Format de stockage

Utilisation vs Distribution

Introduction

- Stockage pour la diffusion vs utilisation
- Type de données
- Formats compatible avec le calcul distribué (fragmentation des données.)
- Pile logicielle Pangeo
- Cas d'utilisation

Big Data (plus de 100 Go)

Le volume des ensembles de données scientifiques croît à un rythme exponentiel et les scientifiques ont du mal à suivre le rythme.

Quelques exemples de grands ensembles de données dans les domaines de l'océanographie et de la climatologie :

- Projet de comparaison des modèles couplés du Programme mondial de recherche sur le climat (3.3 PB)
- La réanalyse atmosphérique ERA5 du Centre européen de prévision à moyen terme (~5PB)
- Température de la surface de la mer à très haute résolution multi échelle de la NASA (12 To)

Problématique

Pour traiter ces volumes de données, il faut pouvoir les traiter dans un mode parallèle (multi processus) ou dans un mode de calcul distribué y compris dans le "cloud computing".

L'extraction et le stockage des données constituent souvent le principal goulot d'étranglement, et de nouvelles approches du stockage des données sont nécessaires pour accélérer les calculs distribués et leur permettre d'évoluer sur une variété de plates-formes.

Les données multidimensionnelles, telles que celles qui sont généralement stockées dans les formats HDF et NetCDF, sont difficiles d'accès sur les plates-formes de calcul distribués et sur le stockage traditionnel des clouds (S3, GFS, etc.)

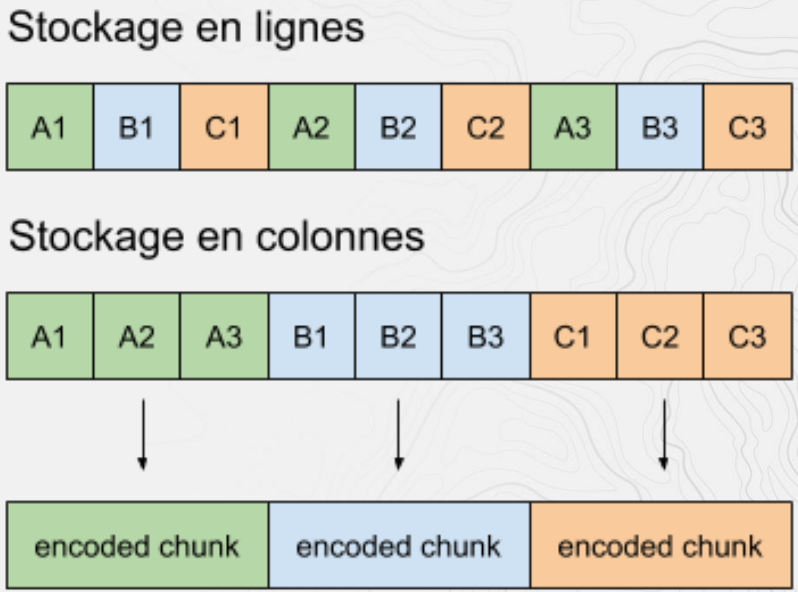
Type de données

Comment peut-on modéliser simplement les données scientifiques ? Au risque de simplifier à l'excès, la plupart des données géoscientifiques se répartissent en deux catégories:

- Données tabulaires : lignes et colonnes ; tout ce que vous pouvez mettre dans une feuille de calcul Excel, un fichier CSV ou une base de données. *Par exemple, un tableau des hauteurs des océans mesurés par des marégraphes*.
- Tableaux multidimensionnels : tout ce que vous pouvez mettre dans un fichier netCDF ou HDF. *Par exemple, les observations satellitaires de la température de l'océan.*

Données tabulaires : stockage colonne

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3



Accès aux données nécessaires

En colonnes

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Statistiques

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Fragment

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Stockage colonne

Limite les E/S aux données réellement nécessaires :

- Charger uniquement les colonnes auxquelles il faut accéder.

Gain de place :

- La disposition en colonnes compresse mieux
- Encodages spécifiques.

Active les moteurs d'exécution vectorisée.

Parquet : le format colonne sur disque

Le format Apache Parquet est un standard pour stocker des données tabulaires qui peuvent être lues et écrites à partir des langages de programmation les plus courants.

Ce stockage en colonnes offre les avantages suivants :

- La compression par colonne est efficace et permet d'économiser de l'espace de stockage.
- Des techniques de compression spécifiques à un type peuvent être appliquées, car les valeurs des colonnes ont tendance à être du même type.
- Les requêtes qui récupèrent des valeurs de colonnes spécifiques n'ont pas besoin de lire la totalité des données brutes, ce qui améliore les performances.
- Différentes techniques d'encodage peuvent être appliquées à différentes colonnes.

