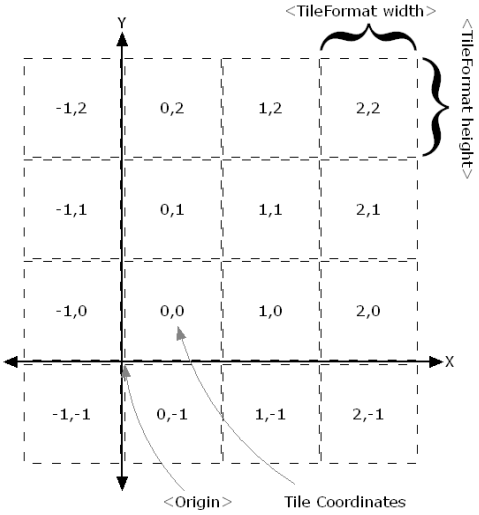

Figura 7.Exemple de transformació d'un eix de coordinades a tessel·les en format {x,y}.

Aquesta mateix tècnica pot ser utilitzada en qualsevol servei OGC on s'indiquen uns límits geogràfics (paràmetre BBOX), únicament és necessari una transformació de número de tessel·la a límits geogràfics i viceversa.

A la figura 8 podem veure un exemple de composició de dades geogràfiques vectorials a partir de dues tessel·les vectorials ({x,y} i {x,y+1}).

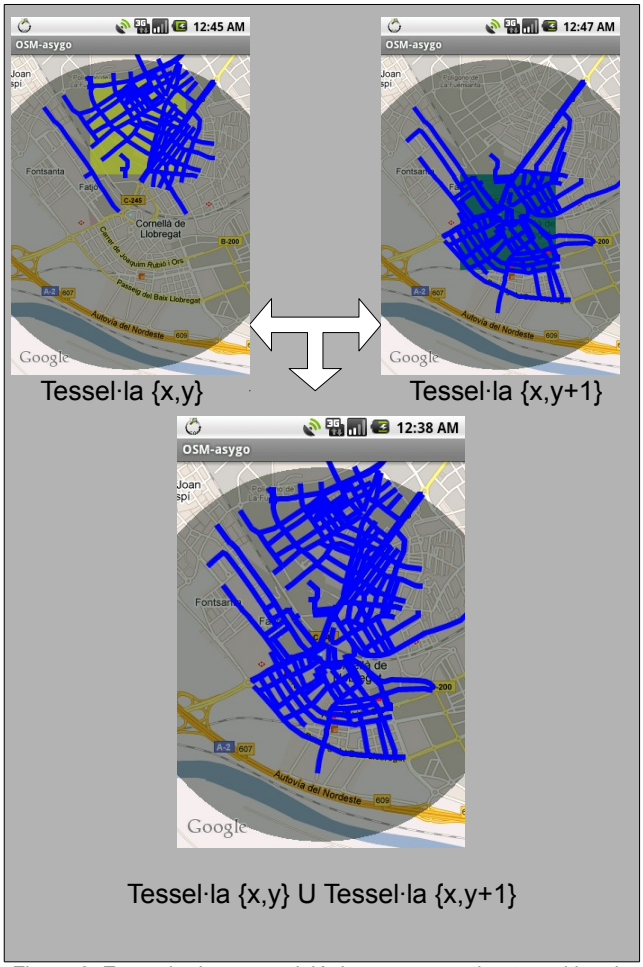

Figura 8. Exemple de composició de mapa a partir e tessel·les de dades vectorials.

Com s'ha comentat anteriorment, únicament es descarreguen les dades que envolten a l'usuari, el que anomenem zona d'interès. Per composar el mapa (imatge) de la zona d'interès de l'usuari s'utilitzarà la tècnica clàssica de tessel·lat, on les tessel·les varien segons el nivell de zoom del visionat (núm. de tessel·les=$2^{\text{nivell de zoom}}$). Amb aquesta tècnica, una zona d'interès definida amb un radi de 2 Km. i 18 nivells de zoom, estaria composada per **582 tessel·les**.

Per l'adquisició de dades vectorials d'OSM es fixarà el zoom al nivell 15, amb aquest nivell i un radi d'acció de 2km el mapa queda compost per **9 tessel·les**.

Figura 9.Tessel·les amb zoom 15 (font:pròpia)

El radi d'interès pot variar segons a desig de l'usuari, per aquest motiu s'ha generat una activitat Android (figura 10) que permet, entre d'altres configuracions, adaptar el radi d'interès (0<radi<2Km).

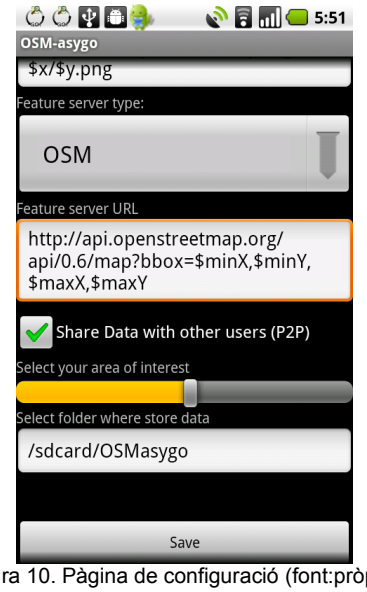

Figura 10. Pàgina de configuració (font:pròpia)

Degut a la necessitat que la descàrrega de dades sigui constant segons canvia de posició l'usuari, aquesta no pot corre en una activitat (Activity) Android, ja que el sistema operatiu para i mata l'aplicació quan no està visible i necessita els recursos del sistema per a una altra aplicació (més informació sobre el cicle de vida de les aplicacions a http://developer.android.com/guide/topics/fundamentals.html).

