GIO 2 - GÉOINFORMATIQUE OPÉRATIONNELLE

GÉOCOMPUTATION

Détail sur le projet

ADRIEN GRESSIN

2022 - 2023

INTRODUCTION

PLANNING PRÉVISIONNEL

- 10/05 : Organisation du projet + calcul de base sur 1 machine
- 17/05 : Séance Libre / Répartition des calculs
- 24/05 : Séance Libre / Visualisation des résultats
- 31/05 : Séance Libre

ORGANISATION DU PROJET

ORGANISATION DU PROJET

Objectif : Calculer une carte de type "heatmap" de trace GPS (VTT ou autre) sur un cluster de RPI

Heatmap from Strava

ORGANISATION DU PROJET

Produit attendu : image couleur (colormap) / multi échelle (Timemap XYZ)

Rendus attendus :

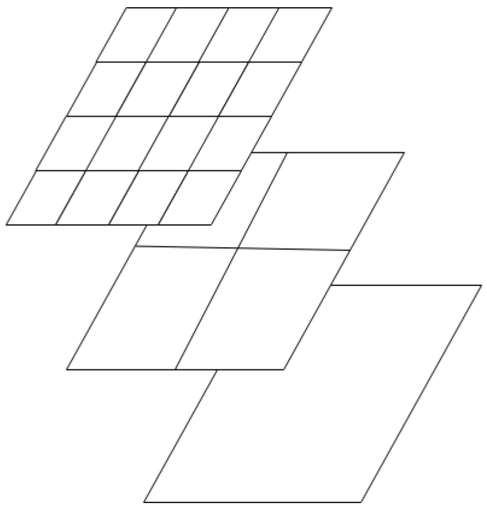
- Méthode de calcul distribué permettant le calcul des "heatmap"
- Suivi des calculs (# de RPI connecté, charge des RPI, pourcentage d'avancement...)
- Visualisation des résultats (page web)
- Présentation
  - Démonstration par groupe (présentation globale de l'infrastructure déployée, ...)
  - Présentation individuelle technique et détaillée sur la partie de votre choix

INFRASTRUCTURE DE CALCUL

- Node : Calcul d'une tuile
  - Récupération des traces
  - Rendu raster
- Scheduler / Client
  - Liste des RPIs connecté (Nodes)
  - 1ere étape : génération des tuiles de base ($Z = Z_{max}$)
    - Distribution des tuiles aux nodes
    - Calcul sur chaque node
    - Récupération des tuiles
  - 2e étape : fusion des tuiles
    - Par niveau de zoom $Z \rightarrow Z - 1$
    - Envoi des tuiles par 4 aux nodes
    - Chaque node calcul la fusion de ces tuiles
    - Récupération des tuiles fusionnées

CALCUL

GÉNÉRATION DE LA TILEMAP



- Calcul des tuiles de bases
- Fusion pour avoir les autres niveaux de zoom

CONVERSION XYZ / LAT,LON

```python
import math
def num2deg(x_tile, y_tile, zoom):
  """
  return the NW-corner of the tile, Use x_tile+1, y_tile+1 to get the other corners
  """
  n = 2.0 ** zoom
  lon_deg = x_tile / n * 360.0 - 180.0
  lat_rad = math.atan(math.sinh(math.pi * (1 - 2 * y_tile / n)))
  lat_deg = math.degrees(lat_rad)
  return lat_deg, lon_deg
```

```python
import math
def deg2num(lat_deg, lon_deg, zoom):
  """
  return the tile containing the point (lat_deg, lon_deg)
  """
  lat_rad = math.radians(lat_deg)
  n = 2.0 ** zoom
  x_tile = int((lon_deg + 180.0) / 360.0 * n)
  y_tile = int((1.0 - math.asinh(math.tan(lat_rad)) / math.pi) / 2.0 * n)
  return x_tile, y_tile
```

https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames#Python

CALCUL D'UNE TUILE DE BASE

- Recevoir les informations d'une tuile XYZ
- Récupérer les traces disponibles sur une zone
- Générer la tuile avec gdal_rasterize

```
gdal_rasterize
    [-burn value] [-add]
    [-of format] [-a_srs srs_def]
    [-a_nodata value] [-init value]
    [-te xmin ymin xmax ymax] [-tr xres yres] [-ts width height]
    [-ot {Byte/Int8/Int16/UInt16/UInt32/Int32/UInt64/Int64/Float32/Float64/
            CInt16/CInt32/CFloat32/CFloat64}]
    <src_datasource> <dst_filename>
```

https://gdal.org/programs/gdal_rasterize.html

FUSION DES TILES

- Concaténation
- Sous échantillonnage par somme / max des pixels

```
import numpy as np
import skimage.measure

a = np.array([
      [  20,  200,   -5,   23],
      [ -13,  134,  119,  100],
      [ 120,   32,   49,   25],
      [-120,   12,    9,   23]
])
fus_max = skimage.measure.block_reduce(a, (2,2), np.max)
fus_sum = skimage.measure.block_reduce(a, (2,2), np.sum)
```