Introduction

An important part of the XSCE content collection is the easy availability of high resolution offline maps. Currently in mid-2015, we include Braddock Gaskill's amazing Internet in a Box's (IIAB) OSM metatile dump as part of the XSCE project, but those tiles are now about 3 years old, and the OSM project has moved forward. Also, this is a good opportunity to have a crystallization of design requirements and a roadmap for the same. What follows is a summary of things discussed in the weekly XSCE call on June 11, 2015.

Team

Feel like you could contribute? - We meet every Thursday 10 AM EDT on Skype. Drop anyone below an email, or to the server-devel mailing list. Also, feel free to add your name to the list below :-)

- Nick ndoiron@mapmeld.com
- Tim tim@timmoody.com
- Jerome gagnonje@gmail.com
- Adam holt@laptop.org
- Anish anishmg@umich.edu

Design principles:

- **No regressions** - All currently supported features must have an equivalent functionality. Regressions if any will be highlighted red.
- **Offline > Semi-offline >> Online** - Our primary use case is to serve offline environments. Then comes a group of schools/regions which have limited connectivity some time of the day. Finally, a very small subset has reliable and constant connectivity. Even in the last case, bandwidth costs are often high.
- **Distribution** - Ideally, distribution should be as simple as copying ONE file onto an XSCE server. There must be a path (whatever the approach), that allows this to be integrated into the admin web-UI. The Kiwix project is one good example to draw inspiration from.
- **Upgradability** - Updating/Refreshing the maps data or tiles ideally should not involve doing everything from scratch. The process should be capable of handling incremental updates. This is desirable, BOTH on the generation side, as well as updating remote images on XSCE installations
- **Customizability** - Based on constraints of storage space, and regional preferences, there could be some level of customizability in making an offline maps package for an XSCE distribution.

- **Multilingual** - Would be great to have the tilesets available in various different languages.

Constraints

- XSCE is currently a Fedora or CentOS based distribution. Whatever solution is chosen must run well and be actively maintained and supported on these platforms.
- The proposed solution must run with acceptable performance on a Raspberry Pi 2 and upwards, on better ARMed machines + x86. If some parts or features require a better performing server, they must be pluggable without crippling the system in their absence. As a point of reference, a lot of deployments are choosing to base on Intel NUC's (Intel atom, i3 & i5) so that could be a reasonable target for a fast/responsive user experience.

Current implementation

Roughly, the way things are setup right now are as following:

- The entire planet's map dataset (in pbf format - about 27GB in size) is imported to a postgres database.
- A rendering tool, renderd, is used to generate metatiles upto the required zoom level.
- Those metatiles are bulk-copied onto an XSCE server, as part of the IIAB distribution. These tiles are then served through a leaflet viewer, with mod_tile (or equivalent) rendering the png images on the server.
- Additionally, Whoosh 2.6 is used to facilitate searching for places on the maps.

Issues

- Generating metatiles requires a fast machine with lots of RAM and even then takes up a lot of time. The tool chain and process is not push-button.
    - Braddock's machine has a 500 GB SSD and even that was not enough to accommodate the postgres database AND the nodecache. The result being that it takes "forever" to generate metatiles.
- The Whoosh-based searching is quite slow to respond and does not contain a comprehensive database of places which are searchable.
- This is not a standalone service as it ideally could be. Requires iiab server software and copying of osm and geocache (name?) folders to the /knowledge folder in order for things to work. Makes customizability hard.
- The rendered tiles do not always show non-Roman character place names.
- (Perhaps this is a minor issue) The font sizes in the currently rendered tiles are too small to show up properly on XO laptops, and small in general. The style file could be tweaked to make them slightly larger or stand out better.

Looking forward…

- At a very basic level, a new design must offer the functionality described above, minus the issues.
- It must offer a path to adding more features, described later in this document.
- Goes without saying that the tool chain must be completely open/free software/content based.