El component que més s'adapta a les necessitats és el tipus servei (Service) configurat com remot (remote), ja que estarà viu des de la primera vegada que l'usuari arrenca l'aplicació i no s'aturà fins que el dispositiu no es reinicii.

Emmagatzematge de les dades

Les dades descarregades, tant les tessel·les d'imatges com de dades de OSM, s'emmagatzemaran en la targeta de memòria del dispositiu (SD card), respectant els directoris de les tessel·les definits per l'estàndard TMS (estructura z/x/y).

Com s'ha comentat anteriorment el servei de descarrega està configurat com remot, Android no permet un intercanvi de dades directe entre les activitats de l'aplicació i el servei remot (únicament missatges, classe messeger), per aquest motiu és imprescindible la creació de bases de dades les quals emmagatzemen la configuració i els estats de l'aplicació, les quals són accessibles tant des del servei com des de les activitats de l'aplicació.

Aquestes bases de dades són les següents:

- Configuració: Conté la configuració de l'usuari per a l'aplicació (servidor WMS, o TMS, radi d'interès, servidor OSM, etc.).
- Tessel·les: Conté el llistat de les tessel·les, tant d'imatges com de dades geogràfiques, que conformen la zona d'interès de l'usuari. Aquesta base de dades s'actualitza segons l'usuari vagi canviant de posició.
    - La taula “tiles” conté la “x” i “y” de les tessel·les en tots el s nivells de zoom (“z”) que permet OSM (18 nivells).
    - Per a les tessel·les geogràfiques només és necessari emmagatzemar la “x” i “y”, ja que el nivell de zoom està fixat a 15.

La base de dades de tessel·les conté les següents taules:

Taula	Camp	Descripció
Tiles	z	Zoom en el qual s'ha de mostrar la tessel·la
	Y	Posició y de la tessel·la
	X	Posició x de la tessel·la
Data	Y	Posició y de la tessel·la de dades vectorial
	X	Posició x de la tessel·la de dades vectorial

Figura 11. Base de dades de tessel·les

La base dades de configuració conté les següents taules:

Taula	Camp	Descripció
config	myLastLocation	Darrera localització de l'usuari
	featureServerType	S'indica si es vol descarregar dades des de OSM, i en cas positiu s'indica si seran XML o JSON
	featureServerURL	URL del servidor on estan les dades OSM
	mapServerType	S'indica si es vol descarregar dades des de WMS o TMS
	mapServerURL	URL del servidor WMS o TMS
	SDFolder	Directori on s'emmagatzemen les dades geogràfiques
	Meters	Grandària del radi d'acció em metres
	Sharedata	Indica si es vol activar el servidor web local del dispositiu per crear una xarxa P2P

Figura 12 Base de dades de configuració

La decisió de crear la BD de configuració separada de la BD de tessel·les, és per evitar problemes de concurrència (Sqlite té baixa concurrència), ja que el servei sempre està viu, modificant la base de dades de tessel·les, mentrestant l'usuari podria modificar la configuració de la seva aplicació. Sqlite bloqueja la BD mentrestant s'està escrivint en ella, per tant si les dues bases de dades es fusionessin en una, l'usuari no podria guardar la seva configuració si el servei està modificant les tessel·les o viceversa.

A més d'aquestes dues bases de dades es crearà una base de dades per cadascuna de les tessel·les d'informació geogràfica descarregada d'OSM en format XML. L'estructura d'aquest tipus de base de dades copia l'estructura de la base de dades d'OpenStreetMap (http://wiki.openstreetmap.org/wiki/Database/Model), eliminant les taules no necessàries, quedant les taules de la següent manera:

Taula	Camp	Descripció
member	type	Tipus d'entitat (node o way)
	changeset	
	lat	latitud
	lon	longitud
	timestamp	Temps de creació
	Uid	Id universal
	User	usuari
	Version	Versió e l'element
	visible	Visibilitat de l'element
tags	K	Clau de la metadata
	V	Valor de la metadata
	member_id	Element de la metadata
wayelements	way_id	Id de del camí
	node_id	Id del node que conté el camí

Figura 13. Base de dades d'OSM

A continuació podem veure la relació entre la base de dades de l'aplicació i les taules de la base de dades d'OSM.

