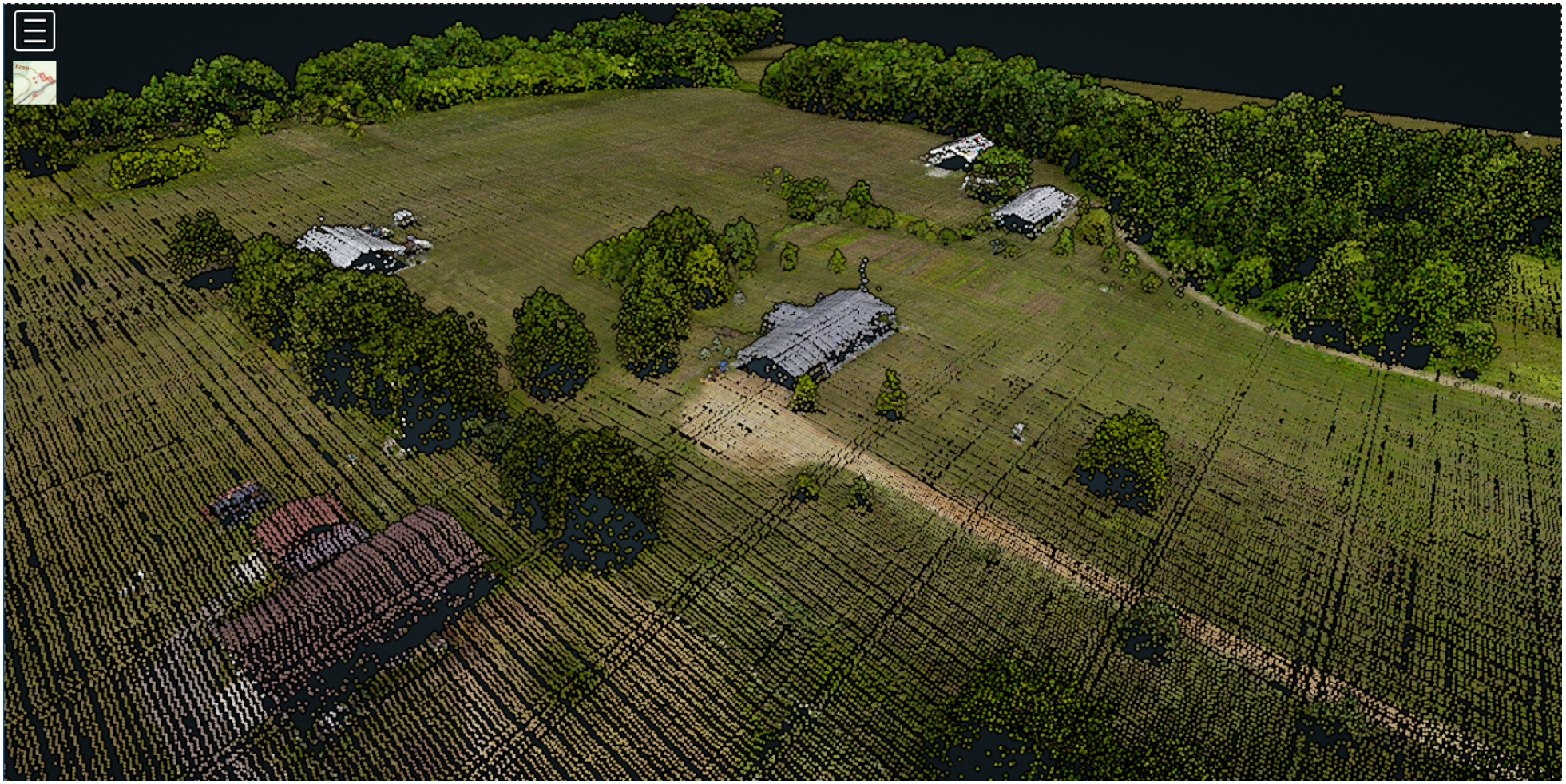
Nuages de Points

Concepts, chargement, stockage et manipulation

Ludovic Delauné PG Session #8 - Lyon

nuage de points ?

Large collection de points multi-dimensionnels
(x, y, z, time, RGB, intensity...)



Comment obtient-on des nuages de points ?

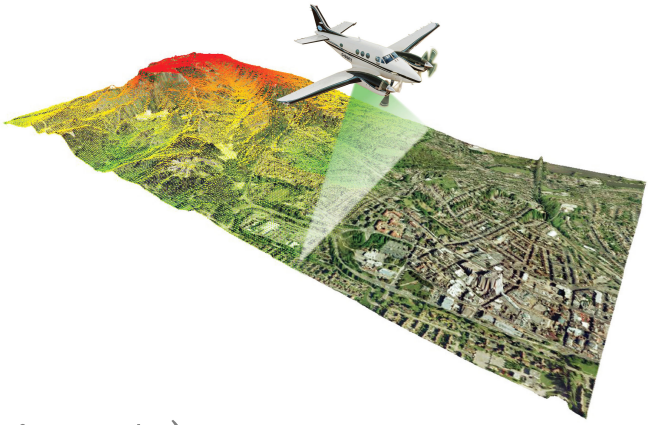
Photogrammétrie

Lasergrammétrie

(LIDAR : Light Detection And Ranging)

Scanner Laser

Au sol, embarqué ou aéroporté



Principe

Balayage haute fréquence (2M de points/s)

Le laser émet des impulsions

Le scanner récupère les signaux réfléchis par le sol, la végétation, les bâtiments...

