Building a fast tile server

Presented by :

Guillaume SUEUR

GIS Expert and OpenSource Consultant

NEOGEO – Toulouse – FRANCE

http://www.neogeo-online.net/ (blog in french)

CAPE TOWN SOUTH AFRICA

FOSS4G 2008

2008 FREE AND OPEN SOURCE
SOFTWARE FOR GEOSPATIAL CONFERENCE

OSGeo

GISSA

Geo - Information Society of South Africa

Building a fast tile server

- Tiling geographic data is not only a technic, it is also a strategy
- Software strategy
- Architecture strategy
- Tileset construction strategy
- As we don't have Google's powerful infrastructure, we need to be smart !

Software strategy

- Rather straightforward for WMS server : MapServer ou GeoServer will do the job
- Tiling software : TileCache (GDAL2Tiles was not ready yet, and can only build full tilesets)
- Web server : Apache vs Lighttpd benchmark

Tilecache

- See http://tilecache.org/
- Python program for generating tiles from WMS server
- Able to render them back via WMS, TMS...
- Creates a tile tree for each cached layer
- Plugged into a web server able to handle python scripts.

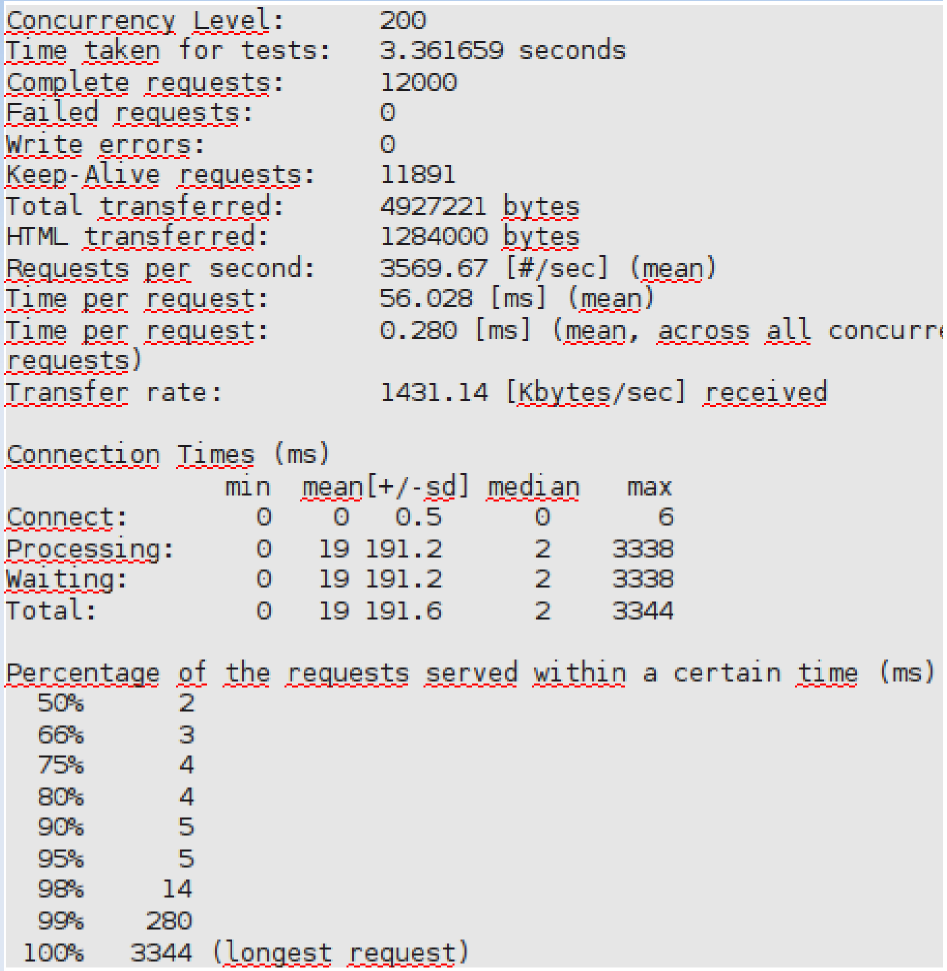
Apache

- Apache is an excellent webserver, able to handle very fine tuning
- It can execute python scripts via fastCGI or a dedicated module : mod_python.
- fastcgi python is rather slow, while mod_python performances are excellent in terms of speed and ressources usage.