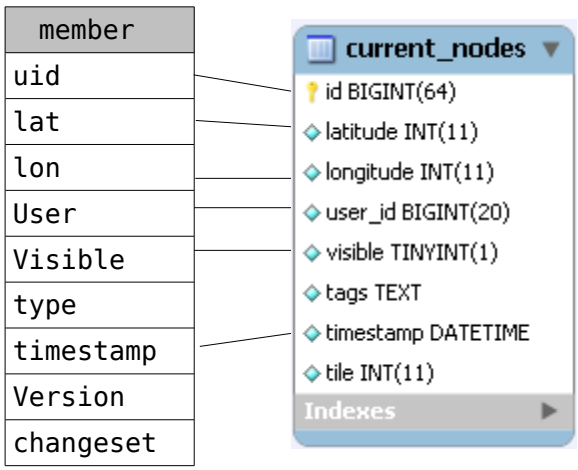

Figura 14. Relació de la taula que conté la informació de nodes de BD de l'aplicació i BD original d'OSM

Com s'observa, hi ha atributs de la taula de l'aplicació que no tenen relació amb la taula d'OSM original, com són type, version i changeset, però aquests si que són rebuts en l'XML d'OSM, com a propietat d'un node OSM, per aquest motiu són emmagatzemats en la base de dades de l'aplicació.

El camp tags de la taula current_nodes d'OSM no està reflectit en la taula member de l'aplicació, el motiu és que tota la informació de tags s'emmagatzemarà a la taula tags, on es relacionarà el uid de member amb el seu tag (valor k i v).

Per últim, el camp tile de la taula OSM no es conté a la taula member de l'aplicació, degut a que és un id més que no aporta cap informació a la base de dades de l'aplicació.

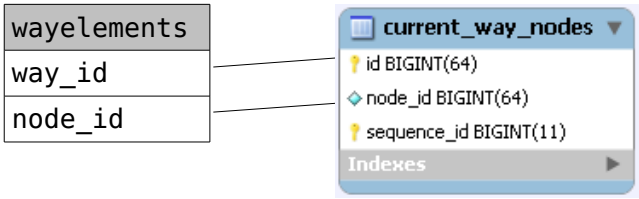

Figura 15. Relació de la taula que conté la informació de camins de BD de l'aplicació i BD original d'OSM

Podem veure que s'està emmagatzemant l'id del camí i dels nodes que el composen, però no es guarda sequence_id, degut a que aquest atribut és per relacions (no s'estan guardant les entitats del tipus relació).

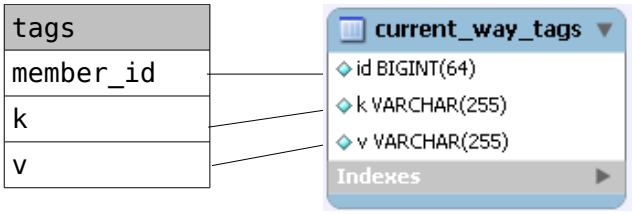

Figura 16. Relació de la taula que conté les metadades de BD de l'aplicació i BD original d'OSM

Els tags són les metadades d'una entitat geogràfica (member_id), en la base de dades d'OSM es separen els tags de les entitats del tipus way i les del tipus node, en canvi, en l'aplicació, la taula tags conté totes les metadades, tant de nodes (member) com de way's.

Pintat de dades

Per a la visualització del mapa, únicament es crearà una activitat Android amb una vista del tipus mapa, sobre la qual es pintaran les tessel·les descarregades de TMS o WMS.

Per al pintat de dades vectorials sobre el mapa es faran servir dos tipus de primitives de gràfics per Android:

- Node: primitiva de pintat android.graphics.Point.
- Way: primitiva de pintat android.graphics.Path.

Compartició de les dades

Guardant les dades respectant els directoris de les tessel·les definit a l'estàndard, i creant un servidor web al mòbil que apunta al directori base de guardat, es crearà automàticament una interfície REST que replicarà l'estructura d'un TMS, però aquest a més de tessel·les d'imatges, contrindrà dades vectorials i informació, el qual podria ser anomenat TFS (Tile Feature Server).

El servidor web del mòbil arrenca en el port 8080, per adquirir les tessel·les d'imatges la crida sería del tipus:

http://ip:8080/$z/$x/$y

$z = Nivell de zoom.

$x = Posició x de la tessel·la.

$y = Posició y de la tessel·la.

Si es volgués obtenir les dades vectorials la crida canviaria a:

http://ip:8080/osmdata/$tipusdedades/$x/$y

$tipusdedades = [1 per XML, 2 per JSON, 3 per bases de dades]

$x = Posició x de la tessel·la.

$y = Posició y de la tessel·la.

Com indica [9] és imprescindible un node de tipus Rendez-Vous en la xarxa de compartició de dades per a tenir les IP's dels dispositius actualitzades.Per aquesta prova pilot el node RendezVous ha sigut desenvolupat en llenguatge PHP i la seva única funció és rebre la IP de l'usuari i retornar les IP dels usuaris connectats.

Cada vegada que l'usuari canvia de xarxa l'aplicació Android rep un event i crida al servidor RendezVous passant-li com paràmetre la seva nova IP. D'aquesta forma s'aconsegueix que sempre estiguin actualitzades les IP's dels usuaris mòbils.

