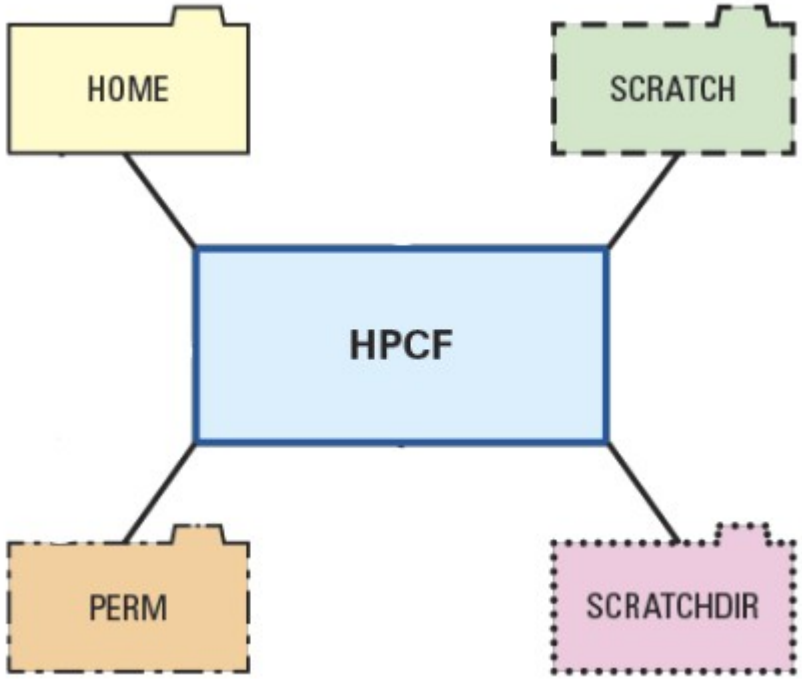
Geoclimate setup on HPC (cca - ccb)

Emmanuel Renault

IT Engineer.
Lab-STICC, CNRS

File Systems



Source: Carsten Maass – Introduction tio File Systems. ECMWF

Machines – accessible to users

Local file systems

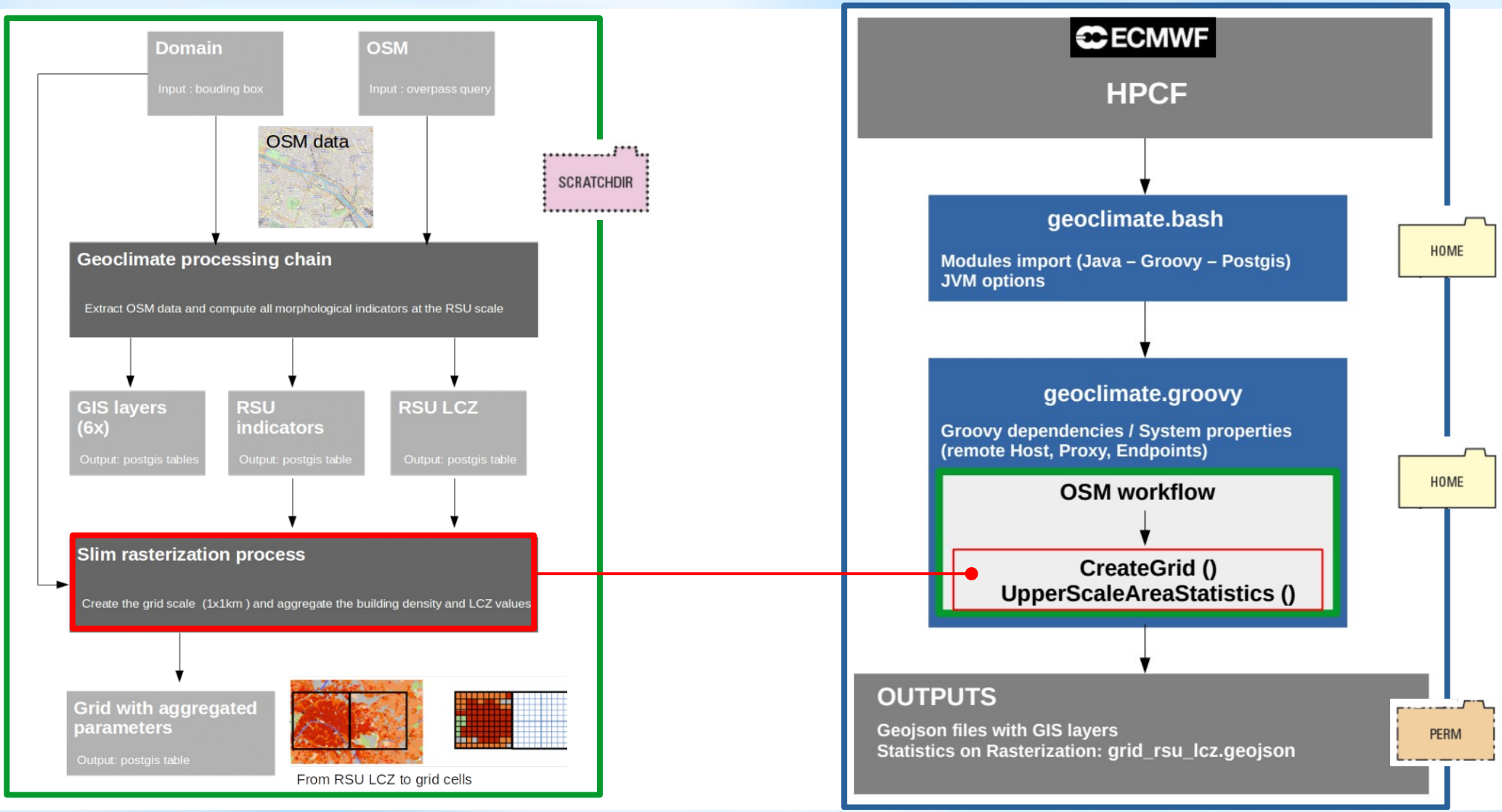
File systems – suitable for permanent files ('large' quota, no backup)