Lighttpd (say Lighty)

- Recent, small and fast webserver
- primaly designed to deliver static pages at very high speed
- handles python scripts via fastCGI

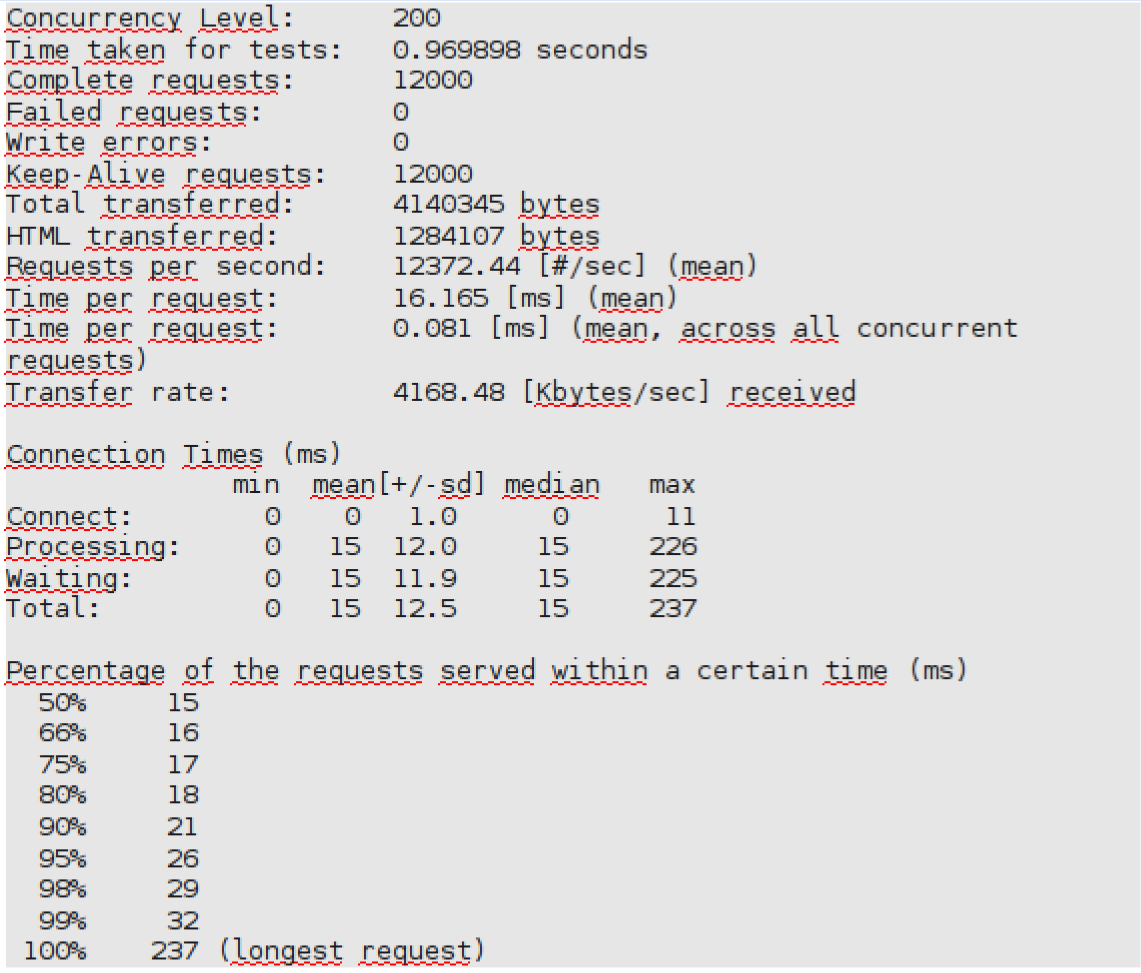
Apache vs Lighttpd on static files



```
Concurrency Level:      200
Time taken for tests:   3.361659 seconds
Complete requests:      12000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    11891
Total transferred:      4927221 bytes
HTML transferred:       1284000 bytes
Requests per second:    3569.67 [#/sec] (mean)
Time per request:       56.028 [ms] (mean)
Time per request:       0.280 [ms] (mean, across all concurre
requests)
Transfer rate:          1431.14 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       6
Processing:     0   19 191.2      2    3338
Waiting:        0   19 191.2      2    3338
Total:          0   19 191.6      2    3344

Percentage of the requests served within a certain time (ms)
  50%      2
  66%      3
  75%      4
  80%      4
  90%      5
  95%      5
  98%     14
  99%    280
 100%   3344 (longest request)
```