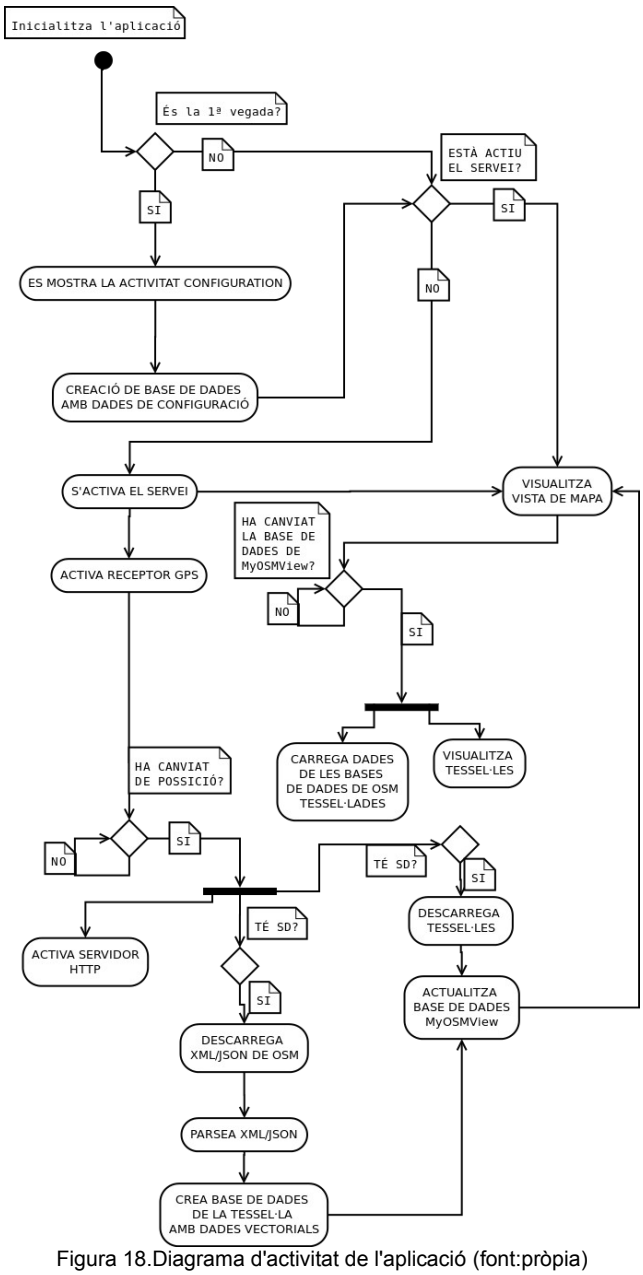

Figura 18.Diagrama d'activitat de l'aplicació (font:pròpia)

PROVES DE RENDIMENT

Per verificar la viabilitat del sistema proposat es faran proves de rendiment per mesurar l'impacte sobre el dispositiu mòbil. En aquestes proves es mesurarà la càrrega de la CPU i el temps de resposta del sistema. En dispositius mòbils és molt important no abusar de la CPU ja que té una repercussió directe sobre el consum de la bateria.

Les proves de rendiment s'han fet amb dos dispositius diferents, HTC Hero i Nexus One.

	HTC Hero	Nexus One
Versió	2.1	2.2.1
CPU	528Mhz	1Ghz.
RAM	288Mb	512Mb
Resolució	320x480 px.	480x800 px

Figura 19.Comparativa HTC Hero amb Nexus One

A continuació es detallen les proves realitzades en aquest projecte:

- Quin impacte té la descàrrega de totes les tessel·les (imatges) que conformen el mapa base de la zona on es troba l'usuari? Què aporta aquesta descàrrega?
- Com proposa [3], les dades han de ser adaptades al client mòbil per reduir el cost computacional del dispositiu client, per aquest motiu s'analitza, quin és el format més adient per intercanviar dades vectorials i amb metadades entre el servidor i el client (XML o JSON)? La plasmació d'aquestes dades en bases de dades és molt costosa?
- Quin és el cost temporal i computacional, per al dispositiu client, de descarregar les dades vectorials que conformen la seva zona d'interès?
- Millora per a l'usuari el fet de tenir totes les dades descarregades en el seu dispositiu? Aquesta prova es realitza mitjançant la comparació d'una cerca de punts d'interès en sistemes clàssics client-servidor, com GoogleMaps o NameFinder de OSM, i en el sistema implementat a la prova pilot.
- Quin cost computacional té el pintat de dades vectorials en dispositius Android?
- Quin cost computacional té la compartició de dades entre dispositius? Per a que es puguin compartir dades els clients han de tenir actiu un servidor HTTP, quantes consultes pot rebre?

4) Resultats

Descàrrega de Tessel·les de WMS i TMS segons canvia la posició

La descàrrega de l'àrea d'interès de 2 km, suposa la descàrrega de més de 500 tessel·les, que té una equivalència en bytes de més de 4Mb.

Amb una connexió ADSL de 6Mb, el temps de visualització entre la càrrega des de SD o des del servidor és inapreciable o menor de 3 segons, temps més que acceptable per a la visualització d'una tessel·la que pertany a un nivell de zoom molt alt.

Des d'un punt de vista individual, la càrrega constant de tessel·les en SD és útil únicament quan és sàpiga que el dispositiu es quedarà sense cobertura, i és totalment indispensable disposar el mapa, ja que el seu cost és molt elevat (Consum de CPU major que el 25%, consum d'ample de banda elevat) i els seus beneficis molt limitats.

