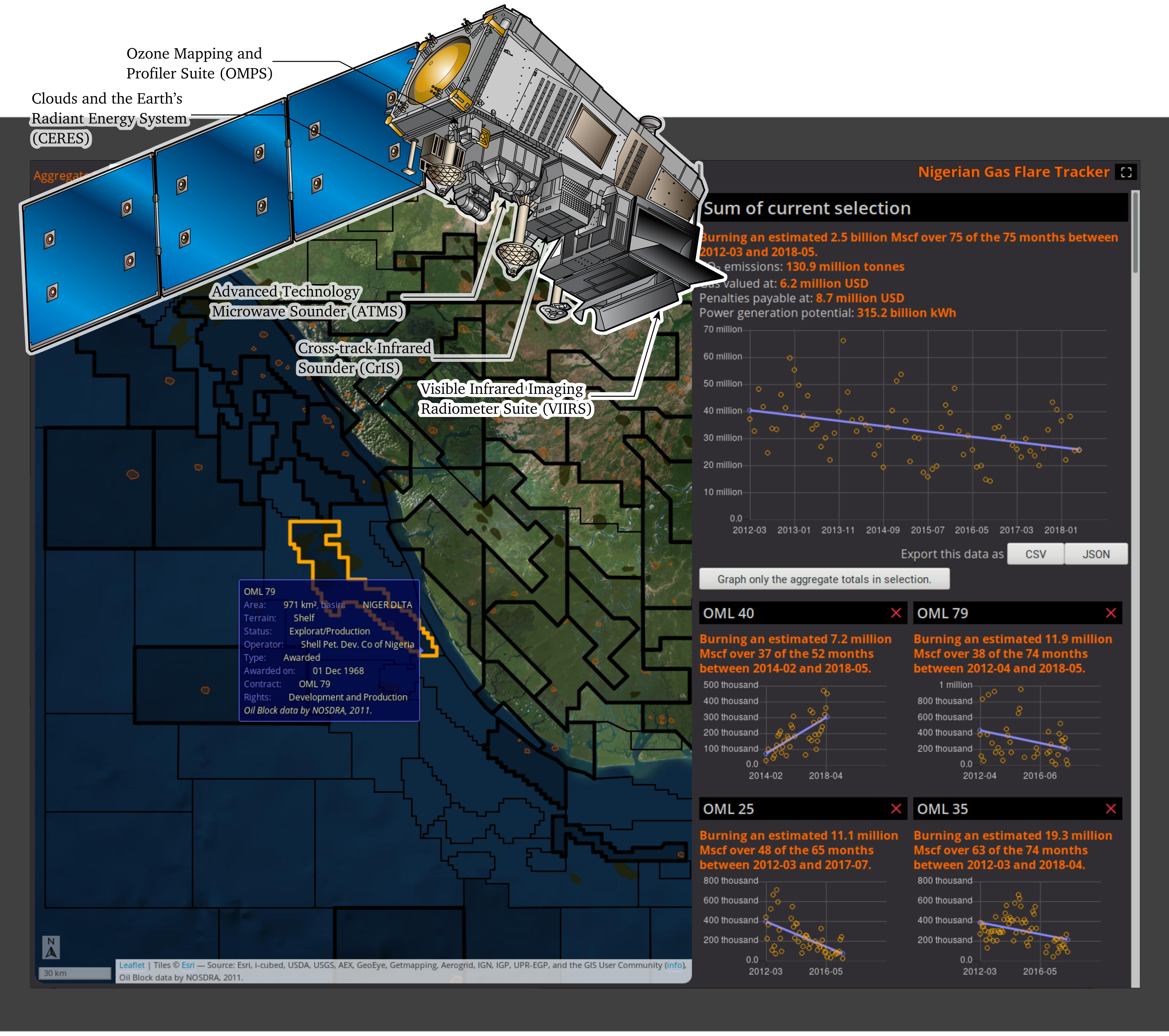
The Nigerian Gas Flare Tracker



Sum of current selection

Burning an estimated 2.5 billion Mscf over 75 of the 75 months between 2012-03 and 2018-05.

O_2 emissions: 130.9 million tonnes

as valued at: 6.2 million USD

Penalties payable at: 8.7 million USD

Power generation potential: 315.2 billion kWh

OML 40

Burning an estimated 7.2 million Mscf over 37 of the 52 months between 2014-02 and 2018-05.

OML 79

Burning an estimated 11.9 million Mscf over 38 of the 74 months between 2012-04 and 2018-05.

OML 25

Burning an estimated 11.1 million Mscf over 48 of the 65 months between 2012-03 and 2017-07.

OML 35

Burning an estimated 19.3 million Mscf over 63 of the 74 months between 2012-03 and 2018-04.

OML 79	
Area:	971 km², basin: NIGER DLTA
Terrain:	Shelf
Status:	Explorat/Production
Operator:	Shell Pet. Dev. Co of Nigeria
Type:	Awarded
Awarded on:	01 Dec 1968
Contract:	OML 79
Rights:	Development and Production

Oil Block data by NOSDRA, 2011.

Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community (info), Oil Block data by NOSDRA, 2011.

Alberto González Palomo

alberto@sentido-labs.com

Sentido Labs.com

Revision: 1.1

Date: 2019-07-03

Contents

Introduction

The Nigerian Gas Flare Tracker is a web application that identifies gas flares in Nigeria, and estimates the energy wasted, value of gas burnt, missing fees (penalties for gas flaring are seldom enforced), potential for electricity generation, and CO_2 emissions.

This 2018 edition is the successor of the first Nigerian Gas Flare Tracker (GFT) I made in 2014, which was an adaptation of the Nigerian Oil Spill Monitor (OSM). The following was part of my e-mail to Stakeholder Democracy Network (SDN) and the Nigerian National Oil Spill Detection and Response Agency (NOSDRA) on 2014-01-13:

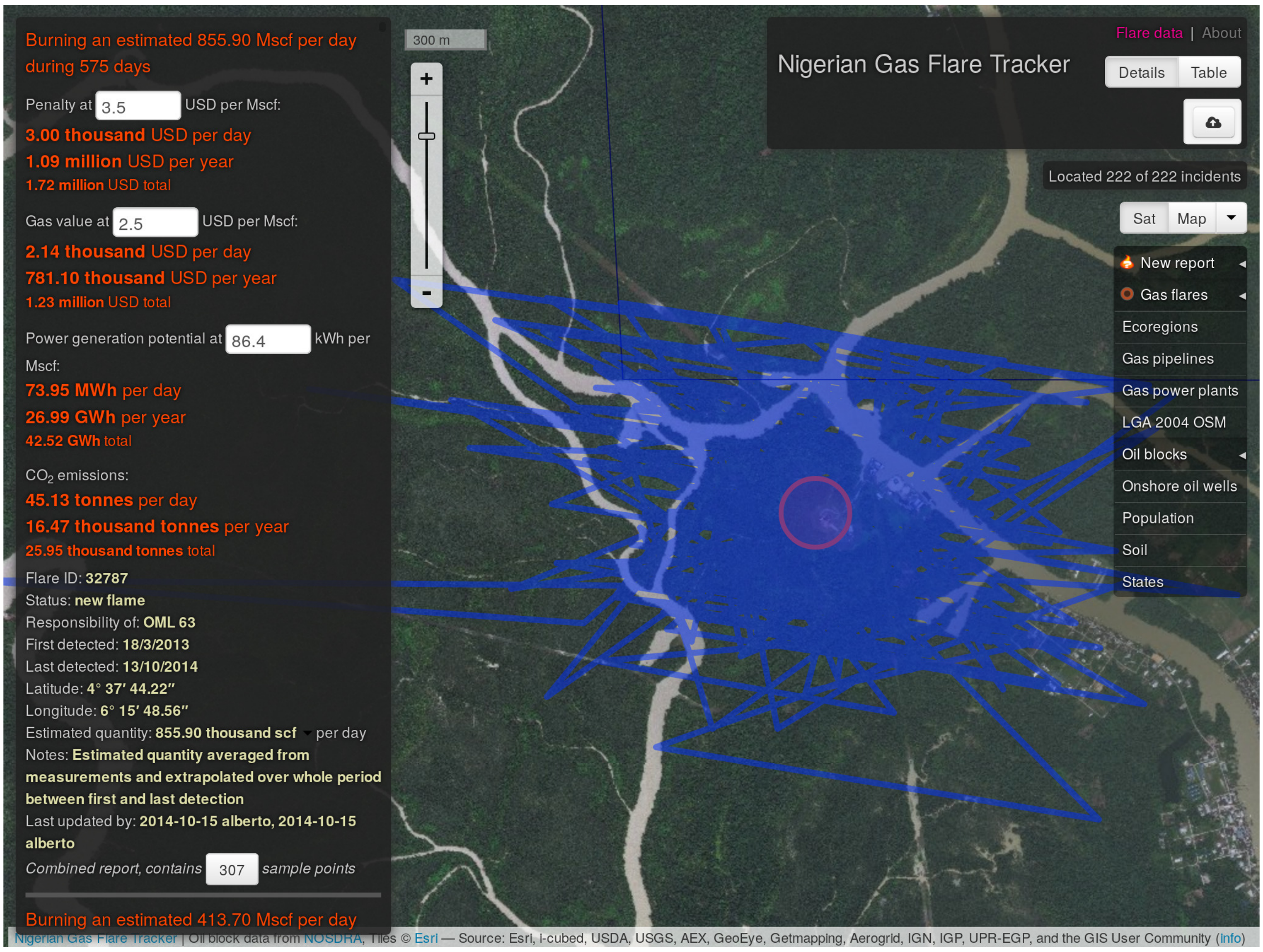
> I have ideas for improvements in the application that I want for myself, and I'll be glad to add them to SDN's version if they consider them useful. One that I just did and is relevant for this project is a new Flares data layer. There was a previous Gas Flares layer that only had locations of supposed gas flares but no additional details: `http://a.tiles.mapbox.com/v3/nigeriaoil.gas-flare-locations/page.html`
>
> The new one is built from the satellite measurements conducted by the NOAA/NASA mission VIIRS Nightfire, and includes a link to an academic paper where they detail the exact analysis process used to produce that data. You can see that layer in the application:
>
> `https://oilspillmonitor.ng/`
>
> On the right side, in the list of overlay layers, find the one called "Flares" and click on it. After a few seconds you'll see a collection of circles of different colours and sizes added to the map. (It gets quite crowded but you can click on "Oil company/Third party/ Pending visit" to hide those other layers) Clicking on any of those new circles will bring up an "information balloon" like that in the Oil blocks layer, with the flare temperature and other data on it, including a CO_2 emissions estimation if available. The colours correspond to temperature ranges: red and yellow are very hot flames, typical of gas flares. Blue and purple are colder, which suggests that they are wood (or other biomass) burning, with green somewhere in between. The size indicates the total heat produced. If you zoom out a bit, you'll see that the flares in the delta are mostly high-temperature flames, while those further north are lower temperature. Keep in mind that these are measurements from a satellite so there is some margin for error, but the general picture is quite clear. That layer includes only flares detected during the night of the 6th of January of 2014. The satellite makes one pass over the whole world each night and the results are published the next day.