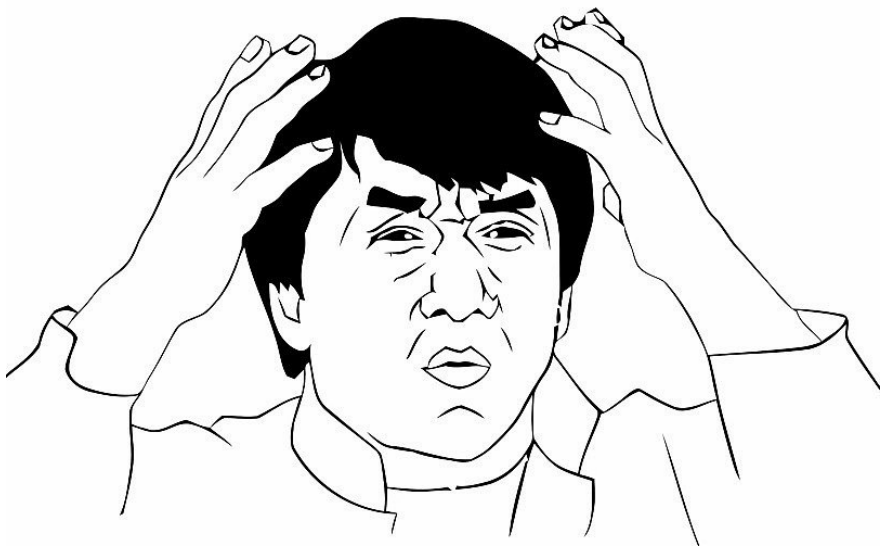
Problème : Le LIDAR c'est très volumineux

2.10^6 pts/s

1h d'acquisition $\longrightarrow$ 7.10^9 points

1 journée de 6h $\longrightarrow$ 42.10^9 points

+ 12 attributs / pt $\longrightarrow$ **2To** la journée



Un standard (de facto) pour échanger et stocker les données lidar

- LAS
- LAZ (version compressée du LAS)

las: http://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html

laz: https://www.cs.unc.edu/~isenburg/lastools/download/laszip.pdf

Problèmes du stockage fichier

beaucoup de petits fichiers

gestion arborescente compliquée

nécessite des scripts parallélisés pour traiter ces gros volumes de données

....

Utiliser un SGBD ?

- Requêtes
  - spatiales
  - temporelles
  - attributaires
- Croiser les données
- Mettre à jour
- Centraliser la gestion de la donnée

Chaque point est géoréférencé $\longrightarrow$ ~~Postgis ?~~

difficile de stocker un point par ligne avec une telle volumétrie

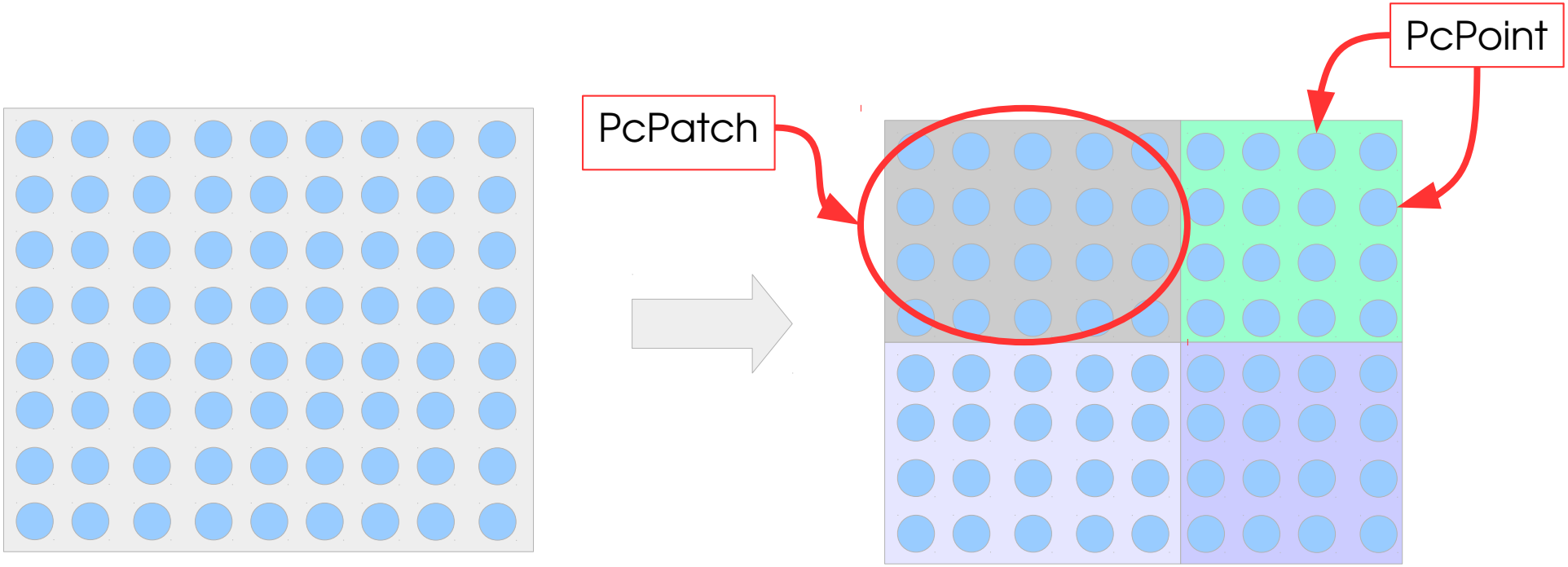
Besoin de regrouper les points pour optimiser le stockage

pgPointCloud

https://github.com/pgpointcloud/pointcloud

Extension PotgreSQL pour stocker les nuages de points.