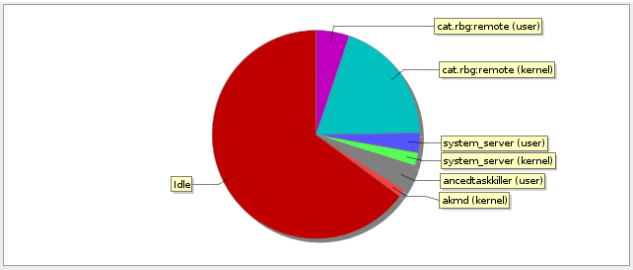

Figura 20. Estat de la CPU quan s'està descarregant les tessel·les, cat.rbg és l'aplicació OSMasygo (font:pròpia)

Des d'un punt de vista infraestructural, la possibilitat que altres dispositius continguin les tessel·les a més del servidor central, i les puguin compartir amb altres usuaris fa que sigui molt més robust i escalable.

XML, JSON o Base de dades

A continuació s'analitza les diferencies entre carregar les dades des de XML, JSON o base de dades.

Per fer aquest test s'ha fet la mitjana de pes de les tessel·les de dades vectorials que formen la ciutat de cornellà de Llobregat i s'ha triat el fitxer amb pes real més proper a la mitjana, el resultat és que el fitxer es http://api.openstreetmap.org/api/0.6/map?bbox=2.076416015625,41.368564077449825,2.08740234375,41.376808506564686, corresponent a la tessel·la 16573,12240. La qual conté 689 elements del tipus way, 591 del tipus node.

Primer de tot mirem el pes dels fitxers resultants:

XML	JSON	BBDD
164.1K	167K	80K

Figura 21.Comparativa de pes d'una tessel·la de dades vectorials

Pot cridar l'atenció que el pes de JSON sigui major que XML, ja que el més comú és el cas contrari. Les claus per a que els XML d'OSM siguin menys pesats que la seva transformació a JSON són:

- Usen atributs dels nodes del document, evitant l'abús d'elements fills.
- Tancament de nodes mitjançant l'ús de "/>" en comptes de "</*nom_del_node*>", en cas que el node no contingui fills.
- L'ús de cometes en JSON per contenir la clau (no només el valor) suma 2 caràcters per atribut.

A continuació comparem el temps que triga el dispositiu en llegir i plasmar-ho en objectes.

Dispositiu	XML	JSON	BBDD	Guardat en BBDD
HTC Hero	13278ms	3313ms	4432 ms	3639ms
Nexus One	875ms	809ms	720ms	876ms

Figura 22.Comparativa de temps de càrrega de les dades.

Podem veure que dels dos formats d'intercanvi de dades, JSON és més ràpid en els dos dispositius, on hi ha una diferencia major que el 75%.
També, s'observa que l'emmagatzematge de les dades en base de dades té un cost temporal i computacional extra. Però aquest cost es compensat amb la reducció de l'espai que ocupa i la millora de l'organització de les dades, que facilita a l'aplicació la gestió i la consulta d'aquestes.

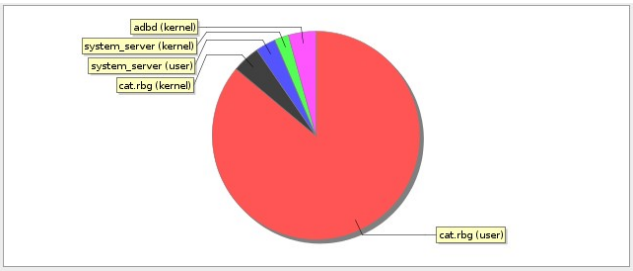

Figura 23. Estat de la CPU quan s'està carregant en memòria una tessel·la de dades vectorials des de base de dades, cat.rbg és l'aplicació OSMasygo

Per últim, podem observar la gran diferencia que existeix entre aquest dos dispositius en quant a velocitat, aquest resultat serveix per fer-se una idea dels problemes que comporta desenvolupar una aplicació que sigui optima per tots els dispositius.

Com podem veure el format més propici per intercanviar les dades amb un dispositiu Android és JSON, però en aquest cas suposa un problema afegit, ja que el servidor d'OSM únicament és capaç de retornar dades en XML. Per tant, és obligatori la introducció d'un nou node dins de la infraestructura per traduir els missatges XML a JSON.

En el cas de la prova pilot s'ha utilitzat un petit programa en llenguatge PHP que mitjançant XSL (http://xml2json.duttke.de) transforma les respostes provinents del servidor OSM a JSON.

El temps de transformació de XML a JSON és de 4 segons corrent en un PC portàtil amb sistema

operatiu Ubuntu, servidor Apache i CPU Core2Duo a 2.2Ghz, però gràcies a tècniques de catching aquest temps només existiria en la primera crida.

Descàrrega de Tessel·les de dades vectorials

Una zona d'interès de radi de 2km, equival a 9 tessel·les, el pes mitjà d'una tessel·la en aquest nivell és de 160 Kb, el pes mitjà total d'una zona d'interès de 2km és de 1440Kb.