In June 2014 SDN went ahead with a project to locate flare sites and estimate how much it was costing the Nigerian people in wasted energy, missing income and fees, and how much CO_2 was being dumped into the atmosphere.

Their initial plan was to enter flare sites manually, starting from the satellite measurements as a guide on where to look for flares in the Google Maps base layer ("satellite view") which at the time was the most detailed freely-available aerial photography. This turned out to require much more manual work than they expected.

What I did then was to process the data off-line to locate likely flare sites. The satellite measurements do not track individual flares over time: due to small inaccuracies in geolocation (see "VIIRS Geolocation Accuracy Monitoring") the coordinates of the measurements for a same flare varied some times more than one Km between different nights.

The result totals were below the official figures, which is probably because the usual cloud cover on the Niger Delta obscures the flares for most of the year. To obtain yearly totals it might be better to use the approach from "Remote Mapping of Gas Flares in the Niger Delta with MODIS imagery" [ABW]: using the measurements during the Harmattan weather period when that dry wind blows from the Sahara (roughly December to February), and extrapolating to the whole year.


Burning an estimated 855.90 Mscf per day during 575 days

Penalty at 3.5 USD per Mscf:
3.00 thousand USD per day
1.09 million USD per year
1.72 million USD total

Gas value at 2.5 USD per Mscf:
2.14 thousand USD per day
781.10 thousand USD per year
1.23 million USD total

Power generation potential at 86.4 kWh per Mscf:
73.95 MWh per day
26.99 GWh per year
42.52 GWh total

CO_2 emissions:
45.13 tonnes per day
16.47 thousand tonnes per year
25.95 thousand tonnes total

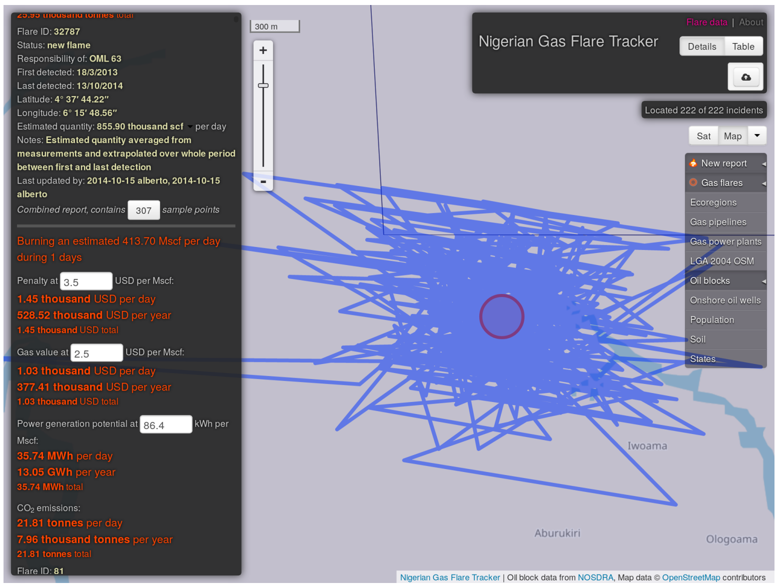
Flare ID: **32787**
Status: **new flame**
Responsibility of: **OML 63**
First detected: **18/3/2013**
Last detected: **13/10/2014**
Latitude: **4° 37′ 44.22″**
Longitude: **6° 15′ 48.56″**
Estimated quantity: **855.90 thousand scf** per day
Notes: **Estimated quantity averaged from measurements and extrapolated over whole period between first and last detection**
Last updated by: **2014-10-15 alberto, 2014-10-15 alberto**
Combined report, contains 307 *sample points*

Burning an estimated 413.70 Mscf per day

Nigerian Gas Flare Tracker | Oil block data from NOSDRA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community (Info)

Figure 1.1: Flare site composed of 307 measurements: the red circle is the estimated flare site location, and the blue lines connect each measurement from one day to the next. *(GFT 2014)*

My algorithm started by taking the measurements from the first day as initial flare sites. Then for each next day it found the distances between the new measurements and the flare sites and computed their statistical mode (smoothed by bucketing) for each axis (Δx,Δy), and for each measurement computed the corrected coordinates applying (-Δx,-Δy) and assigned it to the closest flare site if found within a radius of 0.0445 degrees (<5 Km, guessed by looking at a distance distribution graph), or stored it as a new flare site otherwise. Finally, the position of each flare site was the average of the positions of the measurements assigned to it, and it did another pass using these flare sites as initial to reduce the influence of outliers that might happen to be at the beginning of the list of measurements, the ones used as initial in the first pass.


25.95 thousand tonnes total

Flare ID: **32787**
Status: **new flame**
Responsibility of: **OML 63**
First detected: **18/3/2013**
Last detected: **13/10/2014**
Latitude: **4° 37′ 44.22″**
Longitude: **6° 15′ 48.56″**
Estimated quantity: **855.90 thousand scf** per day
Notes: **Estimated quantity averaged from measurements and extrapolated over whole period between first and last detection**
Last updated by: **2014-10-15 alberto, 2014-10-15 alberto**
Combined report, contains 307 *sample points*

Burning an estimated 413.70 Mscf per day during 1 days

Penalty at 3.5 USD per Mscf:
1.45 thousand USD per day
528.52 thousand USD per year
1.45 thousand USD total

Gas value at 2.5 USD per Mscf:
1.03 thousand USD per day
377.41 thousand USD per year
1.03 thousand USD total

Power generation potential at 86.4 kWh per Mscf:
35.74 MWh per day
13.05 GWh per year
35.74 MWh total

CO_2 emissions:
21.81 tonnes per day
7.96 thousand tonnes per year
21.81 tonnes total

Flare ID: **81**

Nigerian Gas Flare Tracker | Oil block data from NOSDRA, Map data © OpenStreetMap contributors

Figure 1.2: Clicking on the sample count ("Combined report, contains 307 sample points") expands the detail view (left pane) to show each sample point in full detail. *(GFT 2014)*

1.1 Remote pyrometry, detecting fire from orbit

The Gas Flare Tracker depends completely on the remote pyrometry measurements provided openly by the Earth Observation Group of the United States National Oceanic and Atmospheric Administration, NOAA. So much so, that in early 2019 when NOAA's web service was down during the US Government shutdown, the application stopped being able to offer current data. We kept serving the historic data we had until then, of course.