Organise les points en Patch pour réduire la taille de la table !



L'extension **pointcloud** ajoute :

PcPoint : type pour stocker un point et toutes ses dimensions

PcPatch : type permettant de stocker un ensemble de PcPoint

pointcloud_formats : table de description des points via un schéma XML

pointcloud_columns : vue permettant de lister les tables contenant des données pointcloud (équivalente de la geometry_columns de postgis)

pc_* : ensemble de fonctions permettant de manipuler les patch/points

L'extension **pointcloud_postgis** ajoute

- le cast en géométrie PostGIS
- les fonctions d'intersections entre géométries PostGIS et PcPoint/PcPatch

Schéma XML permettant d'interpréter les données

```
INSERT INTO pointcloud_formats (pcid, srid, schema) VALUES (1, 4326,
'<?xml version="1.0" encoding="UTF-8"?>
<pc:PointCloudSchema xmlns:pc="http://pointcloud.org/schemas/PC/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <pc:dimension>
    <pc:position>1</pc:position>
    <pc:size>4</pc:size>
    <pc:description>X coordinate as a long integer. You must use the
                    scale and offset information of the header to
                    determine the double value.</pc:description>
    <pc:name>X</pc:name>
    <pc:interpretation>int32_t</pc:interpretation>
    <pc:scale>0.01</pc:scale>
  </pc:dimension>
  <pc:dimension>
    <pc:position>2</pc:position>
    <pc:size>4</pc:size>
    <pc:description>Y coordinate as a long integer. You must use the
                    scale and offset information of the header to
                    determine the double value.</pc:description>
    <pc:name>Y</pc:name>
    <pc:interpretation>int32_t</pc:interpretation>
    <pc:scale>0.01</pc:scale>
  </pc:dimension>
  <pc:dimension>
    <pc:position>3</pc:position>
    <pc:size>4</pc:size>
    <pc:description>Z coordinate as a long integer. You must use the
                    scale and offset information of the header to
                    determine the double value.</pc:description>
    <pc:name>Z</pc:name>
    <pc:interpretation>int32_t</pc:interpretation>
    <pc:scale>0.01</pc:scale>
  </pc:dimension>
```

Compression des patchs :

(Définie dans le schema XML)

dimensional : réordonne les données des patchs par dimensions et applique une compression en fonction des statistiques (RLE, ZLIB, SIGBITS) (x4)

GHT: GeoHashTree (x2)

LAZ (contribution Oslandia) : compression x10 $\longrightarrow$ x20

Compression à la volée avec la fonction **PC_Compress()**

création de patch :

```
CREATE TABLE mylidar AS
SELECT
    PC_Patch(PC_MakePoint(3, ARRAY[x,y,z,intensity])) as pa
FROM (
 SELECT
 -127+a/100.0 AS x,
   45+a/100.0 AS y,
        1.0*a AS z,
         a/10 AS intensity,
         a/400 AS gid
  FROM generate_series(1,100000) AS a
) AS values GROUP BY gid;
```

nombre de patchs :

```
# SELECT count(*) from mylidar;
  sum
--------
 251
(1 ligne)
```

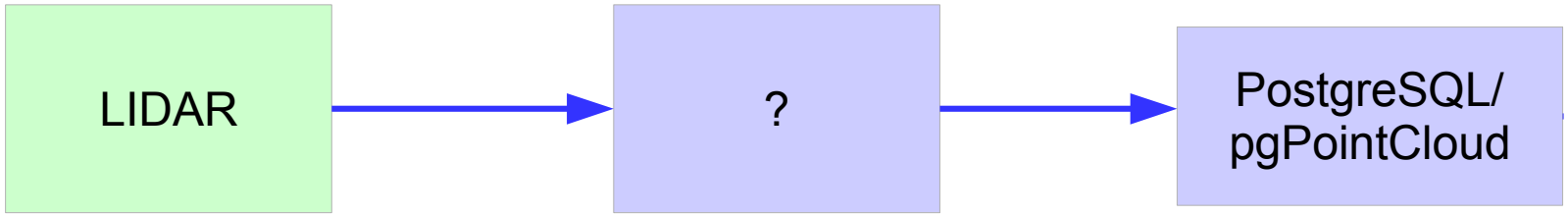
nombre de points :

```
# SELECT sum(pc_numpoints(pa)) from mylidar;
  sum
--------
 100000
(1 ligne)
```

indexation sur l'enveloppe d'un patch:

```
create index on mylidar using gist(geometry(pa))
```

PointCloud Stack



PDAL (Point Data Abstraction Library)