File systems – suitable for temporary data

File systems – directories automatically deleted at end of job

File systems – suitable for permanent files ('small' quota, backup)

Geoclimate workflow



Software requirements

- OpenJDK Runtime Environment (version 8)

- Groovy script language: https://groovy-lang.org/
  can be installed with the SDKMAN toolkit: https://groovy-lang.org/install.html

  using the following command lines (if not already installed on localhost):

```
> sdk list java
> sdk install java 8.0.262.hs-adpt
> sdk current

> sdk install groovy
 (sdk update)
```

- A PostGIS database if you have to store output data into tables.
  → comes with the setup of a Postgresql server.

  → See in Annex Documentation to handle the initialization process of the server

Environment - Bash

Modules setup has been made on the following path : `/perm/us/usxa/slim/modulefiles/`

To run the Geoclimate chain, several modules have to be available and loaded in a bash script:

```
module use -a perm/us/usxa/slim/modulefiles
module load java groovy postgis
```

⚠ java version must be at least : java version "1.8.0_51"
Java(TM) SE Runtime Environment (build 1.8.0_51-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.51-b03, mixed mode)

To set verbose compilation for **debugging**, one can set the following Groovy command line inside bash script:

```
-Dgroovy.grape.report.downloads=true -Divy.message.logger.level=4
```

Environment - Bash

Running the Geoclimate chain has to be performed on **HPC (cca - ccb)** servers.
Two operating modes are considered:

- **`INTERACTIVE`** environment
  which is the operational mode (for instance)

```
> ./geoclimate.bash
```

- **`NON-INTERACTIVE (BATCH)`** environment
  submitting jobs to the batch system, using PBS directives

```
> qsub geoclimate.bash
```

generated output files: `geoclimate.bash.[o/e]jobId`

Sources

Two scripts are used to launch the Geoclimate data processing chain. Their location must reside in the `$HOME` path.

- **`geoclimate.bash`**

  A bash script to launch the chain in interactive mode, calling the Groovy script below.

  ```
  > ./geoclimate.bash
  ```

- **`geoclimate.groovy`**

  The main script that calls the processes to be executed:

  - OSM workflow
  - createGrid
  - upperScaleAreaStatistics

Inputs

A list of bounding boxes in lat-lon coordinates (WGS84 projection System). File location has to be in the `$HOME` path.