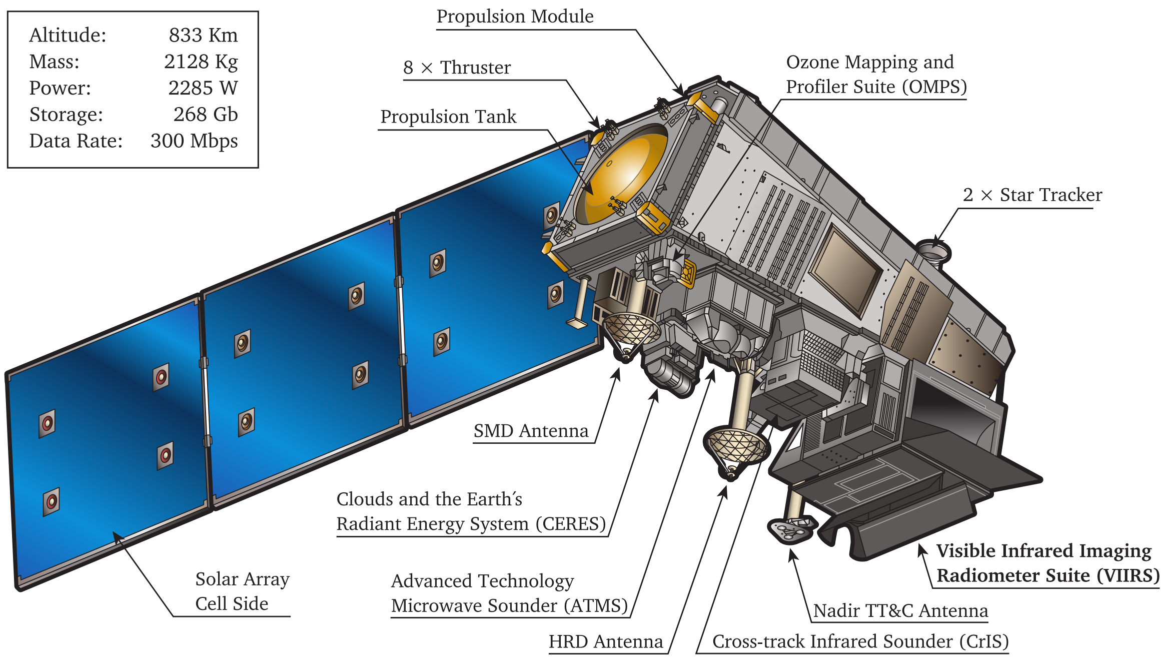

Altitude: 833 Km
Mass: 2128 Kg
Power: 2285 W
Storage: 268 Gb
Data Rate: 300 Mbps
Propulsion Module
8 × Thruster
Propulsion Tank
Ozone Mapping and Profiler Suite (OMPS)
2 × Star Tracker
SMD Antenna
Clouds and the Earth's Radiant Energy System (CERES)
Solar Array Cell Side
Advanced Technology Microwave Sounder (ATMS)
HRD Antenna
Visible Infrared Imaging Radiometer Suite (VIIRS)
Nadir TT&C Antenna
Cross-track Infrared Sounder (CrIS)

Figure 1.3: The Suomi NPP satellite. The images taken each night by its VIIRS instrument are processed by NOAA's Earth Observation Group into their VIIRS Nightfire [EZHB13] product that provides us with the daily fire location and temperature information. On 2017-11-18 its successor JPSS-1/NOAA-20 joined it with an improved VIIRS orbiting 50 minutes ahead. *Image from NASA's Press Kit, modified.*

In July 2014 while checking the calculations by myself I found a consistent 1.73 % discrepancy in the CO_2 emissions. In one example, a Methane_EQ of 0.681103 m^3/s gave me 1225.68495 g/s of CO_2 (Methane_EQ×**656**×**2.74323**), while NOAA's value was 1247.34 g/s.

- First we need the combusion reaction of methane: $CH_4 + 2\,O_2 \rightarrow CO_2 + 2\,H_2O$
- Now their weight ratio, from their molecular weights:
  - $CH_4 = 12.011 + 4\times1.00794 = 16.04276$
  - $CO_2 = 12.011 + 2\times15.999 = 44.009$

  Weight ratio $CO_2/CH_4 = 44.009/16.04276 = 2.7432312145790374 \simeq \mathbf{2.74323}$
  Therefore, the amount of CO_2 produced by combustion of one gram of CH_4 is 2.74323 g
- Given that the density of methane is 0.656 g/L at 25°C, 1 atm:
  $1L = 10E\text{-}3\ m^3 \Rightarrow 0.656\ g/L = \mathbf{656}\ g/m^3$

I wrote to their Defense Meteorological Satellite Program detailing my steps and they replied with a full account showing that the difference came from me using the density of methane at 25°C instead of 20°C as they did.

I keep being impressed by NOAA's and NASA's openness. That they provide unfettered access to such an amount and quality of data sources and even their own time to answer questions for the benefit of foreign citizens (Spanish, British, and Nigerian) stands in contrast with the attitude of other agencies like the European Space Agency[1] (ESA) and the German Aerospace Center[2] (Deutsches Zentrum für Luft- und Raumfart, DLR). I must admit that the USA does such things better.

[1]ESA "How to access ESA data", requires registration and accepting being tracked by Google.
[2]DLR "Data Access and Products", requires registration and approval by them of your usage.

1.2 2018 edition

This 2018 edition is a new application designed from the ground up for this purpose. It still uses the remote pyrometry data provided by the VIIRS instrument (§ 1.1 *Remote pyrometry, detecting fire from orbit*) with a new clustering and calibration algorithm by Rory Hodgson [Hod18].

The database management system is a standard SQL Relational Data Base Management System (RDBMS) with extensions for Geographical Information Data (GIS) management: PostgreSQL with the PostGIS extension.

The application front-end that runs on the server is, like in the OSM, written in Javascript with HTML, CSS and SVG. The map component is Leaflet, and graphs are built on Charts.js.

To bridge the two, there is a thin adapter (see § 6 *Developing the server API*) running on the Apache web server, written in PHP. The reason for choosing PHP for the HTTP API end-points (the URLs used by the front-end to fetch data) is to keep maintenance as low as possible: PHP is already pre-installed practically everywhere, and each request is served by a different instance of the interpreter that disappears after delivering the response. This means we do not have to install and update it, and there is no server process that has to be kept running.

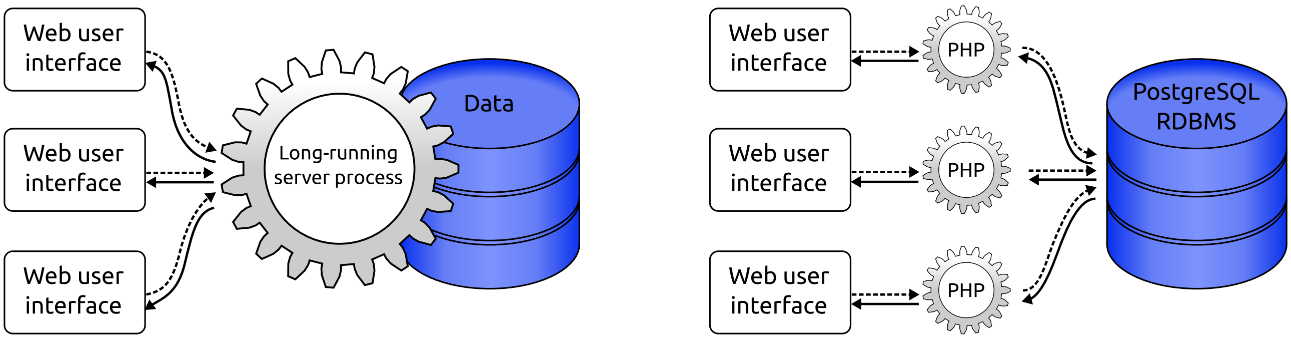


Figure 1.4: Two designs for web applications: a long-running process that keeps the data loaded, and short-running processes (here PHP programs) that start, serve one request, and stop.

In contrast, a long-running process would be more efficient as it could share resources among all requests, but would have to be correctly installed and maintained.