El temps de càrrega d'aquestes dades, en cas de ser des de base de dades serà de quasi 40 segons en el cas de HTC Hero i inferior a 8 en Nexus One. Aquest temps de càrrega és fa de cara a l'usuari, ja que corre en una activitat, no en el servei. Tenint en compte que són dispositius d'ús puntual, els 40 segons, en el cas de HTC Hero, de càrrega han d'estar molt ben justificats, o sigui, que el servei que s'ofereix ha de valdre la pena aquesta espera.

Cerca de dades (POI)

A continuació es farà una comparativa entre els temps que triga el dispositiu en fer una cerca de punts d'interès (POI) en Google Maps, Nominatim, i OSMasygo (aplicació pilot).

GoogleMaps

Nom buscat	Tenps de resposta	Resultat satisfactori
Mercadona	3 seg.	SI
Carretera d'Esplugues	8 seg.	SI
Eroski	2 seg.	SI
Mercat Marsans	3 seg.	NO (Carrega la Caixa que està a prop)
Castell de Cornellà	2 seg.	NO (Resultats erronis)
Mitjana	3,6 seg.	

Figura 24. Resposta de Google Maps a la cerca de POI

Nominatim

Nom buscat	Tenps de resposta	Resultat satisfactori
Mercadona	4 seg.	SI, dos resultats
Carretera d'Esplugues	4 seg.	SI
Eroski	9 seg.	SI
Mercat Marsans	7 seg.	SI
Castell de Cornellà	5 seg.	SI
Mitjana	5,8 seg.	

Figura 25. Resposta de Nominatim a la cerca de POI

OSMasygo

Nom buscat	Tenps de resposta	Resultat satisfactori
Mercadona	0 seg.	SI, dos resultats
Carretera d'Esplugues	0 seg.	SI, es mostra la carretera no únicament un punt.
Eroski	0 seg.	SI
Mercat Marsans	0 seg.	SI
Castell de Cornellà	0 seg.	SI
Mitjana	0 seg.	

Figura 26. Resposta de l'aplicació desenvolupada a la cerca de POI

Com podem comprovar a les taules anteriors, l'aplicació desenvolupada té un comportament excepcional en les cerques, això és degut a que les dades ja estan carregades i únicament canvia la seva visibilitat.

Podem concloure que l'aplicació serà molt efectiva si es fan varies cerques, ja que el cost temporal de la primera càrrega es veurà compensat amb la velocitat de les cerques (figura 27).

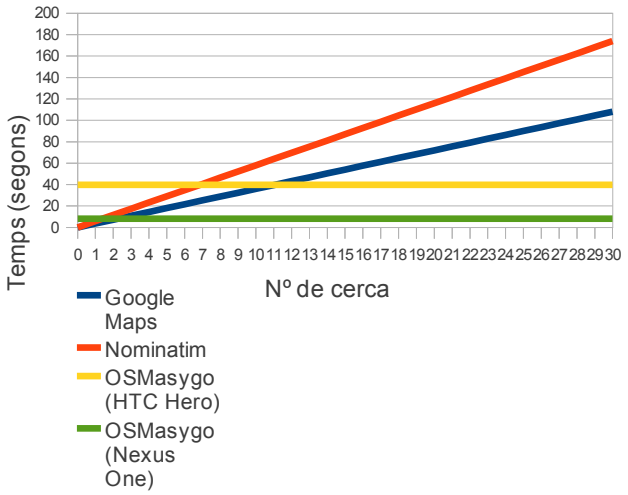

Figura 27. Comparació dels sistemes de cerca de POI de Google Maps, Nominatim i OSMasygo (aplicació pilot)

Pintat en Android

Com ja s'ha posat de relleu, Android no disposa de cap llibreria de rasteritzat de SVG nativa, i això obliga a l'ús de primitives per crear el mapa vectorial. Aquest fet fa que a major número de primitives menor sigui el framerate, fins que arriba un punt inacceptable des del punt de vista de l'usuari (Figura 28).

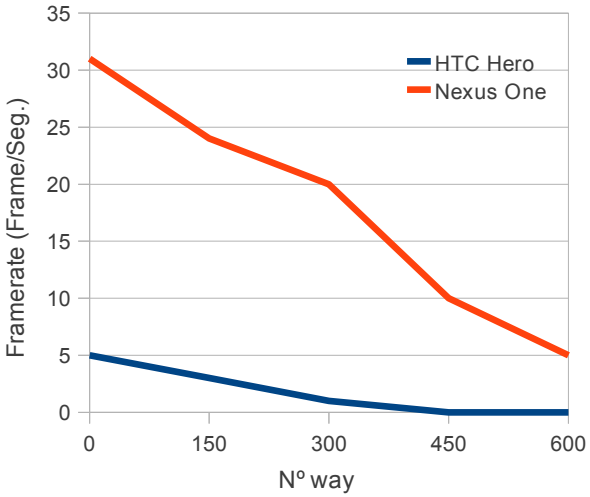

Figura 28.Número de frames/segons segons el número de primitives del tipus way (android.graphics.Path)

A continuació veiem una taula on es mostra el comportament del framerate (frames/seg.) en un exemple d'àrea d'interès.