- **`osmFilters.json`**

  a Json file that has been created with a Python script,
  to store the coordinates of each domain of 10x10 km².

  In a general way, the dataset has to be parsed
  according the following attributes:

  . `id` an integer representing the current domain to be processed.
  . `bbox` a list of lat-lon coordinates.
  . `N` an integer representing the number of domains to be processed.

Outputs

12 Geojson files per processed domain:

- 1 file that contains the Geodataframe of data rasterization (Metric System)
- 11 files that defines GIS layers

Below, the file containing the data from rasterization process :

- **`grid_rsu_lcz.geojson`**

  a Geojson file that contains the result of aggregation process on the gridded domain, for a given Geoindicator.

  For each processed domain, the output is stored in a sub-folder:
  `$PERM/osm/osm_[bbox_coordinates]`

Configuration

One can make a configuration to run the Geoclimate chain, by setting the `json` parameters inside the Groovy script itself: `geoclimate.groovy`

The osmFilters variable can be assigned as :

- a list of bounding boxes coordinates returned after parsing a `json` input file: `osmFilters.json`
- a list set manually :

```
"osmFilters" : [48.83426, 2.24943, 48.91759, 2.33276]
```

```
[
   "description" :"run the OSM workflow",
   "geoclimatedb" : [
           "folder" :'/tmp/osm',
           "name" : "geoclimate_db;AUTO_SERVER=TRUE",
           "delete" : true
   ],
   "input" : [
      "osmFilters": [48.83426, 2.24943, 48.91759, 2.33276]
   ],
   "output" : output = [
       "folder" : '/tmp/osm/',
       "tables": [
           "building_indicators",
           "block_indicators",
           "rsu_indicators",
           "rsu_lcz",
           "zones",
           "building",
           "road",
           "rail" ,
           "water",
           "vegetation",
           "impervious"
       ]
   ],
   "parameters": [
           "distance": 0,
           "indicatorUse": ["LCZ"],
           "svfSimplified": false,
           "prefixName": "",
           "hLevMin": 3,
           "hLevMax": 15,
           "hThresholdLev2": 10
   ]
]
```

Configuration (bis)

Download the file to get Geoclimate dependencies (.jar), to bypass any proxy if ever:

- https://jenkins.orbisgis.org/job/geoclimate-with-dependencies/

and **execute** the following command line to launch the chain :

- `java -jar Geoclimate.jar -f osmConfigFile.json`

where the configuration file (`osmConfigFile.json`) can be as simple as in this example :

```
{
    "description" :"Processing OSM data",
    "geoclimatedb" : {
        "folder" : "/tmp/osm",
        "name":"geoclimate_db_test;AUTO_SERVER=TRUE",
        "delete" :true
    },
    "input" : {
         "osmFilters" : [48.83426, 2.24943, 48.91759, 2.33276]
    },
    "output" :{
     "folder" :  "/tmp/osm"},
    "parameters":
    {
        "indicatorUse": ["LCZ"]
    }
}
```

Dependencies

To be operational, the Geoclimate chain has to be set up with Groovy dependencies management.

Geoclimate library is **automatically downloaded** from a remote repository, using a Groovy dependency resolver (Grab annotation @ → next slide).

Dependencies are stored in a local hidden folder: `~/.groovy/grapes`

```
cyem@cca-login4:~/.groovy/grapes> ls -lrth
total 20K
drwxr-x--- 3 cyem copext 4,0K 28 nov.  01:34 org.orbisgis.orbisprocess
drwxr-x--- 4 cyem copext 4,0K 28 nov.  01:34 org.codehaus.groovy
drwxr-x--- 3 cyem copext 4,0K 28 nov.  01:34 org.orbisgis.orbisdata.datamanager
drwxr-x--- 3 cyem copext 4,0K 28 nov.  01:38 org.orbisgis.orbisanalysis
drwxr-x--- 3 cyem copext 4,0K 28 nov.  01:38 org.orbisgis
```

⚠ Make sure to update Groovy dependencies by removing this hidden folder (it is automatically refreshed when recompiling the chain).