In this case, since the heavy lifting is done by PostgreSQL, the impact in the overall system performance is tiny and well worth the price to get a simpler to maintain and extend application.

1.3 Front-end features

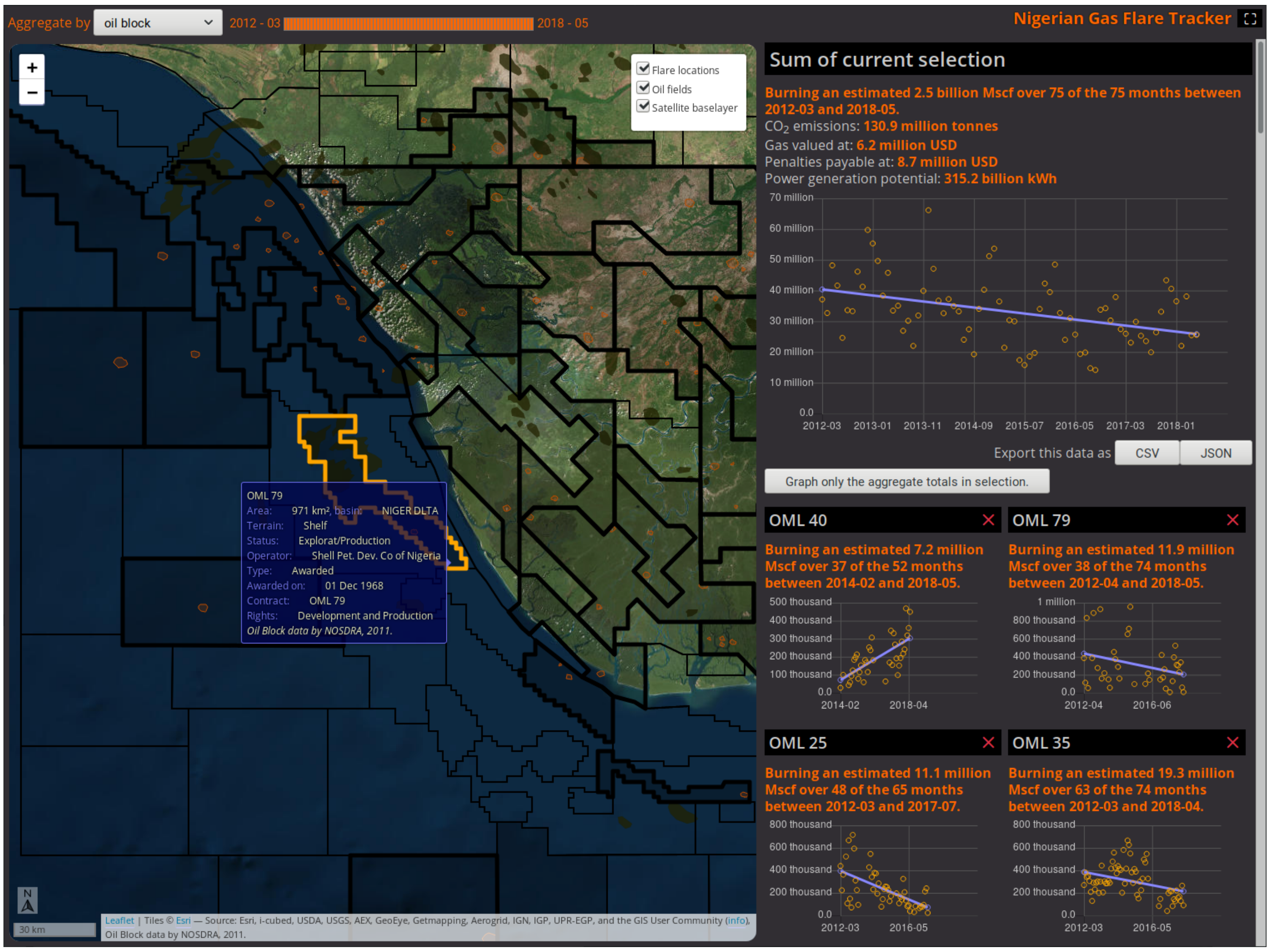
1. Data filter by date interval, aggregated by either: State, LGA, Oil block, Flare site / cluster identified by the method described in [Hod18], Company, Oil field, or Offshore/onshore areas. See § 2.1 *Filter*
2. Interactive layered map view. See § 2.2 *Map view*
3. Detail view with graphs and trend lines, including the aggregate values and also separate graphs for each aggregation field, for instance each state. See § 2.3 *Detail view*

1.4 Back-end features

1. Data storage and processing with PostgreSQL, described accurately in their website as “a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance”.
2. Automated merging of raw CSV data from VIIRS. See § 7.1 *Importing VIIRS data*
3. Simplified development of new HTTP end-points (queries) as detailed in § 6.2 *New HTTP endpoints with the PostgreSQL adaptor*.

User interface

The user interface is a web application. It has a filter bar at the top, a map view on the left side, and a detail view with graphs on the right side. The design is based on mock-ups provided by Rory Hodgson for SDN.



OML 79
Area: 971 km², basin: NIGER DLTA
Terrain: Shelf
Status: Explorat/Production
Operator: Shell Pet. Dev. Co of Nigeria
Type: Awarded
Awarded on: 01 Dec 1968
Contract: OML 79
Rights: Development and Production
Oil Block data by NOSDRA, 2011.

Sum of current selection

Burning an estimated 2.5 billion Mscf over 75 of the 75 months between 2012-03 and 2018-05.
CO_2 emissions: **130.9 million tonnes**
Gas valued at: **6.2 million USD**
Penalties payable at: **8.7 million USD**
Power generation potential: **315.2 billion kWh**

OML 40
Burning an estimated 7.2 million Mscf over 37 of the 52 months between 2014-02 and 2018-05.

OML 79
Burning an estimated 11.9 million Mscf over 38 of the 74 months between 2012-04 and 2018-05.

OML 25
Burning an estimated 11.1 million Mscf over 48 of the 65 months between 2012-03 and 2017-07.

OML 35
Burning an estimated 19.3 million Mscf over 63 of the 74 months between 2012-03 and 2018-04.

Figure 2.1: Filter (§ 2.1 *Filter*) at the top, map (§ 2.2 *Map view*) on the left, details (§ 2.3 *Detail view*) on the right. The big graph top-right shows the aggregate for the selected areas, the smaller graphs below show the evolution of each area.

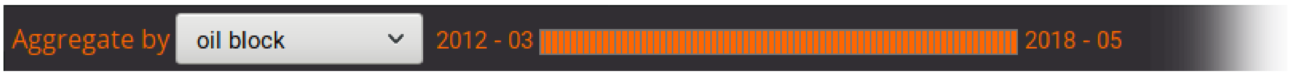


Figure 2.2: Close-up of the area and date range filters.

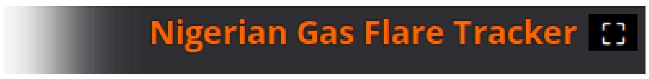


Figure 2.3: Full-screen toggle at the top right corner.

2.1 Filter

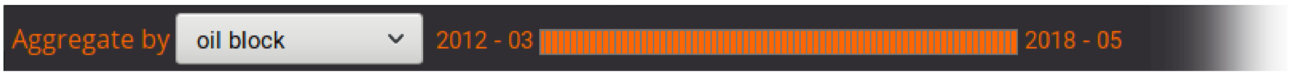


Figure 2.4: Query/filter selectors, area class and date range.

The satellite measurments can be aggregaged according to different areas:

- **state** the states in Nigeria.
- **lga** Local Government Administrations in Nigeria, roughly analog to provinces in other countries.
- **oil block** the oil exploitation blocks granted to different companies by the Nigerian government.
- **flare site** clusters of detected flares. The detection position from the satellite is not perfectly accurate and this clustering groups together detections that can reasonably assumed to belong to a same flare location.
- **company** as a given company can hold several oil blocks, this aggregation groups together all known sites operated by each company.
- **oil field** these are areas where oil/gas is found. Some fields span several oil blocks, and many oil blocks contains several fields.
- **onshore/offshore** splits flare locations on land from those off the coast.

Date range selector

The date slider selects the month range.

Clicking anywhere on the line will move the closest segment extreme there, and click-and-dragging will move the extreme closest to the button press point to the place where the button is released, very similar to click-and-dragging that extreme except that you do not need to click exactly on it: any point in the vecinity will do.