Dades que es mostren	Nº Node	Nº Way	HTC Hero	Nexus One
MapView	-	-	2-5	27-31
Tessel·les	-	-	1-3	20-25
Amenities	45	7	1-3	21-27
Places	8	0	1-4	25-30
Leisure	0	13	1-3	25-29
Historic	0	1	2-4	27-31
Shops	9	0	1-3	25-29
Roads	19	630	0-1	5-7
Railway	118	354	0-1	15-20

Figura 29.Framerate en diferents estats de l'aplicació en HTC Hero i Nexus One

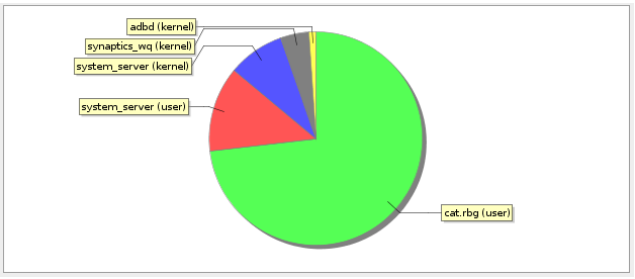

Figura 30.Estat CPU HTC Hero amb càrrega de totes les carreters, cat.rbg és l'aplicació OSMasygo

La rasterització és la tècnica de transformació de dades vectorials en dades tipus raster (matriu de pixels), aquesta tècnica permet reduir d'una suma N de dades vectorials a una única imatge.

La rasterització de les primitives vectorials ens permetria que el comportament sempre sigués el mateix (estat tessel·les), ja que totes les primitives que es pintessin serien carregades com una única imatge.

HTTPServer en Android

La senzillesa del servidor HTTP permet que pugui rebre fins a una gran quantitat de peticions, sense que la CPU pateixi en excés.

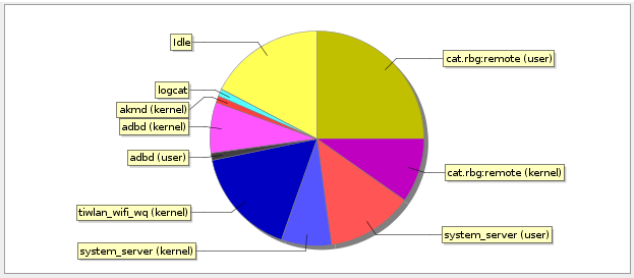

Figura 31.Estat CPU HTC Hero amb càrrega de peticions HTTP, cat.rbg és l'aplicació OSMasygo

El servidor HTTP canvia la velocitat de processament segons la resta de serveis que hi hagin actius al dispositiu mòbil, a més depenent a la xarxa que es trobi tindrà un ample de banda més o menys gran. A continuació podem veure alguns test sobre el servidor fent una petició de una tessel·la de 412Kb.

Nº d'usuaris en paral·lel	Nº de peticions	Temps de procés de totes les peticions	Temps de procés per petició
1	1	2.1 seg.	2.1 seg.
2	2	2.7 seg,	1.3 seg.
3	3	2.9 seg.	0.9 seg.
4	4	4.1 seg.	1.0 seg.
5	5	6.0 seg.	1.2 seg.
6	6	7.6 seg.	1.2 seg.
7	7	7.9 seg.	1.1 seg.
8	8	9.9 seg.	1.2 seg.

Figura 32.Temps de resposta del servidor web del dispositiu mòbil amb diferents número de peticions

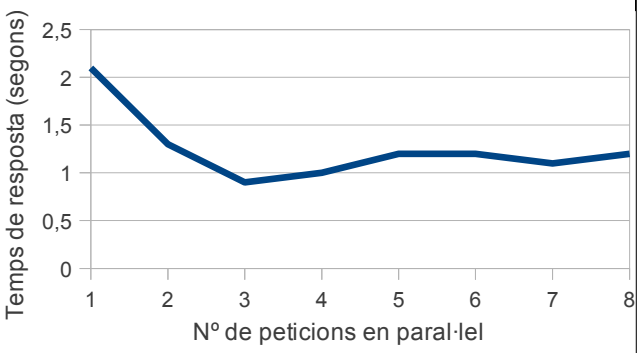

Figura 33.Tendència temps unitari de resposta per número de peticions en paral·lel

A la gràfica podem detectar que el punt més eficient és de tres peticions en paral·lel, però el comportament del servidor del dispositiu mòbil segueix sent molt estable amb un número de peticions alt.

S'ha de tenir en compte que l'ample de banda de les xarxes mòbils (sobretot la velocitat de pujada) és limitat i molt variant. Però per evitar que aquest problema afecti en excés a l'aplicació s'han implementat varies mesures:

- La divisió en tessel·les (tant d'imatges com de dades vectorials) permet descarregar les dades de diferents servidors o peers de manera paral·lela (cadascuna corre en un thread), d'aquesta forma el temps de descàrrega de totes les dades serà igual al temps que triga la tessel·la més lenta en descarregar-se.
- La descàrrega de dades la realitza un component del tipus servei (service), el qual contínuament comprova i descarrega les dades que formen part de la zona d'interès de l'usuari, encara que l'aplicació no estigui visible i l'usuari estigui realitzant altres operacions amb el seu dispositiu mòbil. Amb aquesta mesura, l'impacte del temps de descàrrega no el pateix l'usuari.
- A l'aplicació es deixa en mans de l'usuari la possibilitat de compartir les seves dades.

