ODATIS

Contribution Pangeo & Intake

CLS GROUP

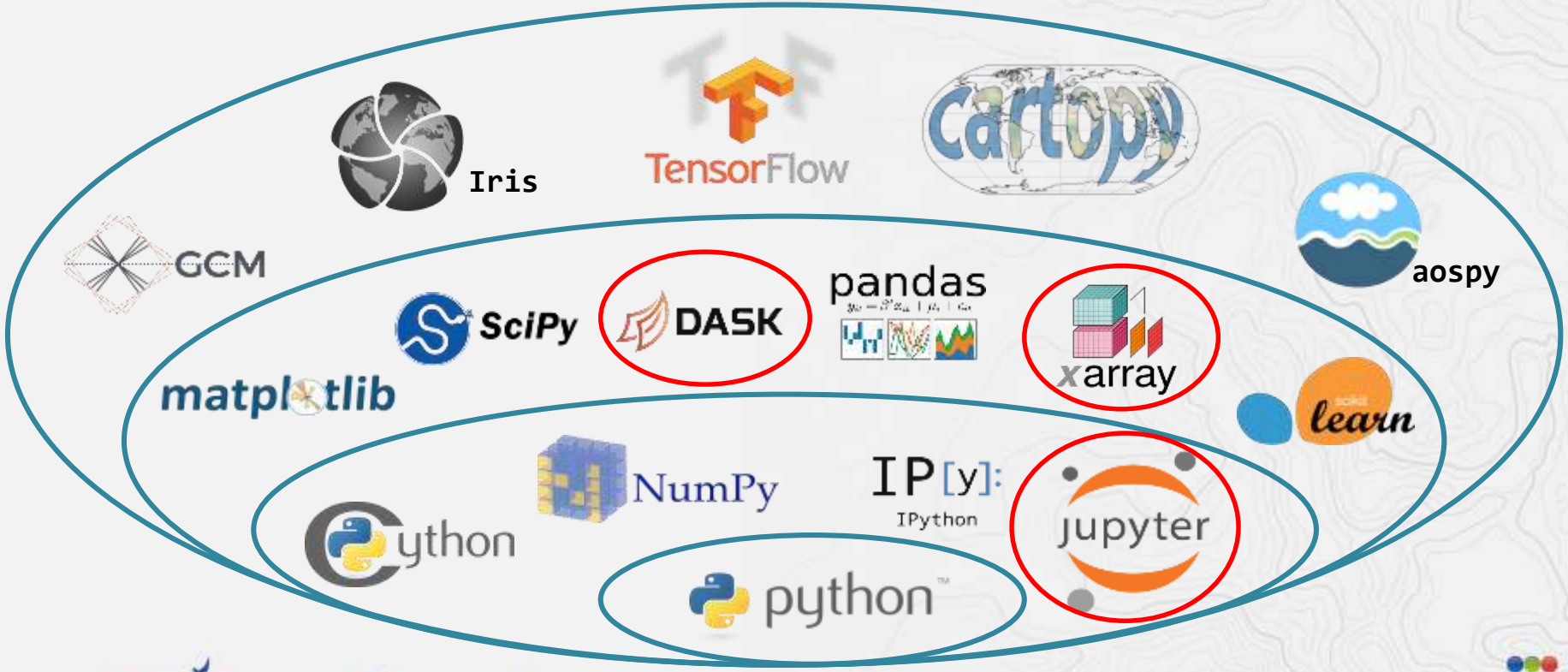
Pangeo

Pangeo est une communauté qui travaille au développement de logiciels et d'infrastructures pour faciliter la mise en œuvre des géosciences, dans le domaine du « Big Data ».

- **Mission** : entretenir un écosystème dans lequel la prochaine génération d'outils, d'analyse « open source » pour les géosciences appliquée à de gros volumes de données peut être développée, distribuée et maintenue.