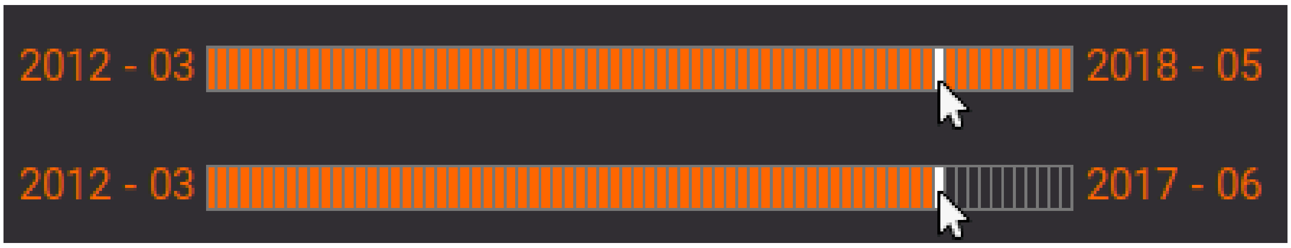
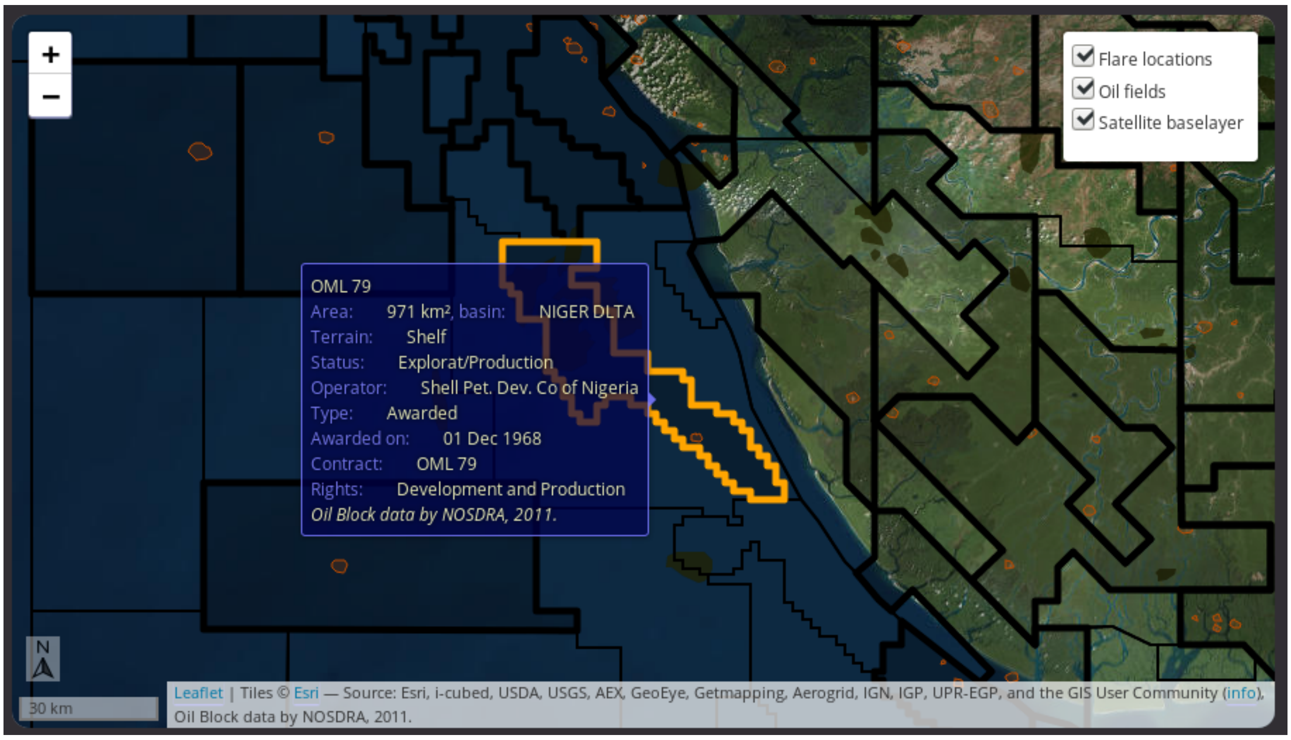


Figure 2.5: Date range selector: click or click-and-drag to move any of the range extremes.

Holding the Control key while clicking on the slider selects only that month.

2.2 Map view



OML 79
Area: 971 km², basin: NIGER DLTA
Terrain: Shelf
Status: Explorat/Production
Operator: Shell Pet. Dev. Co of Nigeria
Type: Awarded
Awarded on: 01 Dec 1968
Contract: OML 79
Rights: Development and Production
Oil Block data by NOSDRA, 2011.

Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community (info), Oil Block data by NOSDRA, 2011.

Figure 2.6: Map view, with an information pop-up showing the properties of the oil block under the cursor.

The shaded areas are those where flares were detected, and are the ones that can be selected/de-selected by clicking on them.

At the bottom left there is a scale: it corresponds to the *center* of the map, but in Nigeria there is very little distortion in the Mercator projection we use so it is a good approximation anywhere on the map.

The layer selector at the top right allows hiding certain layers for clarity.

When moving the mouse cursor over the map, a blue box shows the properties of the area under it. The information shown depends on the data available for that area in the database.

2.3 Detail view

It is divided in a top area for the aggregate totals, and smaller panels below for each of the selected areas.

A couple of buttons in the aggregate area allow downloading the currently-selected data as either JSON or CSV data that can be loaded in other tools like spreadsheets for inclusion in reports or further analysis.

The *blue line* is the trendline of the data.

Small graphs for each area

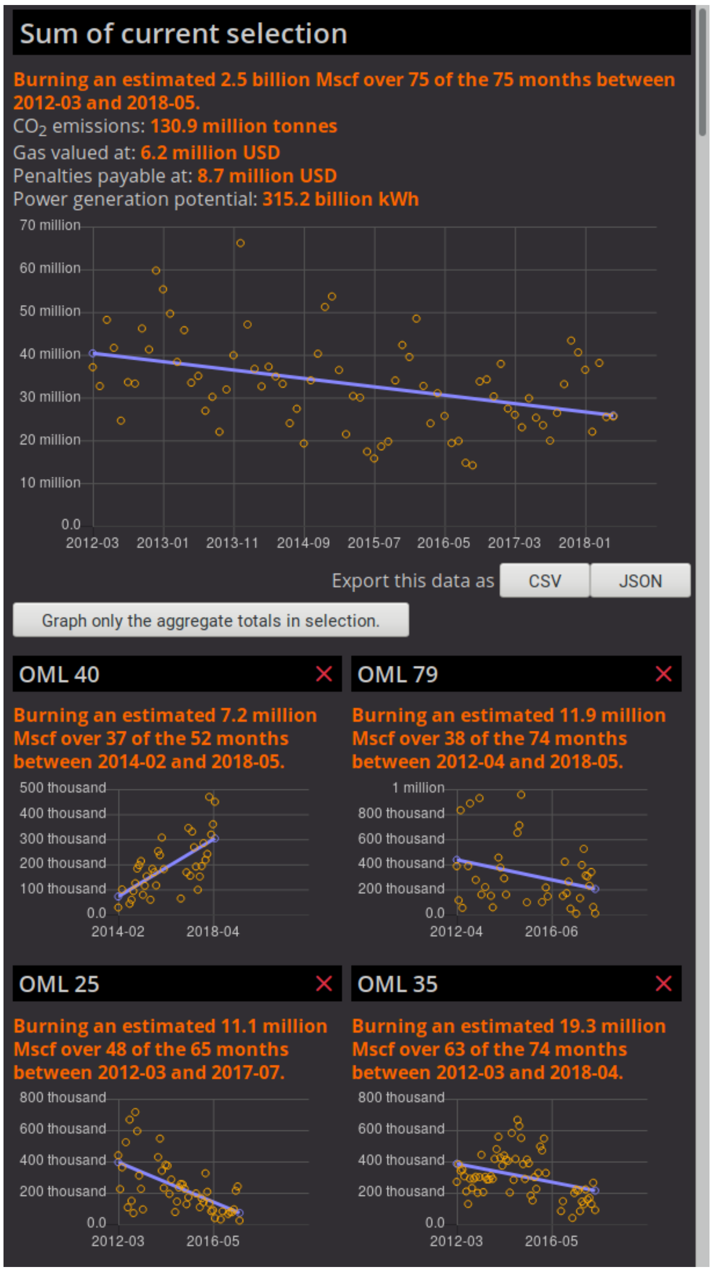
The individual graphs for selected areas are half the width of the aggregate to clearly distinguish them from the main aggregate graph.

Clicking on one of those panels (anywhere, title or graph works the same) will center and highlight it on the map.

The *cross mark* at the right of each title will de-select it.