Dependencies

After having executed the Groovy script, the tree structure of Groovy dependencies must contain at least 3 file formats: `.jar`, `.xml`, `.properties`
Ex: Geoclimate library dependencies

- `geoclimate-1.0.0-SNAPSHOT-jar-with-dependencies.jar`
- `ivy-1.0.0-SNAPSHOT.xml`
- `ivydata-1.0.0-SNAPSHOT.properties`

geoclimate.groovy

```groovy
// MAVEN repository
@GrabResolver(name="orbisgis", root="https://nexus.orbisgis.org/repository/orbisgis/")

// GEOCLIMATE dependencies
@Grab(group="org.orbisgis.orbisprocess", module="geoclimate", version="1.0.0-SNAPSHOT",
      classifier="jar-with-dependencies", transitive=false)

// JSON dependencies
@Grab(group="org.codehaus.groovy", module="groovy-json", version="3.0.4")

// JDBC dependencies
@Grab(group="org.orbisgis.orbisdata.datamanager", module="jdbc", version="1.0.1-SNAPSHOT")

import org.orbisgis.orbisprocess.geoclimate.Geoclimate
import org.orbisgis.orbisdata.datamanager.jdbc.h2gis.H2GIS
import groovy.json.JsonOutput
import groovy.json.JsonSlurper
```

Endpoints

The Geoclimate chain uses 4 APIs, 1 of which allow access to a server in order to extract OSM data of an area defined from bounding lat-lon coordinates:

- https://overpass-api.de/api

The data extraction file is downloaded and then automatically removed from path `/tmp` after the execution process. It should have the following shape (example):

```
a38d5c1a6ac0be595c039323d61a7450a080a446cc9ad54e87c9b4ac868246e0.osm
```

System properties like `proxy` and `environment variables` have to be set in the bash script (`geoclimate.bash`), for the process to be able to download data from these externals APIs.

See the following link below, to get more information:

- Proxy settings

Launch test - Execution

INTERACTIVE

geoclimate.bash

```
#!/bin/bash
#
module use -a perm/us/usxa/slim/modules/modulefiles
module load java groovy postgis
#
groovy geoclimate.gvy
```

Input List

osmFilters.json

```
{
    "0": {
        "bbox": [
            48.83426,
            2.24943,
            48.91759,
            2.33276
        ]
    },
    "N": 1
}
```

Execution

```
cyem@cca-login4:~> ./geoclimate.bash
```

Launch test - Execution

BATCH

Execution

```
cyem@cca-login4:~> qsub geoclimate.bash
5010999.ccapar
```

```
cyem@cca-login4:~> qstat -u cyem

ccapar:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
5010999.ccapar  cyem     ns       geoclimate  42489   1   1 1024mb 48:00 R 00:02
```

Launch test – Data Extraction

```
[INFO] org.orbisgis.orbisdata.processmanager.process.GroovyProcessFactory - Reading file parameters from /perm/ms/copext/cyem/osm/osmConfigFile.json
[INFO] org.orbisgis.orbisdata.processmanager.process.GroovyProcessFactory - run the OSM workflow and store the results in /perm/ms/copext/cyem/osm/
[INFO] org.orbisgis.orbisdata.processmanager.process.GroovyProcessFactory - 1 osm areas will be processed
[INFO] org.orbisgis.orbisdata.processmanager.process.GroovyProcessFactory - Extract the OSM data
[INFO] org.orbisgis.orbisdata.processmanager.process.GroovyProcessFactory -
The cached OSM file /tmp/666aa23b2ffee787763e22b1eb475e1a257274f1b62b223716ee524657fdc04a.osm will be re-used for the query :
[maxsize:1073741824][bbox:48.83426,2.24943,48.91759,2.33276];
((
        node["building"];
        node["railway"];
        node["amenity"];
        node["leisure"];
        node["highway"];
        node["natural"];
        node["landuse"];
        node["landcover"];
        node["vegetation"];
        node["waterway"];
        way["building"];
        way["railway"];
        way["amenity"];
        way["leisure"];
        way["highway"];
        way["natural"];
        way["landuse"];
        way["landcover"];
        way["vegetation"];
        way["waterway"];
        relation["building"];
        relation["railway"];
        relation["amenity"];
        relation["leisure"];
        relation["highway"];
        relation["natural"];
        relation["landuse"];
        relation["landcover"];
        relation["vegetation"];
        relation["waterway"];
);
>;);
out;.
```