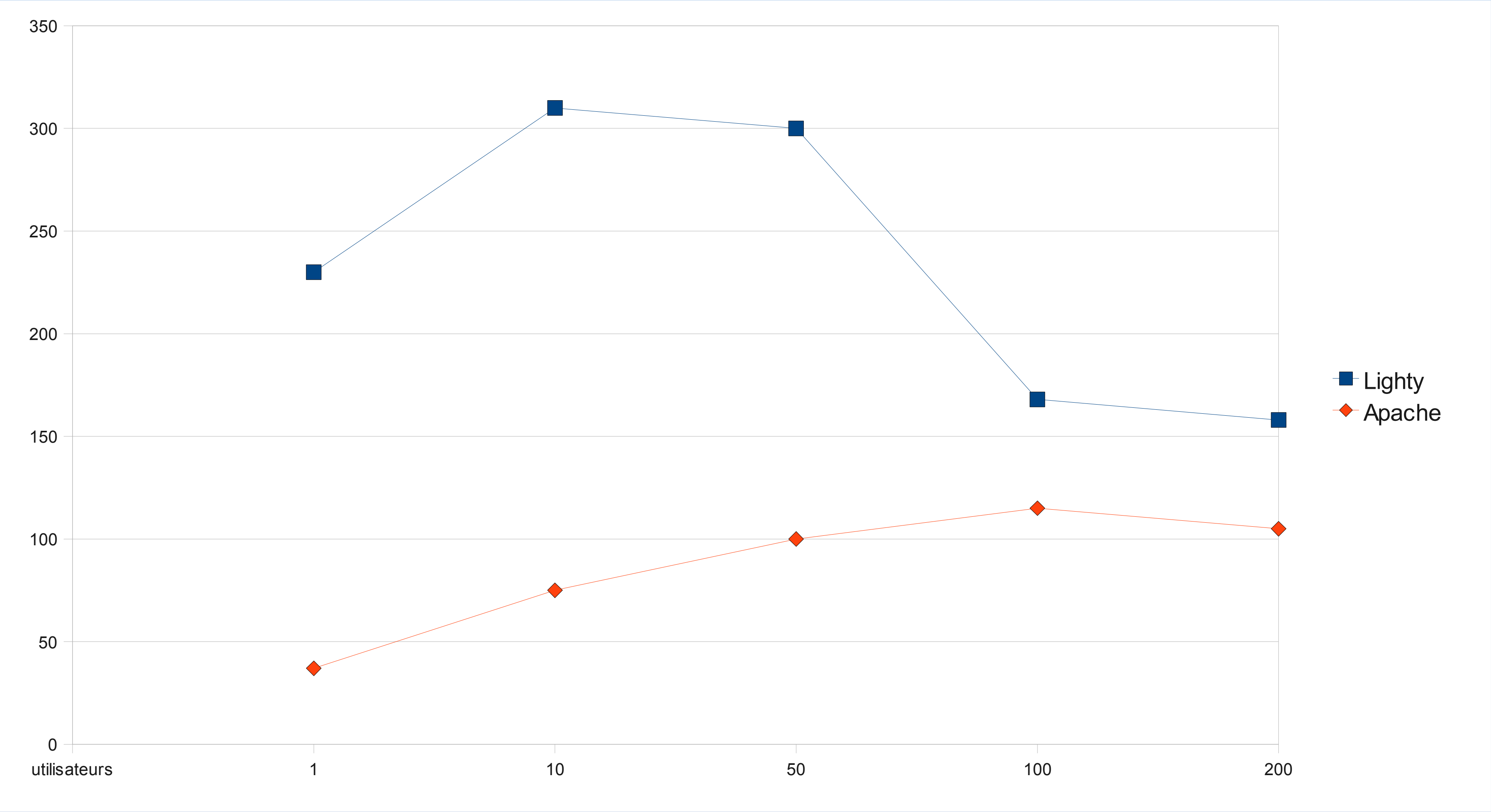
```
Concurrency Level:      200
Time taken for tests:   0.969898 seconds
Complete requests:      12000
Failed requests:        0
Write errors:           0
Keep-Alive requests:    12000
Total transferred:      4140345 bytes
HTML transferred:       1284107 bytes
Requests per second:    12372.44 [#/sec] (mean)
Time per request:       16.165 [ms] (mean)
Time per request:       0.081 [ms] (mean, across all concurrent
requests)
Transfer rate:          4168.48 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   1.0      0      11
Processing:     0   15  12.0     15     226
Waiting:        0   15  11.9     15     225
Total:          0   15  12.5     15     237

Percentage of the requests served within a certain time (ms)
  50%     15
  66%     16
  75%     17
  80%     18
  90%     21
  95%     26
  98%     29
  99%     32
 100%    237 (longest request)
```