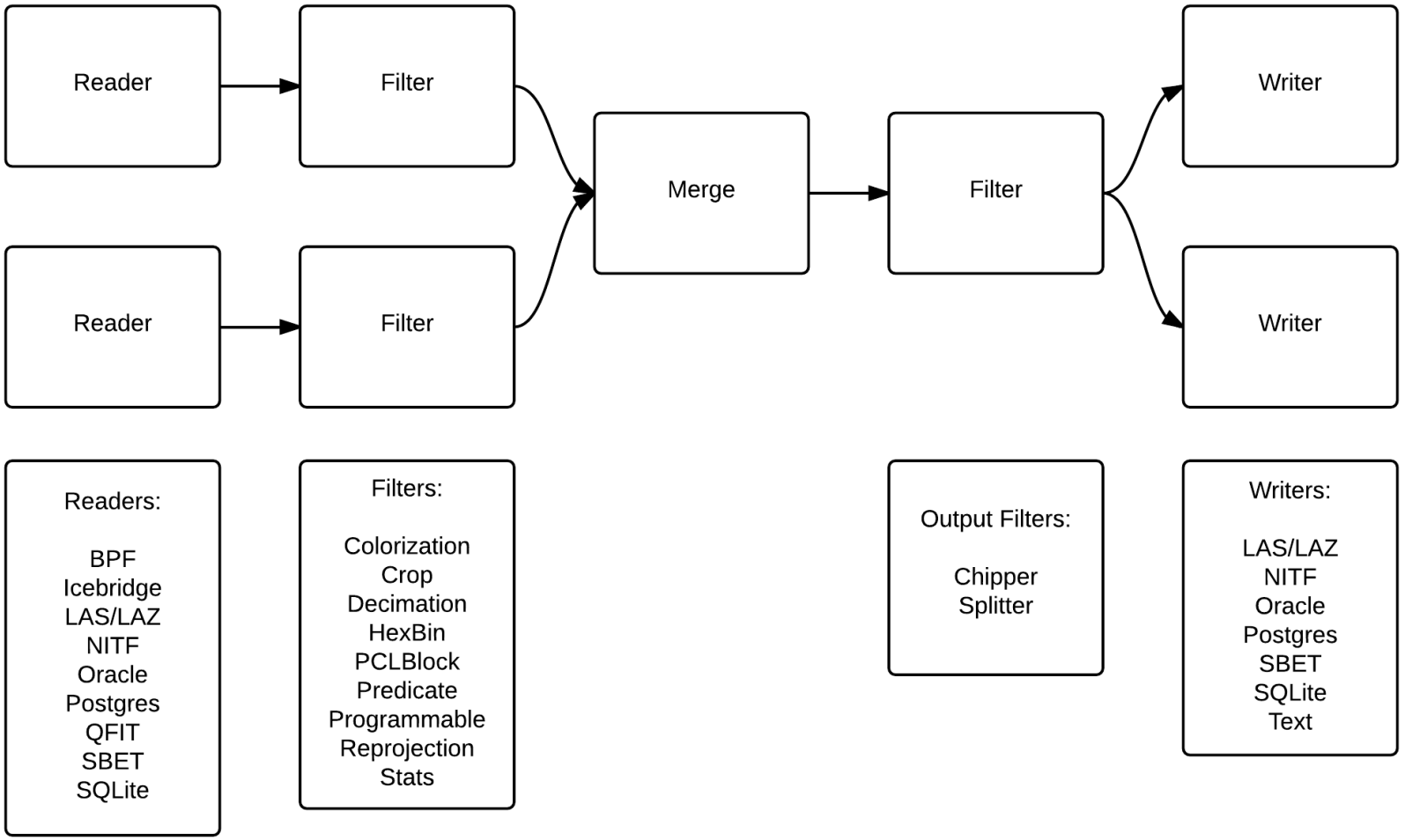
outil en ligne de commande

filtrage / lecture / écriture dans différents formats

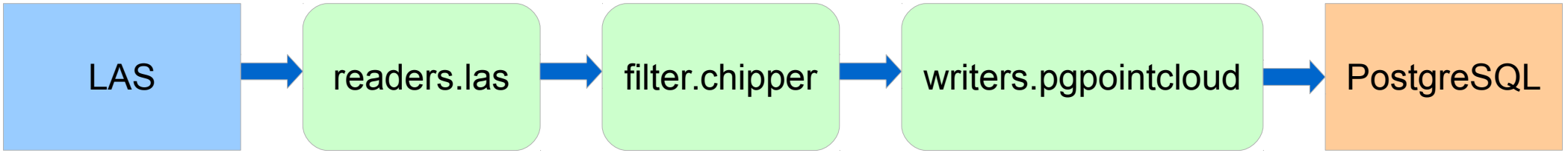
Concept de pipeline pour chainer les opérations sur les données

autrefois XML pour la description → JSON depuis la v1.2

PDAL


Reader
Filter
Reader
Filter
Merge
Filter
Writer
Writer
Readers:
BPF
Icebridge
LAS/LAZ
NITF
Oracle
Postgres
QFIT
SBET
SQLite
Filters:
Colorization
Crop
Decimation
HexBin
PCLBlock
Predicate
Programmable
Reprojection
Stats
Output Filters:
Chipper
Splitter
Writers:
LAS/LAZ
NITF
Oracle
Postgres
SBET
SQLite
Text