When first loading a dataset, there is a button to select automatically all the areas where flares were detected. This is not done automatically because in some cases, for instance when grouping by LGA, there are many areas and it is more useful to start from scratch.

It is particularly useful when the areas are not easy to spot on the map at the initial zoom level. For instance when looking at oil fields, there are few with flares in them, and they are quite small in area.



Sum of current selection

Burning an estimated 2.5 billion Mscf over 75 of the 75 months between 2012-03 and 2018-05.
CO_2 emissions: 130.9 million tonnes
Gas valued at: 6.2 million USD
Penalties payable at: 8.7 million USD
Power generation potential: 315.2 billion kWh

OML 40

Burning an estimated 7.2 million Mscf over 37 of the 52 months between 2014-02 and 2018-05.

OML 79

Burning an estimated 11.9 million Mscf over 38 of the 74 months between 2012-04 and 2018-05.

OML 25

Burning an estimated 11.1 million Mscf over 48 of the 65 months between 2012-03 and 2017-07.

OML 35

Burning an estimated 19.3 million Mscf over 63 of the 74 months between 2012-03 and 2018-04.

Figure 2.7: The detail view.

Server API

All the server API endpoints reply in JSON (MIME type `application/json`), and any error messages include a list of valid parameters.

```
[
    {"error": [
        {"missing-parameters": ["area"]},
        "No matching SQL query for HTTP parameters",
        {"unknown-format": false},
        {"parameters": {
            "area": ["country", "state", "lga", "block"]
        }}
    ]}
]
```

Figure 3.1: Example of an error response, indicating that there is an `area` parameter whose value must be one of `country`, `state`, `lga`, or `block`.

3.1 Endpoints

feature-collection.php

fetch a GeoJSON feature collection from the goemetry objects in the database.

`table`= name of the table in PostgreSQL that contains the geometric features, for instance `boundary`, `states`, `lgas`, `blocks`, `company`, `location`, `oilfields`, `onshore_offshore`, `cluster_boundaries`, `population`, or `clusters`.

mscf.php

estimated monthly amounts of gas being burned for a geographical area, with fields `name`, `mscf`, and `month`.

`area`= either `country`, `state`, `lga`, `block`, `cluster`, `company`, `oilfield`, or `onshore_offshore`.

Installation

Just copy the "`gft-2018`" directory somewhere in the web application's content directory, and copy "`api/conf/db.php.example`" into "`api/conf/db.php`". You need to have PostgreSQL v9.5 at least.

4.1 Database set-up

The database must have two users defined:

- `gft` with `SELECT` privileges on the tables.
- `gftwrite` with all privileges on the database, capable of creating table views and inserting data when importing VIIRS measurements.

This separation is for security: the `gft` account only needs to read the data to serve it to the front-end, while the `gftwrite` account is not reachable from the web and is only used by the CLI tools that import VIIRS data and update the data views.

The password for the `gft` user must be put into "`api/conf/db.php`" (see § 5 *Configuration*), and the password for `gftwrite` is given on the command line during maintenance as described in § 7.1 *Importing VIIRS data* and § 7.2 *Updating the database*.

```
gft2018=# grant SELECT on all tables in schema public to gft;
gft2018=# grant SELECT on all sequences in schema public to gft;
gft2018=# alter default privileges in schema public
  grant SELECT on tables to gft;
gft2018=# alter default privileges in schema public
  grant SELECT on sequences to gft;

gft2018=# grant all privileges on database "gft2018" to gftwrite;
gft2018=# grant all privileges on all tables
   in schema public to gftwrite;
gft2018=# grant all privileges on all sequences
   in schema public to gftwrite;
gft2018=# alter default privileges in schema public
   grant all privileges on tables to gftwrite;
gft2018=# alter default privileges in schema public
   grant all privileges on sequences to gftwrite;
```

Figure 4.1: Setting the user privileges in PostgreSQL: gft can only read, and gftwrite can create and modify tables.

Configuration

The server can be configured in the files under “`api/conf/`”.

1. “`api/conf/`**`general.php`**” sets general options such as whether to display server error messages to the user or only put them in the Apache web server log file.

   It also defines utility functions to be used in other configuration files:

   `conf_set``(`**`$name`**`, `**`$value`**`)` set this entry in the global configuration.

   `conf_get``(`**`$name`**`)` get and entry from the global configuration.

2. “`api/conf/`**`db.php`**” tells how to access the PostgreSQL RDBMS.

   `dbconn_host`= host machine where it is running.

   `dbconn_dbname`= database name, because a RDBMS can store several databases.

   `dbconn_user`= default user for read operations: this should be a user that is only allowed to read data, not modify it. The scripts that need to write to the database override these settings to log in as a user with write privileges. See § 4.1 *Database set-up*.

   `dbconn_password`= the password for `dbconn_user`.

   It defines the function `dbconn_params()` that combines those settings into the string needed by `pg_connect(...)`.

```php
<?php
conf_set('dbconn_host', 'localhost');
conf_set('dbconn_dbname', 'gft-database');
conf_set('dbconn_user', 'gft-user');
conf_set('dbconn_password', '1234');

function dbconn_params()
{
  return
  'host='.conf_get('dbconn_host')
  .' dbname='.conf_get('dbconn_dbname')
  .' user='.conf_get('dbconn_user')
  .' password='.conf_get('dbconn_password')
  ;
}
?>
```

Developing the server API

6.1 CORS settings

Cross-Origin Resource Sharing allows using this API from a front-end in a different server. There is a small helper for handling CORS at “`lib/cors.php`”.

```
<?php require_once 'lib/cors.php';
      cors_allowed(['GET' => '*']);
      ... ?>
```

Figure 6.1: Example of allowing `GET` requests from any website.

6.2 New HTTP endpoints with the PostgreSQL adaptor

To help build new end-points/queries with a minimum of boilerplate, there is an adaptor under “`get_json.php`”.

It needs four defintions: `parameters`, `sql_parameters`, `description_response`, and `sql_queries`. From that, it produces a valid JSON response served over HTTP that is ready to use by the web front-end or other applications.

One example is “`feature-collection.php`” shown abbreviated in Figure 6.2. It provides the different data map layers such as “States” and “Oil fields”, called “feature collections” in GeoJSON.

`sql_queries` contains the list of possible queries, each of them with three fields:

`'params'`: is a pattern to match the request parameters. This entry will be picked if each paramenter mentioned here has the specified value.

`'format'`: is the format for the HTTP response.

If it is **`JSON`**, the response will be an array with one entry per data row, using the column names as field names.

If **`NDJSON`**, the result will be Newline Delimited JSON, that is, instead of an array of objects, there will be one line per object.

`GeoJSON` will produce a GeoJSON feature collection, with one feature per row. The table must have a `gid` column with unique identifiers, and a `geom` column of a geometry type that can be converted to GeoJSON using ST_AsGeoJSON() (that means a SFS 1.1 geometry type).

`'query'`: is the SQL query to be used if the request parameters match `params`.

```
<?php ...
$parameters = [ ['name' => 'table',  'default' => null] ];
$sql_parameters = [ ];

$sql_queries = [

['params' => ['table' => 'states'],
 'format' => 'GeoJSON',
 'query' => 'SELECT * FROM state_boundaries'
],
['params' => ['table' => 'lgas'],
 'format' => 'GeoJSON',
 'query' => 'SELECT * FROM lga_boundaries'
],
['params' => ['table' => 'blocks'],
 'format' => 'GeoJSON',
 'query' => 'SELECT id AS gid, block_name, status, type, total_sqkm, award_date,
             basin, contract, rights, terrain, operator, wkb_geometry AS geom
             FROM oil_blocks'
],
...
];

...

get_json_row($parameters, $sql_parameters, $description_response, $sql_queries);
?>
```

Figure 6.2: The most relevant parts of "`feature-collection.php`". The actual SQL query to use is picked according to its `'params'` entry: if the `'table'` HTTP parameter is `'states'` the first `'query'` will be used, and so forth.

Maintenance

This application is designed to require little maintenance, just refreshing the database regularly using the VIIRS data import script (§ 7.1 *Importing VIIRS data*), and the update script (§ 7.2 *Updating the database*) that rebuilds the data views used to speed things up.