SEGÜENTS PASOS

En aquesta infraestructura només tenim un node Rendez-Vous, el qual envia un llistat dels usuaris connectats, és el propi client qui verifica que aquests usuaris tenen o no tenen les dades que demana. Si hi ha una quantitat petita d'usuaris aquesta infraestructura és viable, però amb un número d'usuaris elevat els dispositius client no sabrien quins dispositius, dels que estan connectats, hi ha més probabilitat que tingui les dades que ell demana.

Per reduir aquest comportament es proposa crear nodes Rendez-Vous dividits per parcel·les d'espai, d'aquesta manera els usuaris només veuran a altres usuaris connectats a prop seu.

Per una altra banda, si volem donar més intel·ligència al node client per a càlculs geogràfics es preveu imprescindible tenir una base de dades amb aquestes capacitats, com podria ser Spatial Lite.

En Android, avui dia, existeixen quantitats immenses d'aplicacions que consumeixen i realitzen càlculs sobre dades geogràfiques del nostre voltant (Location Based Services). En cadascuna d'aquestes aplicacions s'està replicant bona part del codi, fent que les aplicacions siguin molt pesades però aportin poc nou. Buscant la reducció d'aquest fenomen, Android posa a disposició dels desenvolupadors un component, anomenat Content Provider, que dona la capacitat de compartir contingut entre diferents aplicacions, amb aquesta tècnica es podria crear un únic Content Provider, que adquirís tota la informació del voltant de l'usuari (es podria basar en el desenvolupament fet en aquest projecte) i que donés servei a les aplicacions que vulguin aquestes dades, sense que aquestes ho hagin d'implementar de 0.

5) Conclusions

En el projecte hem vist com Android ens dona la capacitat de millorar l'eficiència en el client i en la infraestructura, deixant de ser un simple consumidor i transformant-se en servidor, tant de mapes com de dades.

Les tècniques de tessel·lat del món, tant d'imatges com de dades, permeten que l'administració de les dades i la compartició sigui molt més fàcil i redueix la intel·ligència del client a l'hora de fer de servidor.

A més, s'ha demostrat que amb una poc costosa transformació de dades de XML a JSON és millora el temps de anàlisis fins a un 75%.

La descàrrega de contingut segons l'usuari camina permet tenir actualitzades les dades geogràfiques que envolten a l'usuari, per quan l'usuari decideixi consumir-les. L'aplicació ha demostrat que mitjançant la càrrega en memòria de totes les dades geogràfiques que envolten a l'usuari, la cerca de POI es pot reduir a un temps mínim.

Per altra banda, al projecte hem vist com avui dia el pintat de dades vectorials en Android és un problema, ja que els dispositius, encara que cada dia són més potents, no són capaços de moure gran quantitat de primitives amb un framerate constant. Aquest problema es solucionaria amb la implementació del rasteritzat de primitives vectorials, transformant-les en una única imatge, o bé el pintat de SVG amb rasteritzat natiu.

Referències

[1] Aijaz,F., Aziz M., Walke, B. , Performance Comparison of a SOAP and REST Mobile Web Server, Third International Conference on Open-Source Systems and Technologies 19-22 December 2009, Lahore, Pakistan, 2009

[2] Android developers http://developer.android.com/index.html

[3] Brinkhoff T. , supporting mobile GIS applications by geospatial web services, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.158.6959&rep=rep1&type=pdf , 2008

[4] Cao, Y., Klamma, R., mrama, S., Wang, W. ,The mobile interfaces for geo-hypermedia databases, Lecture Notes in Computer Science, 2007

[5] Kikkert, S. ,Performance of Web services on Mobile Phones , http://scripties.fwn.eldoc.ub.rug.nl/FILES/scripties/Informatica/Bachelor/2010/Kikke rt.S.C./INF-BA-2010-S.C.Kikkert.PDF , 2010

[6] Li, B., Wilkinson G. ,Research issues in the development of advanced mobile web-based geospatial information services, Third International Conference on Internet Technologies and Applications (ITA 09) , 2009

[7] Montesinos, M., Carrasco, J. , GIS Mobile Comparison, Foss4g conference, 2010

[8] M. Montesinos Lajara(1), J. Carrasco Marimón (2) y A. del Rey Pérez(2) , GvSIG Mini y Phone cache, http://www.sigte.udg.edu/jornadassiglibre2010/uploads/Articles/a41.pdf , 2010

[9]OSGEO, http://wiki.osgeo.org/wiki/GIS_Mobile_Comparison#Feature_Comparison , WMS Tile Caching, 2010

[10] Poorazizi,E., Alesheikh, AA., Behzadi, S. , Developing a Mobile GIS for Field Geospatial Data Acquisition, Journal of Applied Sciences 8 (18), 2008, 3279-3283 p

[11] Shi, W., Kwan, K., Shea, G., Cao, J. , A dynamic data model for mobile GIS, Computers & Geosciences; Nov2009, Vol. 35 Issue 11, p2210-2221, 12p

[12] Srirama, S., Jarke, M., Prinz, W. Mobile Web Service Discovery in Peer to Peer Networks, https://arxiv.org/pdf/1007.3631 , 2007

[13] Srirama, S. Mobile host in enterprise service integration, PhD, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.7756&rep=rep1&type=pdf ,2008