PDAL




```
{
  "pipeline":[
    {
      "type":"readers.las",
      "filename":"lidar.las"
    },
    {
      "type":"filters.chipper",
      "capacity":400
    },
    {
      "type":"writers.pgpointcloud",
      "connection":"host=pg.pdal.net dbname=lidar user=lidar",
      "table":"xxx_patches"
    }
  ]
}
```

```
pdal pipeline pipe_pg.json
```

Limitations

Performances perfectibles (le driver d'écriture dans PG n'utilise pas le mode copy)

Parallélisation non native ⟶ GNU/parallel

Contributions en cours

support du **LAZ** dans le driver **writers.pgpointcloud**

(suite au support récent du LAZ dans pgpointcloud)

https://github.com/LI3DS/pdal/

PointCloud Stack

Foreign Data Wrapper en python (basé sur Multicorn)

- Accéder dynamiquement aux données sources
- Gérer des formats non supportés par PDAL
- 2 formats gérés (sbet et EPT propre à l'IGN)
- Création de patchs à la volée en utilisant le temps (cohérence temps/espace)
- Facilement extensible !

https://github.com/LI3DS/fdw-pointcloud

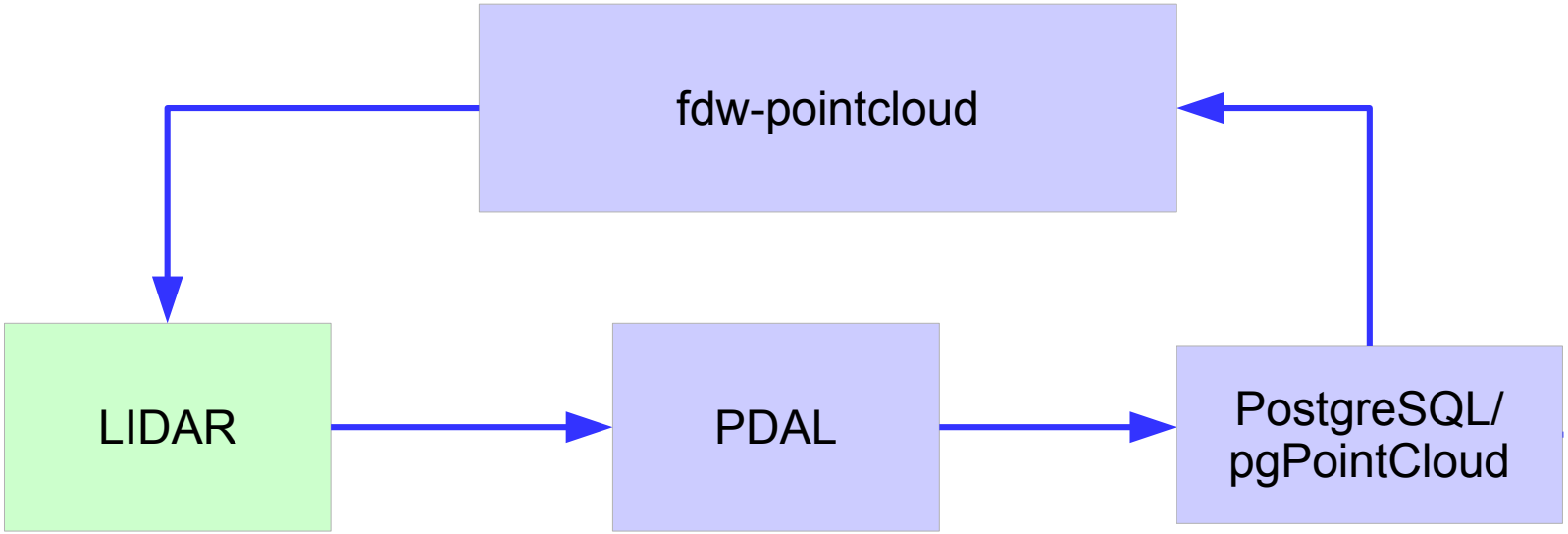
fdw-pointcloud

```
create server sbetserver foreign data wrapper multicorn
    options (
        wrapper 'fdwpointcloud.Sbet'
    );

create foreign table mysbet_schema (
    schema text
)
server route_server
 options (
    metadata 'true'
);

insert into pointcloud_formats (pcid, srid, schema)
select 2, 4326, schema from mysbet_schema;

create foreign table mysbet (
    points pcpatch(2)
) server sbetserver
    options (
        sources 'data/sbet.bin'
        , patch_size '100'
        , pcid '2'
);
```

TODO :

- intégrer un filtre chipper pour regrouper les points spatialement
- ajout du filtrage temporel à la source
- lecture las/laz

PointCloud Stack



Statistiques d'un patch :

```
# select pc_summary(pa) from mylidar

{
  "pcid": 2,
  "npts": 400,
  "srid": -15,
  "compr": "laz",
  "dims": [
    {
      "pos": 0,
      "name": "x",
      "size": 8,
      "type": "double",
      "stats": {
        "min": 1.38141E+9,
        "max": 1.38141E+9,
        "avg": 1.38141E+9
      }
    },
...
```