Open questions

(Some of these questions may become irrelevant as we dig deeper into possible solutions)

- **metatiles v/s png**

metatiles (or mbtiles)	**png**
Requires less space	Requires more space *(by a factor of how much?)*
Requires lots of resources and time to be generated	Can just be bulk downloaded from the internet
Server side cpu usage while serving metatiles to render png	Should theoretically (atleast) be faster
• Are there better solutions? • Are vector tiles (what are they?) worth looking into? - *Following from some discussion, probably not worth our time right now. This is for more shiny stuff which requires decent client side performance, an assumption not always true for our case.*	

- **Architecture: backend python vs front end javascript - perhaps Tim Moody could elaborate?**
- **Question - Would it make sense to have a base tile set without labels, and have multiple transparent layers containing labels in different languages? Is there a use case where a particular deployment would want tilesets in different languages?**

Desireable features

- It is highly desirable to have ‘map-editing’ features. While OSM-style editing would be too resource consuming, basic editing like placing markers or other features on a layer and those being searchable would be very useful.
    - Possible platforms that may offer this are - umap, akshara
- Sync with online servers. Again, would be very useful to integrate changes back to OSM, but even if that’s not possible, some kind of syncing between a web of XSCE servers and an online database would be desireable. *Properly identify privacy implications and be aware of them before implementing this.*

- Multilingual tiles
- <add more here, also we should start prioritizing>

Roadmap & proposed solutions

(Needs organization, just copying from the minutes as is...)

- http://www.openstreetmap.org/user/SimonPoole/diary/34857 (as put up by an OSM community member simonpoole after a convsersation on IRC)
- Mapsforge (they seem to have files for individual continents/countries, but not the entire world - perf issues?)
- Searching -> possible solutions
    - Whoosh
    - Nominatim:
    - photon:
    - Pelias

Proposed MbTiles solution

Mbtiles definition:
The MBTiles specification is an efficient format for storing millions of tiles in a single SQLite database (single file), created by Mapbox.
MBTiles takes advantage of utilities found in SQLite for solving problems like duplicate imagery. Maps that cover large areas of solid color like ocean or empty land can contain thousands of duplicate, redundant tiles. Thousands of tile coordinates can be paired to the same raw image drastically reducing the filesize required to serve a map of multiple zoom levels.
https://www.mapbox.com/guides/an-open-platform/#mbtiles

1. We need to generate MBTiles.
    a. Hardware: Probably some amazon EC2 instance (or other cloud VM provider)
    b. Software:
        i. Download large planet OSM file or per-country OSM file on geofabrik
        ii. osm2pgsql Convert OSM to postgresql postgis database and use openstreetmap-carto.style as style
        iii. osm-bright To apply the styles to the map, which generates a tilemill project
        iv. Tilemill To export pre-rendered tiles to MBTiles
            1. 1 MBTiles file per country (high zoom)
            2. 1 Global MBTiles (not as a high of a zoom, to cover the full planet)
2. We need to host the MBtiles
    a. Hardware: Would probably be cheap enough on S3
    b. Software: Static page to link to each mbtiles previously generated
3. XSCE Server
    a. Need an interface that would allow to download MBtiles for each desired country + the global mbtiles

b. Software:
    i. Choice A: Serve the MBtiles with TileStache, which is a python tile server
        1. Cache to HD once this has been served from mbtiles once, for faster access next time
        2. Can be its own python server or integrate with apache through mod_python
        3. ***To be confirmed:*** Support composite layer , which allows to merge multiple tiles sources into one (to merge the global mbtiles, with per-country ones for example, or add custom user-generated snippets/information on top of the map using UTFGrid).

    ii. Choice B: Serve with TileStream (node.js)
        1. Does not support composite layer, but .mbtiles could be merged after they are downloaded, to create a unique merged-mbtiles file, that can easily be served. Can be done with the patch tool available here: https://github.com/mapbox/mbutil/issues/8

4. Client laptops
    a. Software: Leaflet JS client library

To-Dos

Anish

- Load a small country's pbf in postgres and generate metatiles. Analyze performance
    - As expected, for Nepal (as an example) the whole process was pretty fast. - http://lists.sugarlabs.org/archive/iaep/2015-June/017473.html
- Reach out on #osm IRC

Nick

- Explore umap as a possible part of the solution

Jerome

- Research more about search solutions

Tim, Adam, Jerome

- Meetup in Toronto please? ;-)

Research on various search solutions

Whoosh

https://pypi.python.org/pypi/Whoosh/

- Search solution well integrated with Python.
- Not built for OSM

- Used by old IIAB
- Lightweight

Comment:
This is what the old IIAB solution was using. This is not a solution specific for OSM data, from the look of it, it sounds like IIAB was the only project in the entire OSM community to use Whoosh as search engine. My gut feeling is that it was a very rudimentary solution, that was “working” but was not at the state of the art in terms of search technology and was definitely not searching into all possible OSM data. Though it has the benefit of being very light and built into Python (like the former solution)

Recommendation: **No**: suggest to use a more advanced search engine for better result

Nominatim:

http://wiki.openstreetmap.org/wiki/Nominatim

- API based search engine on top of Postgre SQL DB
- Very heavy on RAM, HD, HD speed
- No search as you type
- Takes a very long time to generate
- Decent search results (JGV: used it before)

Comment:
Nominatim is an API built on top of Postgre SQL, that uses the full planet import of OSM. On a 12-core machine with 32GB RAM and standard SATA disks, the initial import (osm2pgsql) takes around 20 hours and the indexing process another 250 hours. It requires 500Gb of hard drive.