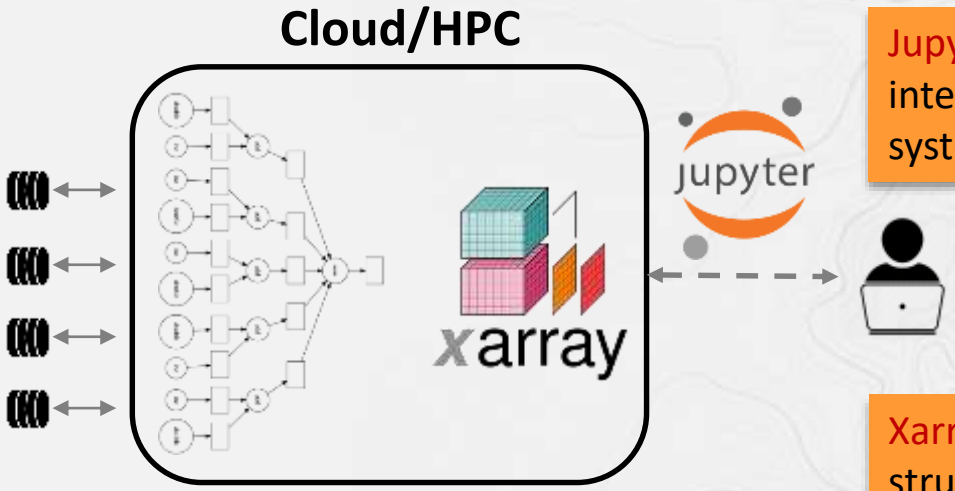
- **Vision**:
  - Développement ouvert et collaboratif.
  - Outils de mise à l'échelle des calculs pour des ensembles de données de petite à très grande taille.
  - Structures pour intégrer l'analyse scientifique aux données.
  - Encourager une culture du développement accueillante et participative.

Pile logicielle Pangeo



Architecture Pangeo

Données prêtes à l'analyse, stockées et cataloguées sur un système de stockage distribué accessible à l'échelle globale (p. ex. S3, GCS)



Jupyter pour un accès interactif sur des systèmes distants.

Système de calcul parallèle construit sur Kubernetes ou HPC. Dask dit aux nœuds ce qu'ils doivent faire.

Xarray fournit des structures de données et une interface intuitive pour interagir avec les ensembles de données.

Indexer les données géographiques

Il existe de nombreuses bibliothèques d'interpolation de grille, mais en général elles sont entièrement écrites en Python et ne sont pas très efficaces : MetPy, Verde, pyresample, etc.

Les outils de base sont d'engourdissement, scipy : RegularGridInterpolator, cKDTree.

Mais aucun de ces outils ne gère la discontinuité des longitudes : la transition autour du méridien de Greenwich.

cKDTree

Combiné avec pyproj, les données géographiques peuvent être interpolées : Conversion LLA (longitude, latitude, altitude) en coordonnées cartésiennes (ECEF).

Cette arborescence de recherche fonctionne correctement, mais dès que le problème devient important, les performances s'effondrent.

Pour 74 649 600 points (MIT/GCM/LLC4320):

- 30 minutes pour la construction de l'arbre.
- La requête des plus proches voisins de 4147200 pixels n'a pas fonctionné dans un temps raisonnable: j'ai tué le processus après 4 heures de calculs.

Boost Geometry - RTree

Données Cartésienne

- L'enveloppe Python gère jusqu'à 5 dimensions cartésienne pour le moment.
- La librairie Boost n'implémente pas de typage dynamique du nombre de dimensions (Boost met l'accent sur la performance et non sur la flexibilité): le nombre de dimensions est géré par une meta classe C++. La classe C++ implémentée est un modèle et il est facile de l'étendre à plusieurs dimensions.

Données géodésiques

- La conversion des coordonnées LLA en coordonnées cartésiennes ECEF est gérée par la classe. Pour ce faire, un système géodésique doit être défini. Par défaut, nous considérons WGS-84.
- Nous utilisons un algorithme plus rapide que ce que nous pouvons trouver dans proj4 : Olson, D.K. "Converting earth-Centered, Earth-Fixed Coordinates to Geodetic Coordinates", IEEE Transactions on Aerospace and Electronic Systems, vol. 32, no 1, janvier 1996, p. 473-476.

La performance de cet algorithme est impressionnante.

Pour le même problème précédent (LLC4320) : 11 secondes pour construire l'arbre de recherche, 4,5 secondes pour effectuer les requêtes « du plus proche voisin ».

Axe

Cet objet gère les axes des grilles régulières et irrégulières.