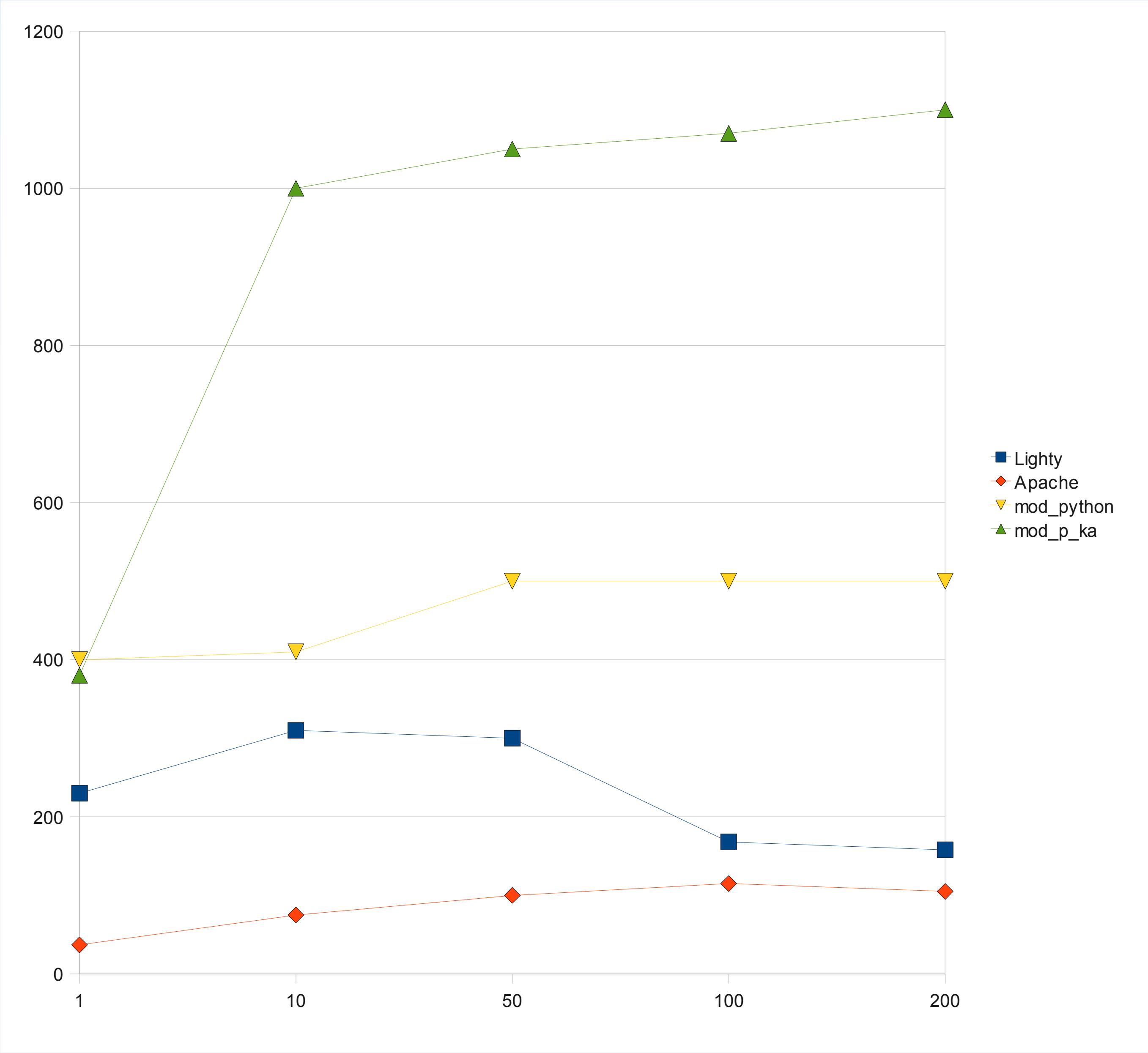
ab results for 60 requests sent by 200 clients at once. It simulates a typical tiled layer display in OpenLayers.