Launch test – Batch ressources report

```
## INFO ------------------------------------------------------------------------------
## INFO  This is the ECMWF job Epilogue. Please report problems to Service Desk, servicedesk@ecmwf.int
## INFO ------------------------------------------------------------------------------
## INFO
## INFO Run at Sat Nov 28 19:27:13 2020 on CCA
## INFO Job Name                   : geoclimate.bash
## INFO Job ID                     : 4811348.ccapar
## INFO Queued                     : Sat Nov 28 19:20:19 2020
## INFO Dispatched                 : Sat Nov 28 19:20:40 2020
## INFO Completed                  : Sat Nov 28 19:27:13 2020
## INFO Waiting in the queue       : 21 seconds
## INFO Runtime                    : 393 seconds
## INFO Exit Code                  : 0
## INFO Account                    : eccopext
## INFO Queue                      : ns
## INFO Owner                      : cyem
## INFO STDOUT                     : /home/ms/copext/cyem/geoclimate.bash.o4811348
## INFO STDERR                     : /home/ms/copext/cyem/geoclimate.bash.e4811348
## INFO Hyperthreads               : 2
## INFO SBU                        : 0.880417 units
## INFO Logical CPUs               : 1
## INFO Historic runtime average            : 302.00 seconds
## INFO Historic runtime standard deviation : 165.17 seconds
## INFO
## INFO   MOM RESOURCES USED | ncpus      cput       runtime    vmem       mem
## INFO ------------------------------------------------------------------------
## INFO   28.11.2020 - 19:27 | 1          32         393        34577784kb 804508kb
## INFO
## INFO Note: Historic runtime average is 302 seconds with a standard deviation of 165.2 seconds
## INFO This runtime falls within 1 standard deviation of average
## INFO
```

Launch test - Outputs

Intermediate datasources tables are stored in the database named `geoclimate_db`, on the following path: `perm/ms/copext/cyem/osm/geoclimate_chain`

```
cyem@cca-login4:/perm/ms/copext/cyem/geoclimate_chain> ls -lrth
total 596M
-rw-r----- 1 cyem copext 594M  1 déc.  01:56 geoclimate_db.mv.db
-rw-r----- 1 cyem copext 355K  1 déc.  01:56 geoclimate_db.trace.db
```

Computation outputs for Geoindicators and Rasterization are stored in the path: `$PERM/osm/osm_[bbox]`