Recommendation: **No**: while it is the standard search engine for OSM and it offers a great API, having to host that offline on a limited hardware is non-practical

photon:

https://github.com/komoot/photon

- Long indexing time
- Built for OSM search
- Built on elasticsearch
- Offers autocomplete
- API for UI integraiton
- Language support
- Supports bias toward items closer to location

Comment:
Open-source elasticsearch OSM search solution maintained by a small company. It essentially export data from nominatim and then imports it into Elasticsearch. So if we ship it with IIAB, we

essentially just ship the indexed elasticsearch + API sitting on top of it (photon). It looks like it provides good performance & good adapted search for OSM. The big downside of the solution is the time to generate all the OSM data in nominatim, but this isn't a concern for IIAB itself (we only need to ship the output of that index).
Unknown is how big would this elasticsearch solution be on the HD and how much RAM this would use.
We might need to filter out some data out of nominatim before we do this export.

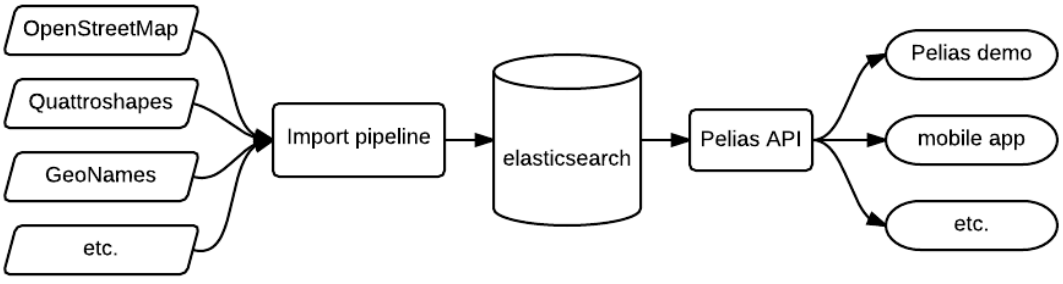
Process:
Import OSM data in Nominatim(Postgre SQL) ---> Stripe out what you don't need from the DB---> Import Nominatim data into Photon (elasticsearch index) --->Ship the elasticsearch index with the deployment (no nominatim needed)

Recommendation: Possible solution.

Pelias

https://github.com/pelias/pelias

- Based on ElasticSearch
- Built for location data (including OSM)
- Offers autocomplete
- API for UI integration
- Supports location bias



This project is actively developed by Mapzen (more updates than photon). Doesn't require nominatim install. Supports OSM and other databases for geolocations.

Similar to photon, you would do the import into elasticsearch before the deployment, and just ship the final elasticsearch index into the field.
Allows to customize the import process to stripe out things that we don't need

Unknown at this point is also RAM usage and HD space usage.
Recommendation: Possible solution.

Updates:

XSCE Map call early this SUNDAY 10 AM EDT, 11th October. Please leave your skype id here (or tinyur.com/xsceminutes) if you want to join in!

Progress

1. metatile rendering works! It is much faster now thanks to better postgresql tuning
    a. the dataset was imported in 26 hours
2. Tiles generation upto zoom level 8 took 3 hours. Upto zoom level 10, possibly 5-6 hours.
    a. Worldwide tileset from scratch in 26+6 = 32 hours
3. Experimenting with:
    a. multilingual tiles (base layer + different transparent label layers)
    b. disputed borders (may be applied as a tileset patch over a limited bbox)
    c. can leaflet conditionally zoom to a higher level at certain coordinates, or a certain bbox? somebody please experiment!
    d. topography - does somebody have access to the ASTER GDEM dataset?
4. Request for generating tilesets:
    a. For a country, the following info is needed
        i. bbox (maybe more than one to optimize the size)
        ii. preferred language