Apache vs Lighttpd in fCGI



(c) Guillaume Sueur - Neogeo 2008

Apache vs Lighttpd



Compromise ?

- The best performances on generating tiles is obtained by apache because mod_python handles tilecache better
- But lighttpd is faster to deliver already generated tiles (static files)
- The ideal could be to use apache to generate the cache, and then lighttpd to deliver the tiles

BUT

- Accessing the tiles throught tilecache is not accessing static files, and is still fastCGI bad performances, even if it doesn't create the tiles.
- A complete cache can be a hassle to generate, especially on high-resolution data (why should I create Gigs of data if no one ever see it !)
- If lighttpd is dedicated to delivering tiles, does the cache need to be complete ?

WORKAROUND

- We need to access to the tiles directly by their own url. The cache must be web published for that.
- The client must be able to compute the whole image url by knowing x,y,z of the tile.
- WMSClients are unable to do that, but TMS clients are.

OpenLayers TC layers

- Direcly computes the url from zoom and x,y position of tile
  - example...
- Gets high performances on hitting directly the cache published by lighttpd
- Get pink tiles when calling a non-existent tile...

Handling 404

- When the tiles does not exist, lighttpd fires 404 error.
- The error can be handled by a specific page, which can be a script...
  - server.error-handler-404 = '/error.py'
- The 404 custom page could redirect the failed request to an apache instance running mod_python

404 script

```
#!/usr/bin/python

from flup.server.fcgi_fork import WSGIServer

import os,sys

def getTile(environ,start_response):

    import urllib2

    start_response('200 OK', [(Content-Type','image/png')])

    req = environ['REQUEST_URI'].split('/')

    layer = req[2]

    z = str(int(req[3]))

    x = str(int(req[6]))

    y = str(int(req[9].split('.')[0]))

    TileUrl = 'http://127.0.0.1:8080/tilecache/tilecache.py/1.0.0/ '+layer+'/'+z+'/'+x+'/'+y+'.png'

    try:

        httpReq = urllib2.Request(TileUrl)

        handle=urllib2.urlopen(httpReq)

        yield handle.read()

    except:

        print TileUrl

WSGIServer(getTile).run()
```

Benefits

- In that configuration, we are now able to deliver a huge quantity of tiles very quickly
- Missing tiles are handled by another server, generating tiles is so handled by another server and does not use our main server performances.