Lister le contenu d'un patch:

```
# select pc_astext(pc_explode(pa)) from mylidar limit 10

                    pc_astext
-----------------------------------------------
 {"pcid":3,"pt":[389,561,51600,5160]}
 {"pcid":3,"pt":[389.01,561.01,51601,5160]}
 {"pcid":3,"pt":[389.02,561.02,51602,5160]}
 {"pcid":3,"pt":[389.03,561.03,51603,5160]}
 {"pcid":3,"pt":[389.04,561.04,51604,5160]}
 {"pcid":3,"pt":[389.05,561.05,51605,5160]}
 {"pcid":3,"pt":[389.06,561.06,51606,5160]}
 {"pcid":3,"pt":[389.07,561.07,51607,5160]}
 {"pcid":3,"pt":[389.08,561.08,51608,5160]}
 {"pcid":3,"pt":[389.09,561.09,51609,5160]}
```

Récupérer l'altitude moyenne d'un patch

```
with tmp as (
    select
        json_array_elements(pc_summary(pa)::json->'dims') as
dims
    from mylidar where id = 1
)
select
    dims->'stats'->'avg'
from tmp
where dims->>'name' = 'Z';
```

Extraire les points assimilés à de l'eau grâce à la classification contenue dans le format LAS

Table 4.9 - ASPRS Standard LIDAR Point Classes

Classification Value	*Meaning*
0	Created, never classified
1	Unclassified[1]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point ("low noise")
8	High Point (typically "high noise"). Note that this value was previously used for Model Key Points. Bit 1 of the Classification Flag must now be used to indicate Model Key Points. This allows the model key point class to be preserved.
9	Water
10	Rail
11	Road Surface
12	Bridge Deck
13	Wire - Guard
14	Wire – Conductor (Phase)
15	Transmission Tower
16	WIre-structure Connector (e.g. Insulator)
17	Reserved
18-63	Reserved
64-255	User definable – The specific use of these classes should be encoded in the Classification lookup VLR.

Extraire les points assimilés à de l'eau grâce à la classification contenue dans le format LAS

```
select
    pt as water_point
from
    mylidar,
    pc_explode(pa) as pt
where
    pc_get(pt, 'Classification') = 9 -- water
```

Trouver le point culminant d'une zone montagneuse

```
with points_in_area as (
    select
        pc_explode(pa) as pt
    from
        lidar
where
    -- gist index usage
    pc_intersects(pa, st_geomfromtext('polygon((696645....))'))
)
select
    max(pc_get(pt, 'Z')) as alt
from
    points_in_area
where
    pc_get(pt, 'Classification') = 2 -- ground
```