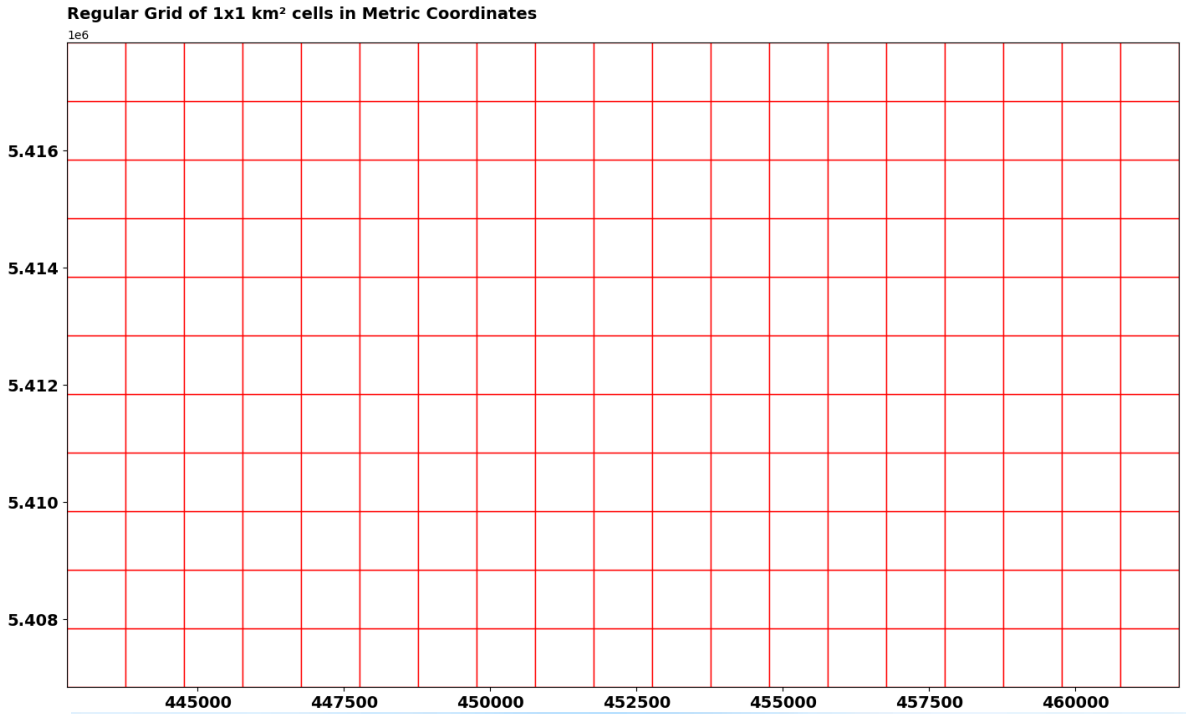
```
cyem@cca-login4:/perm/ms/copext/cyem/osm/osm_[48.83426, 2.24943, 48.91759, 2.33276]> ls -lrth
total 141M
-rw-r----- 1 cyem copext  58M 28 nov.  20:13 building_indicators.geojson
-rw-r----- 1 cyem copext  19M 28 nov.  20:13 rsu_indicators.geojson
-rw-r----- 1 cyem copext 3,6M 28 nov.  20:13 rsu_lcz.geojson
-rw-r----- 1 cyem copext  439 28 nov.  20:13 zones.geojson
-rw-r----- 1 cyem copext  51M 28 nov.  20:13 building.geojson
-rw-r----- 1 cyem copext 4,1M 28 nov.  20:13 road.geojson
-rw-r----- 1 cyem copext 455K 28 nov.  20:13 rail.geojson
-rw-r----- 1 cyem copext 369K 28 nov.  20:13 water.geojson
-rw-r----- 1 cyem copext 3,5M 28 nov.  20:13 vegetation.geojson
-rw-r----- 1 cyem copext 1,9M 28 nov.  20:13 impervious.geojson
-rw-r----- 1 cyem copext 106K 28 nov.  20:13 urban_areas.geojson
-rw-r----- 1 cyem copext  42K 30 nov.  21:31 grid_rsu_lcz.geojson
```