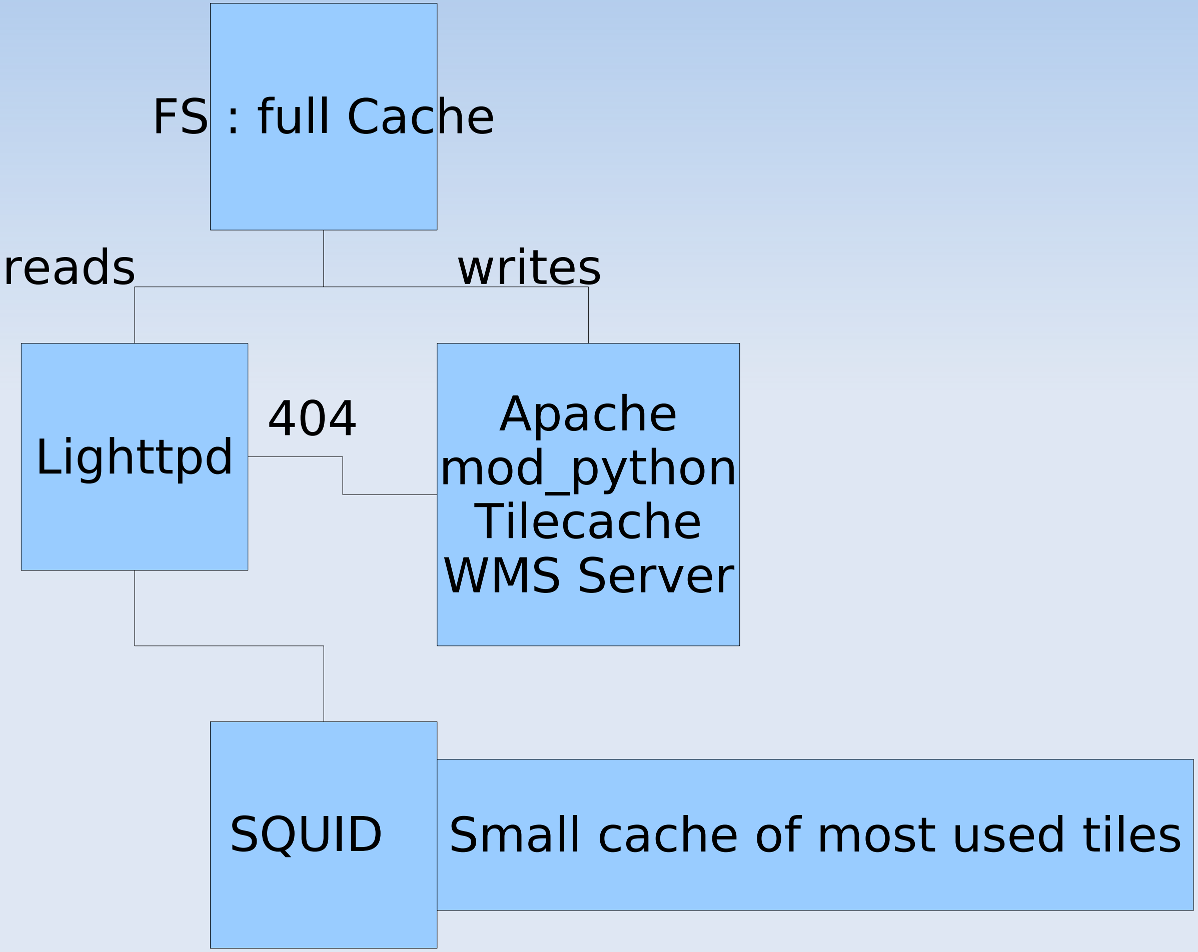
Architecture strategy

- in the real life, only few tiles are used by people, mainly for the higher scales.
- handling a huge tileset on a filesystem cost a lot in terms of IO
- it can be interesting to proxify the most used tiles, then stored in another FS

SQUID

- Configured as a reverse proxy, squid can store the most used tiles
- Reduced to few Gigabytes, this FS is light enough to handle a huge amount os simultaneous connections.

FINAL LAP


FS : full Cache

reads

writes

Lighttpd

404

Apache
mod_python
Tilecache
WMS Server

SQUID

Small cache of most used tiles

(c) Guillaume Sueur - Neogeo 2008