https://github.com/Oslandia/workshop-pointcloud

Contributions pgPointCloud

Support de la compression LAZ → OK

PC_PointN → OK

PC_range → WIP

PC_Interpolate → WIP

PC_PatchTransform → WIP

https://github.com/LI3DS/pointcloud/tree/dev

Perspectives PostGIS / PostgreSQL

PostGIS 2.3 avec l'apport des index BRIN

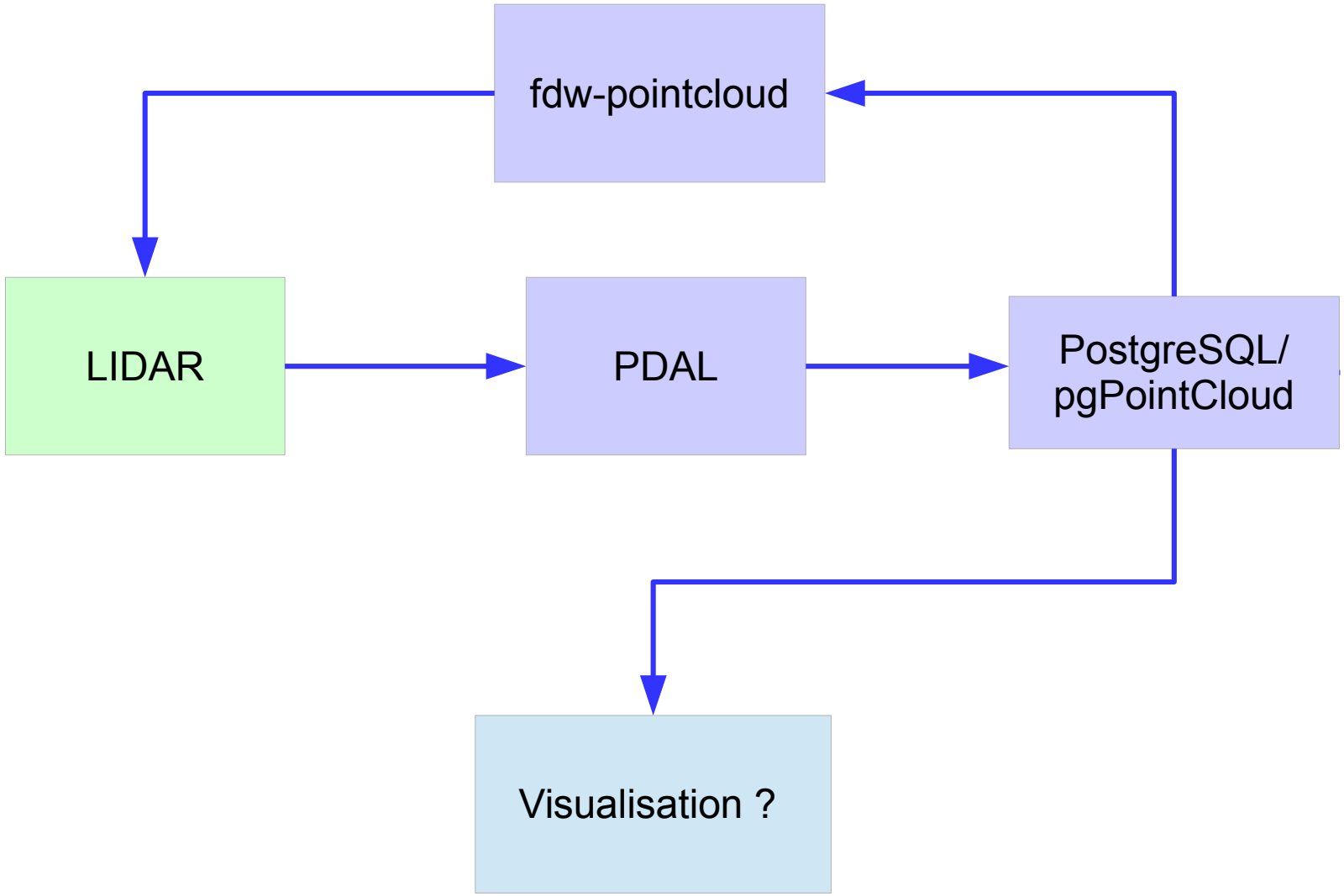
PostgreSQL 9.6 avec la parallélisation

Stockage efficace pour la visualisation 3D?

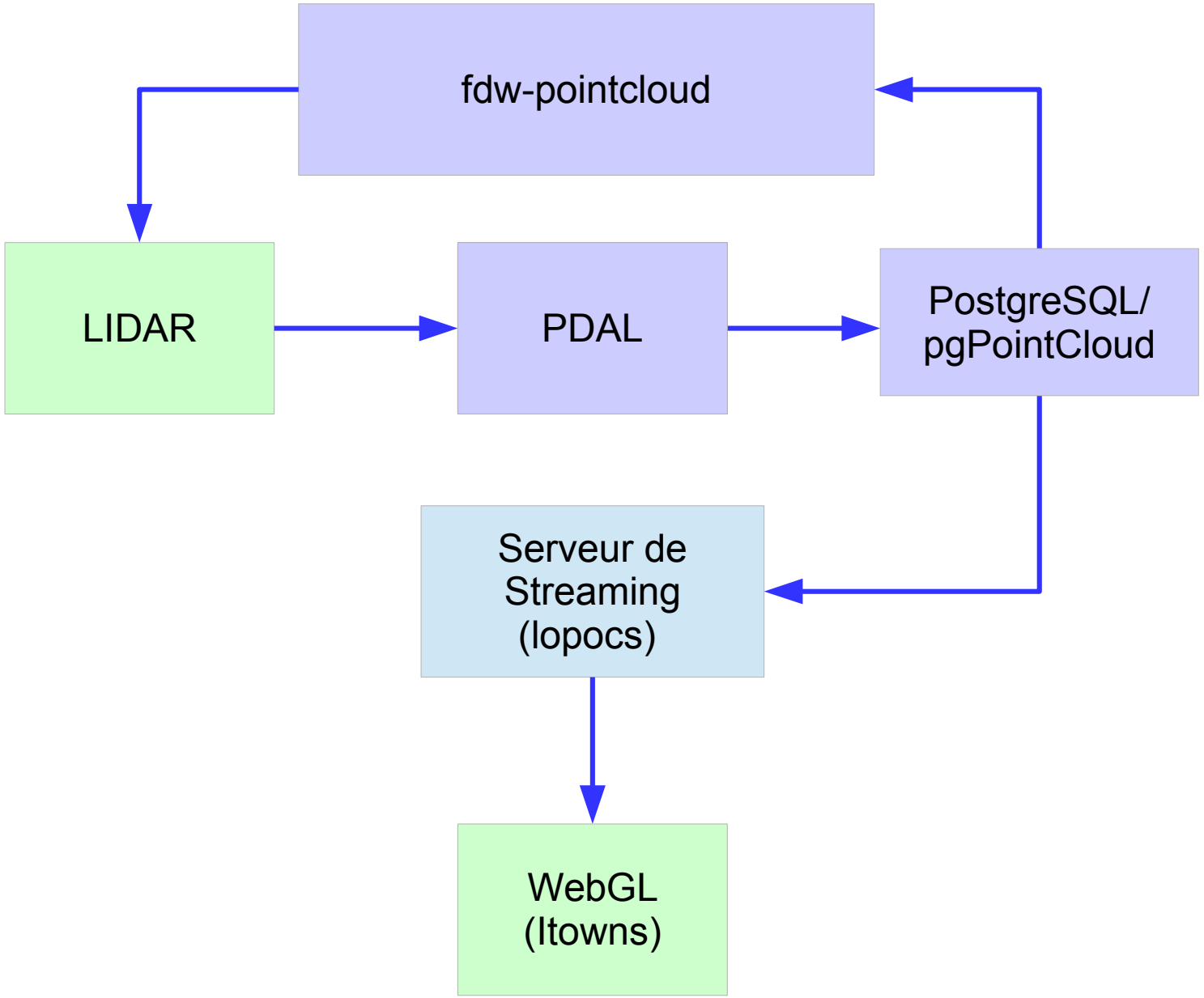
- intégrer un LOD (niveau de détail) directement au niveau des patch (les points d'un patch peuvent être ordonnés par LOD), il suffit alors de récupérer les points avec **pc_range**