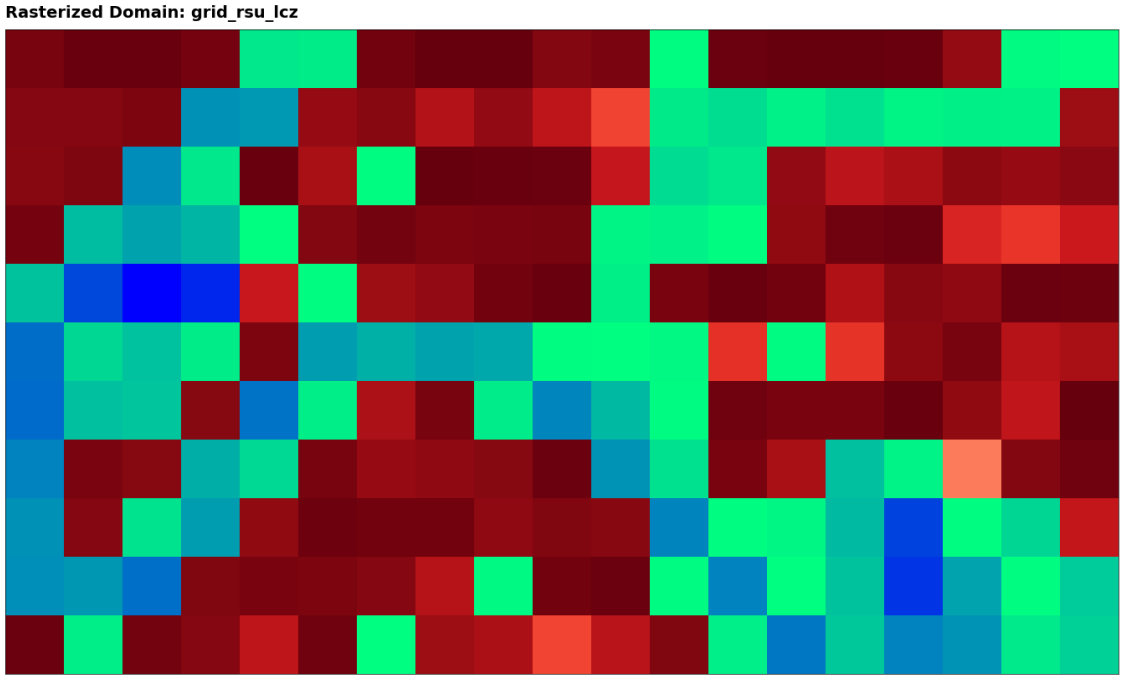
Launch test - Outputs

RASTERIZATION : LCZs Aggregation (Metric Coordinates)
Example of a test case with an area of 19x11 = 209 km²
Bbox used in lat-lon coordinates: [48.813420, 2.220440, 48.904449, 2.471581]

LEFT : Gridded area with 1x1 km² cells
RIGHT : LCZs spatial aggregation on the gridded area: `grid_rsu_lcz.geojson`

Paris Area





Annex

PostgreSQL Server

- Type the bash commands below to **initialize** the Postgres server:

```
INITIALIZING

export PATH=/perm/us/usxa/apps/postgis/3.0.1/bin:$PATH
mkdir -p $PERM/pdb_data
pg_ctl -D $PERM/pdb_data/ -l $PERM/postgres.log init
```

⚠ This is a one-off process

PostgreSQL Server

```
cyem@cca-login4:~> export PATH=/perm/us/usxa/slim/postgis/3.0.1/bin:$PATH
cyem@cca-login4:~> mkdir -p $PERM/pdb_data
cyem@cca-login4:~> pg_ctl -D $PERM/pdb_data/ -l $PERM/postgres.log init
The files belonging to this database system will be owned by user "cyem".
This user must also own the server process.

The database cluster will be initialized with locale "fr_FR.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "french".

Data page checksums are disabled.

fixing permissions on existing directory /perm/ms/copext/cyem/pdb_data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... GMT
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /perm/us/usxa/slim/postgis/3.0.1/bin/pg_ctl -D /perm/ms/copext/cyem/pdb_data -l logfile start

cyem@cca-login4:~>
```

PostgreSQL Server

- Type the bash commands below, in the Batch Job, to **start** the Postgres server:

```
STARTING CONNEXION TO SERVER

pg_ctl -D $PERM/pdb_data/ -l $PERM/postgres.log start
```

```
cyem@cca-login4:/perm/ms/copext/cyem> pg_ctl -D pdb_data/ -l postgres.log start
waiting for server to start.... done
server started
```

```
LOG:  starting PostgreSQL 12.3 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 7.3.0 20180125 (Cray Inc.), 64-bit
LOG:  listening on IPv4 address "127.0.0.1", port 5432
LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
LOG:  database system was shut down at 2020-12-02 11:46:46 GMT
LOG:  database system is ready to accept connections
```

PostgreSQL Server

- One can create a new database (`createdb geoclimate`) and access to it with the command below:

```
cyem@cca-login4:/perm/ms/copext/cyem> psql -d geoclimate
psql (12.3)
Type "help" for help.

geoclimate=#
```

- Type the bash commands below, in the Batch Job, to **stop** the Postgres server:

```
CLOSING CONNEXION TO SERVER

pg_ctl -D $PERM/pdb_data/ -l $PERM/postgres.log stop
```

⚠ To avoid database corruption