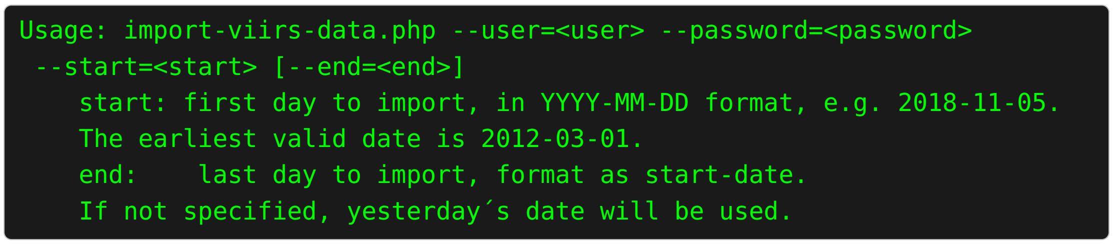
None of those steps requires stopping the web server or restarting anything.

7.1 Importing VIIRS data

`api/cli/`**`import-viirs-data.php`**

The VIIRS data import tool can run while the server continues using the existing data.

The first line printed is either the minimum and maximum dates in the data as "`Existing data ranges from 2012-03-01 to 2018-11-28`", or if there are no records yet which is normally the case the first time this script is run, the message "`The table viirs_merged is empty.`".

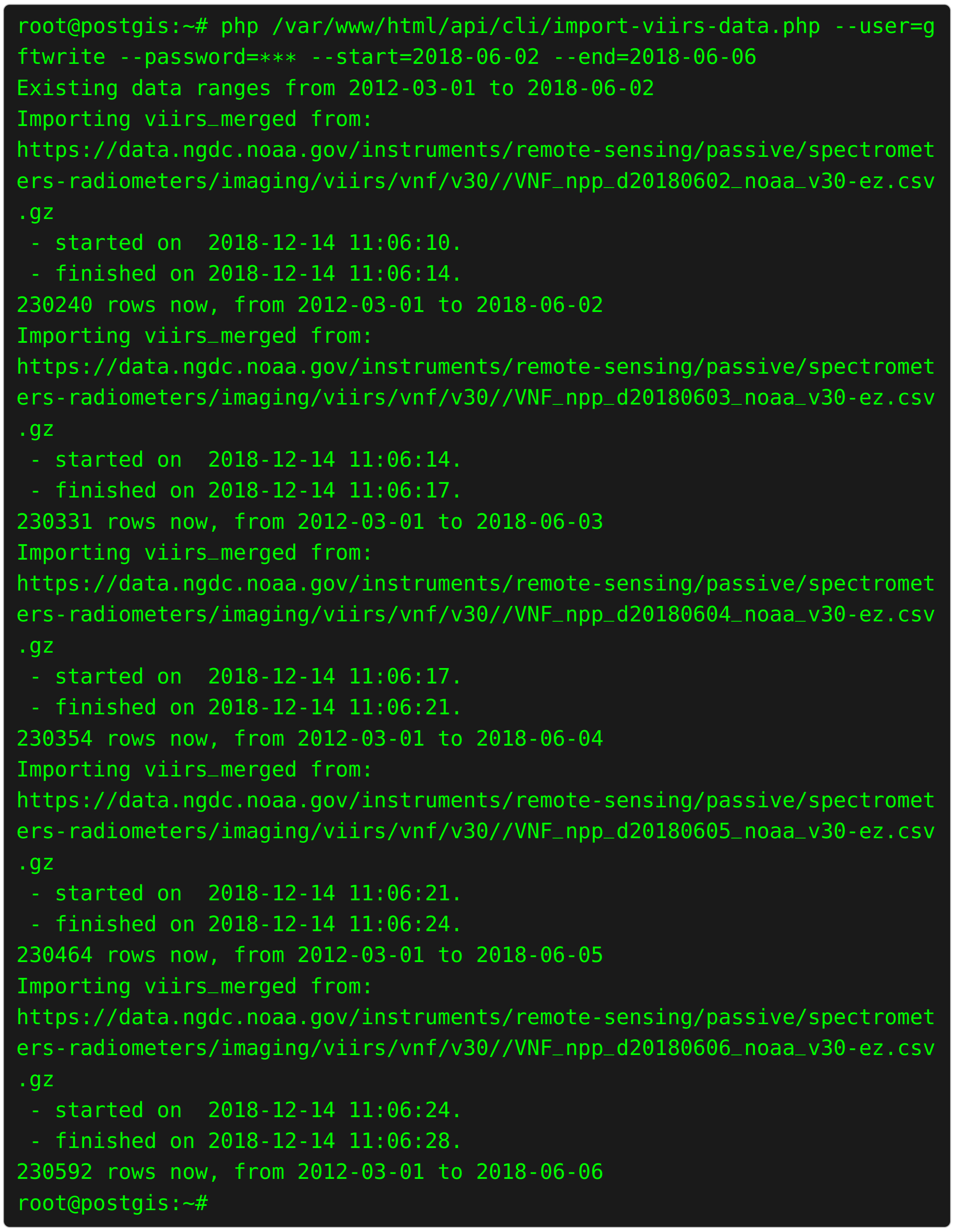


```
Usage: import-viirs-data.php --user=<user> --password=<password>
 --start=<start> [--end=<end>]
    start: first day to import, in YYYY-MM-DD format, e.g. 2018-11-05.
    The earliest valid date is 2012-03-01.
    end:    last day to import, format as start-date.
    If not specified, yesterday´s date will be used.
```

Figure 7.1: Screenshot of usage message.

Data URLs

The URLs for the VIIRS data files have changed over time. The script will compute the correct URL for each date, but this does not cover future changes which will need adjustments in the conditionals at the beginning of the `import()` function. It will also deal with small changes in the data files from different periods, which have had fields added over time.

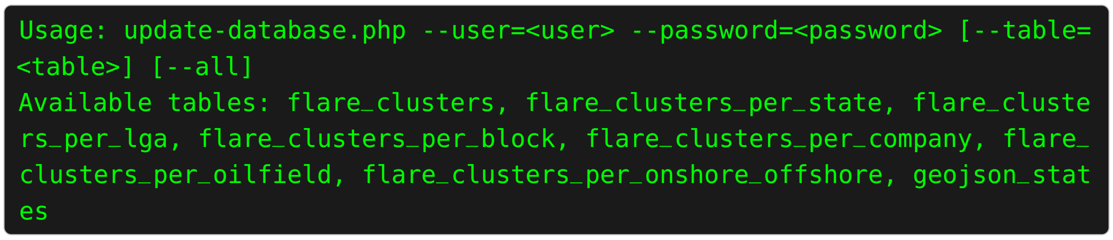


```
root@postgis:~# php /var/www/html/api/cli/import-viirs-data.php --user=g
ftwrite --password=*** --start=2018-06-02 --end=2018-06-06
Existing data ranges from 2012-03-01 to 2018-06-02
Importing viirs_merged from:
https://data.ngdc.noaa.gov/instruments/remote-sensing/passive/spectromet
ers-radiometers/imaging/viirs/vnf/v30//VNF_npp_d20180602_noaa_v30-ez.csv
.gz
 - started on  2018-12-14 11:06:10.
 - finished on 2018-12-14 11:06:14.
230240 rows now, from 2012-03-01 to 2018-06-02
Importing viirs_merged from:
https://data.ngdc.noaa.gov/instruments/remote-sensing/passive/spectromet
ers-radiometers/imaging/viirs/vnf/v30//VNF_npp_d20180603_noaa_v30-ez.csv
.gz
 - started on  2018-12-14 11:06:14.
 - finished on 2018-12-14 11:06:17.
230331 rows now, from 2012-03-01 to 2018-06-03
Importing viirs_merged from:
https://data.ngdc.noaa.gov/instruments/remote-sensing/passive/spectromet
ers-radiometers/imaging/viirs/vnf/v30//VNF_npp_d20180604_noaa_v30-ez.csv
.gz
 - started on  2018-12-14 11:06:17.
 - finished on 2018-12-14 11:06:21.
230354 rows now, from 2012-03-01 to 2018-06-04
Importing viirs_merged from:
https://data.ngdc.noaa.gov/instruments/remote-sensing/passive/spectromet
ers-radiometers/imaging/viirs/vnf/v30//VNF_npp_d20180605_noaa_v30-ez.csv
.gz
 - started on  2018-12-14 11:06:21.
 - finished on 2018-12-14 11:06:24.
230464 rows now, from 2012-03-01 to 2018-06-05
Importing viirs_merged from:
https://data.ngdc.noaa.gov/instruments/remote-sensing/passive/spectromet
ers-radiometers/imaging/viirs/vnf/v30//VNF_npp_d20180606_noaa_v30-ez.csv
.gz
 - started on  2018-12-14 11:06:24.
 - finished on 2018-12-14 11:06:28.
230592 rows now, from 2012-03-01 to 2018-06-06
root@postgis:~#
```

Figure 7.2: A typical run of the VIIRS data import script.