- Code de morton pour une indexation de type octree

PointCloud Stack

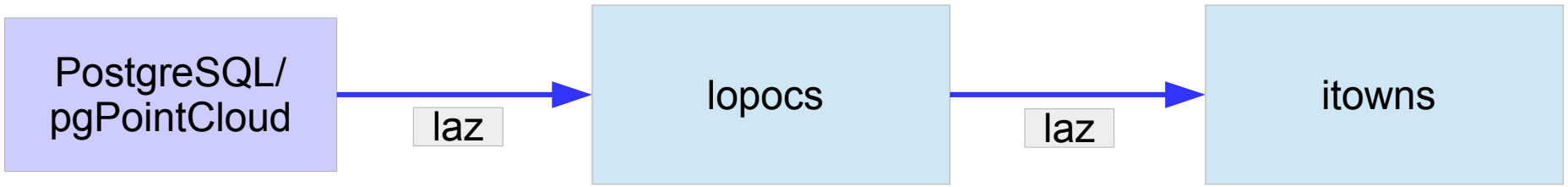


PointCloud Stack



LOPOCS : Light OpenSource PointCloud Server

- utilise le protocole de Greyhound (mais 3Dtiles en cours d'étude)
- Prend une bbox en entrée et renvoie des données laz
- Échanges binaires



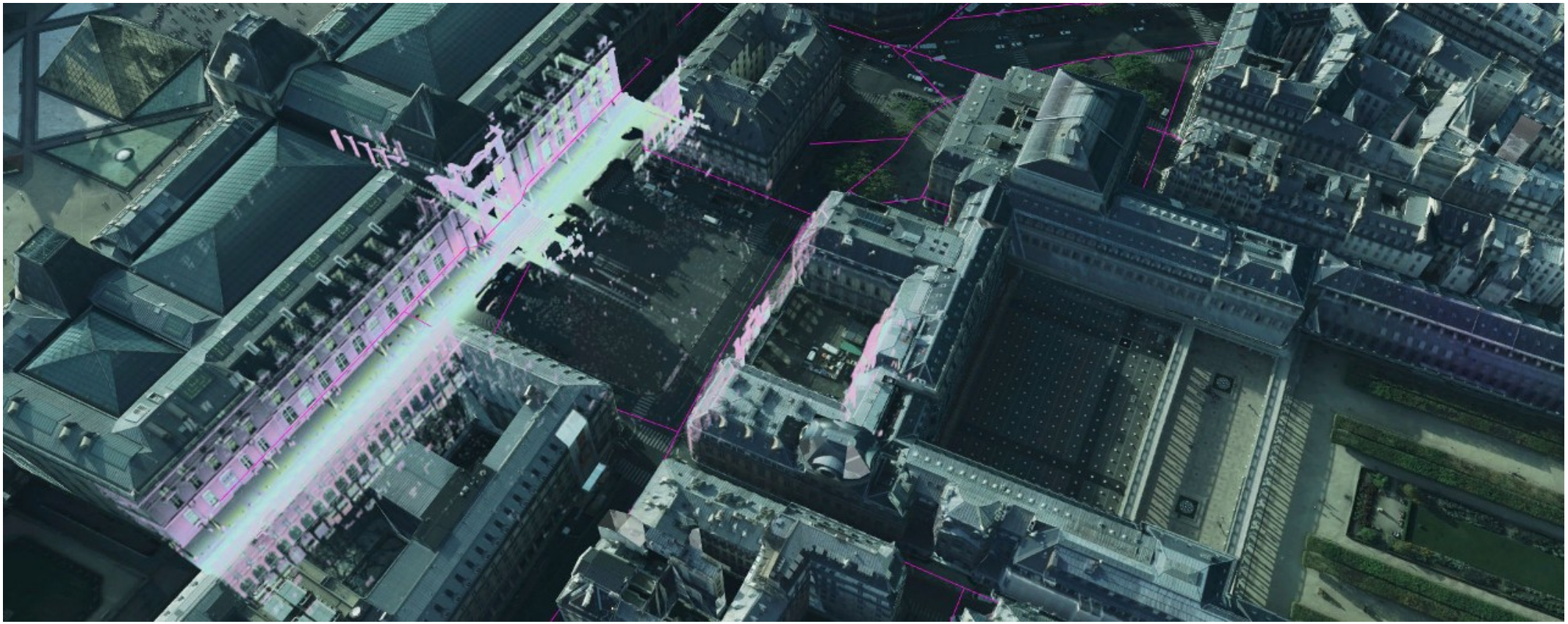
https://github.com/LI3DS/lopocs

Viewer Web(GL) pour données 3D hétérogènes

IGN
INSTITUT NATIONAL
DE L'INFORMATION
GÉOGRAPHIQUE
ET FORESTIÈRE

OSLANDIA

www.itowns-project.org



Démo lidar aérien

Démo immersif

Questions ?