Les données doivent toujours être monotones, c'est un axe d'une grille.

Le constructeur choisira, comme les indices de Pandas, le meilleur indexeur pour traiter les valeurs fournies.

Si les données sont régulières, le calcul suivant est réalisé: "round((coordinate - start_) / step_", sinon une recherche binaire sera effectuée.

L'axe gère les angles : il faut spécifier si l'axe peut représenter un cercle (comme les longitudes) et il gère de manière transparente les problèmes de normalisation des angles ([-180, 180] ou[0, 360]) et la discontinuité du cercle.

Axe

```
import numpy as np
# install with conda: conda install pyindex -c fbriol
import pyindex.core as core
# 1D-Spatial index
# We have longitudes that can define a circle
axis = core.Axis(
       np.arange(-180, 179,1), is_circle=True,
       is_radian=False)
# But this axis is not a circle.
print(axis.is_circle)
```

Axe

```
axis = core.Axis(np.arange(-180, 180,1), is_circle=True)
# Here we have a circle.
print(axis.is_circle)
# In this case, requests can be made without regard to
angular values if they are expressed in degrees.
axis.find_index([360, 359.5, -270])
```

RTree – Données cartésiennes

```
tree = core.cartesian.RTree2D()
# Wikipedia example
coordinates = np.array(
    [[2, 3], [5, 4], [9, 6], [4, 7], [8, 1], [7, 2]])
# The preferred method to use to optimize queries and data
# insertion.
tree.packing(coordinates)
dist, index = tree.query(
    np.array([[2.1, 1.1]]), k=1, num_threads=1)
# We check that the requested point is enclosed by its
# neighbours (important to avoid extrapolation)
dist, index = tree.query(
np.array([[2.1, 1.1]]), k=1, within=True, num_threads=1)
```

CLS GROUP

RTree – Données Géodétiques

```
system = core.geodetic.System()
tree = core.geodetic.RTree()
# or core.geodetic.RTree(system=None)
# or core.geodetic.RTree(system=system)
lon = np.random.uniform(-180.0, 180.0, 2048*4096)
lat = np.random.uniform(-90.0, 90.0, 2048*4096)
alt = np.random.uniform(-10000, 100000, 2048*4096)
tree.packing(np.asarray((lon, lat, alt)).T)
```

CLS GROUP

RTree – Données géodésiques

```
coordinates = np.asarray((
    np.random.uniform(-180.0, 180.0, 10000),
    np.random.uniform(-90.0, 90.0, 10000),
    np.random.uniform(-10000, 100000, 10000))).T

# On my latop
%timeit tree.query(
    coordinates, k=4, num_threads=0, within=False)
# 9.23 ms ± 120 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

CLS GROUP

Intake

IT

Vous ne devriez JAMAIS inclure d'informations d'identification dans vos notebook.

Attendez - ce n'est même pas votre nom d'utilisateur/mot de passe !

e fois ?

']

```
base_url = "http://internal.server.com"
headers = {'password': 'S3cret_StuFF#', 'username':
'madame_user'}
data = []
```

Fournisseur de données

Vous ne devriez pas utiliser
eu une mise à jour. La nouv
stockée sur Azure, vous dev

Dev

Pandas n'a pas de lecteur pour avro. Je dois en faire un, et sur Azure en plus ?

Intake

```
import intake
cat = intake.open_catalog("afile.yaml")
df = cat.useful_data_read()
```

Intake

Utilisateur de données

Quels sont les ensembles de données disponibles ?

Quel ensemble de données est le plus

Intake

Fournisseur de données

M
ut

Co

Di

```yaml
metadata:
version: 1
sources:
example:
description: test
driver: random
args: {}

entry1_full:
description: entry1 full
metadata:
foo: 'bar'
bar: [1, 2, 3]
driver: csv
args: # passed to the open() method
urlpath: '{{ CATALOG_DIR }}/entry1_*.csv'
```

CLS GROUP

Intake

IT	Contrôle d'accès :	Attribution
		Surveillance
	Système d'authentification	
	Surveiller les données entrantes	

Intake

Développeur	Ecrire du code pour charger les données
	Ne pas réécrire et réinventer le code
	Maintenir une expérience utilisateur cohérente
	Délais d'exécution rapides