7.2 Updating the database

api/cli/**update-database.php**
This re-builds the *materialized views* (cached computed data tables) that are used for serving requests.



```
Usage: update-database.php --user=<user> --password=<password> [--table=
<table>] [--all]
Available tables: flare_clusters, flare_clusters_per_state, flare_cluste
rs_per_lga, flare_clusters_per_block, flare_clusters_per_company, flare_
clusters_per_oilfield, flare_clusters_per_onshore_offshore, geojson_stat
es
```

Figure 7.3: Screenshot of usage message.

--table= **flare_clusters** performs the SQL query provided by Rory Hodgson to identify flare clusters from the raw VIIRS daily detections, and is based on his MSc Thesis [Hod18].

--table= **flare_clusters_per_state** is an aggregation by state of the flare locations identified in flare_clusters.

--table= **flare_clusters_per_lga** is the same as above, aggregated by Local Government Administration (LGA).

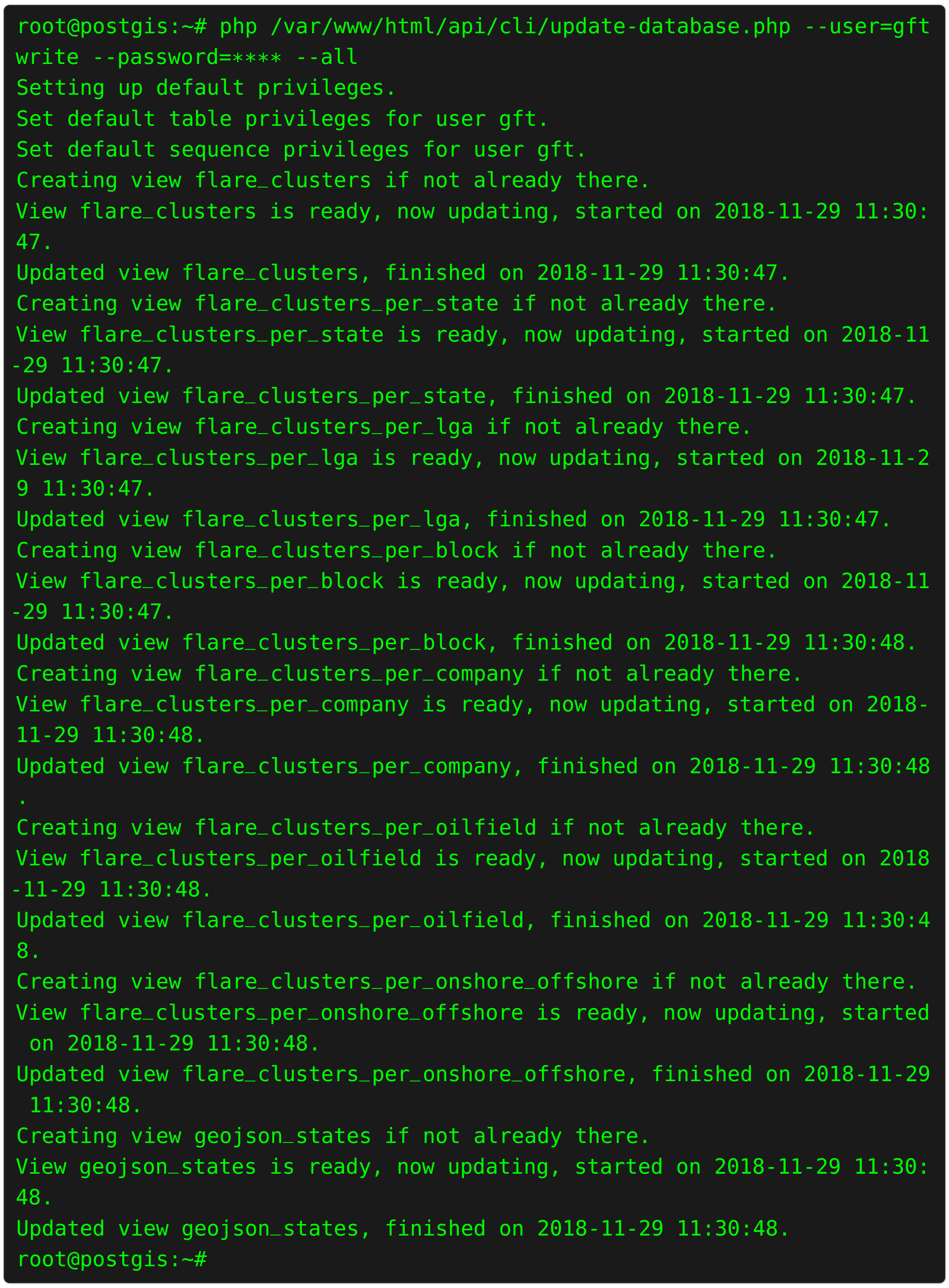
--table= **flare_clusters_per_block** is the aggregation per oil block.

--table= **flare_clusters_per_company** aggregates flare locations per company as identified by Rory Hodgson.

--table= **flare_clusters_per_oilfield** aggregates them per oil field.

--table= **flare_clusters_per_onshore_offshore** totalizes the flare clusters in two sets: over land or at sea.

--table= **geojson_states** is a cached version of the states table already converted to JSON, to avoid having to do the conversion for each page view.



```
root@postgis:~# php /var/www/html/api/cli/update-database.php --user=gft
write --password=**** --all
Setting up default privileges.
Set default table privileges for user gft.
Set default sequence privileges for user gft.
Creating view flare_clusters if not already there.
View flare_clusters is ready, now updating, started on 2018-11-29 11:30:
47.
Updated view flare_clusters, finished on 2018-11-29 11:30:47.
Creating view flare_clusters_per_state if not already there.
View flare_clusters_per_state is ready, now updating, started on 2018-11
-29 11:30:47.
Updated view flare_clusters_per_state, finished on 2018-11-29 11:30:47.
Creating view flare_clusters_per_lga if not already there.
View flare_clusters_per_lga is ready, now updating, started on 2018-11-2
9 11:30:47.
Updated view flare_clusters_per_lga, finished on 2018-11-29 11:30:47.
Creating view flare_clusters_per_block if not already there.
View flare_clusters_per_block is ready, now updating, started on 2018-11
-29 11:30:47.
Updated view flare_clusters_per_block, finished on 2018-11-29 11:30:48.
Creating view flare_clusters_per_company if not already there.
View flare_clusters_per_company is ready, now updating, started on 2018-
11-29 11:30:48.
Updated view flare_clusters_per_company, finished on 2018-11-29 11:30:48
.
Creating view flare_clusters_per_oilfield if not already there.
View flare_clusters_per_oilfield is ready, now updating, started on 2018
-11-29 11:30:48.
Updated view flare_clusters_per_oilfield, finished on 2018-11-29 11:30:4
8.
Creating view flare_clusters_per_onshore_offshore if not already there.
View flare_clusters_per_onshore_offshore is ready, now updating, started
 on 2018-11-29 11:30:48.
Updated view flare_clusters_per_onshore_offshore, finished on 2018-11-29
 11:30:48.
Creating view geojson_states if not already there.
View geojson_states is ready, now updating, started on 2018-11-29 11:30:
48.
Updated view geojson_states, finished on 2018-11-29 11:30:48.
root@postgis:~#
```

Figure 7.4: Screenshot of database update process.

Bibliography

[ABW] Obinna C.D. Anejionu, Alan Blackburn, and Duncan Whyatt, *Remote mapping of gas flares in the niger delta with modis imagery*, 33th EARSeL Symposium (Rosa Lasaponara, Nicola Masini, and Marilisa Biscione, eds.), Available online at http://www.earsel.org/symposia/2013-symposium-Matera/pdf_proceedings/EARSeL-Symposium-2013_2_2_anejionu.pdf, pp. 59–68. 1

[EZHB13] Christopher Elvidge, Mikhail Zhizhin, Feng-Chi Hsu, and Kimberly Baugh, *VIIRS nightfire: Satellite pyrometry at night*, Remote Sensing **5** (2013), no. 9, 4423–4449, available online at https://doi.org/10.3390/rs5094423. 3

[Hod18] Rory Hodgson, *Generating a scalable calibration equation that can be applied to viirs nightfire (vnf) radiant heat calculations to estimate gas flaring volumes in nigeria*, Master's thesis, Birkbeck College, University of London, 9 2018, Available online at https://docs.google.com/document/d/1Oc0YM9CaTRnBenlo3dZrjfx1LPn3BDgzT_SHgBBn5qs/edit. 4, 16