Arrow: le format colonne en mémoire

- Bien documenté et compatible avec tous les langages
- Conception pour tirer profit des caractéristiques modernes des CPU
- Intégrable dans les moteurs d'exécution, les couches de stockage, etc.
- Interopérable
- Structure de données imbriquée
- Maximiser le débit du CPU : pipelines, SIMD, localité de cache, etc.
- E/S dispersion/rassemblement

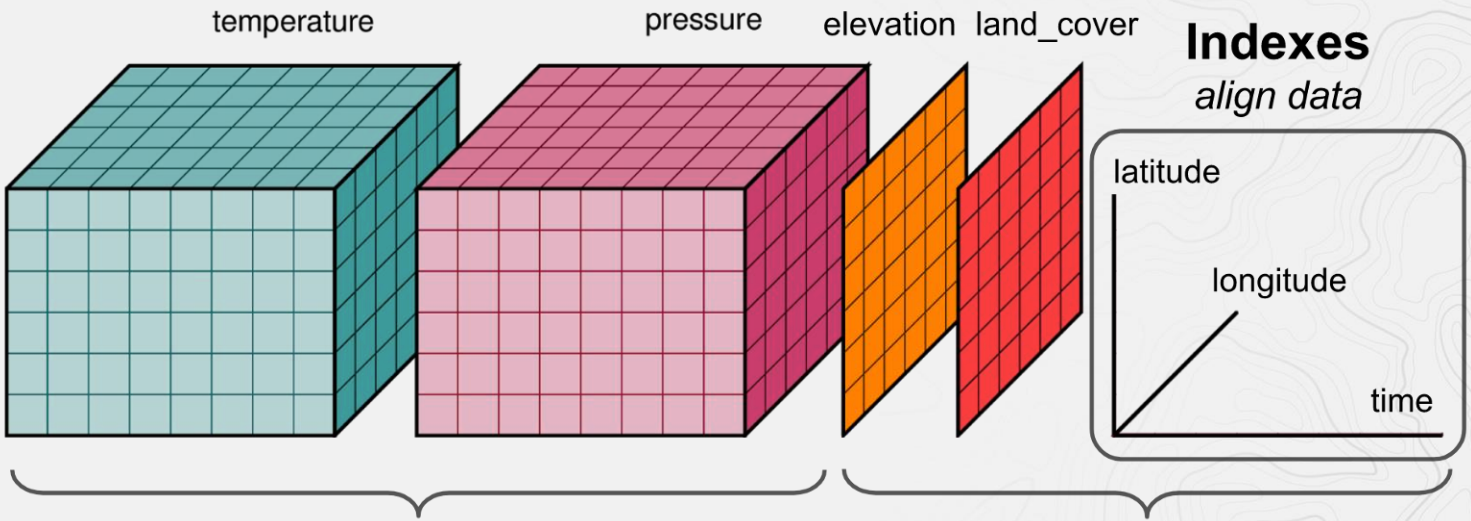
Tableaux multidimensionnels

Les tableaux multidimensionnels sont très courants dans la recherche scientifique ; il y a un certain chevauchement avec les données couvertes par GeoTIFF, mais les tableaux multidimensionnels sont beaucoup plus génériques.

Les données de tableaux multidimensionnels sont conceptuellement simples parce qu'elles ont tendance à être homogènes (toutes les valeurs sont du même type de données).

Cependant, il peut y avoir des relations complexes entre différents tableaux ; par exemple, un tableau (longitude) peut être les coordonnées d'un autre tableau (température).

Tableaux multidimensionnels



+ **Attributes**
metadata ignored by operations

Tableaux multidimensionnels

Un problème fondamental aujourd'hui est qu'il n'existe pas de norme largement acceptée pour les données netCDF optimisées pour le calcul distribué.

Les formats traditionnels étaient destinés à être lus à partir de fichiers locaux et ne fonctionnent pas efficacement dans le stockage d'objets optimisé pour les accès concurrentiels.

Quelques solutions

Cloud Optimized GeoTIFF

- + rapide, bien établi
- - Pas assez sophistiqué pour traiter certains domaines scientifiques

HDF + FUSE : utiliser HDF par l'intermédiaire d'un système de fichiers virtuels FUSE

- + : Fonctionne avec les fichiers existants, aucune modification de la bibliothèque HDF n'est nécessaire.
- - : C'est complexe, probablement pas optimal en matière de performance

HDF + Lecteur personnalisé

- + : Fonctionne avec les fichiers existants, pas d'astuces FUSE complexes
- - : Nécessite des plug-ins pour la bibliothèque HDF et des réglages pour les wrappers Python.

Service distribué : permettre de servir ces données derrière une API Web

- + : dissocier le problème : développer une solution en arrière-plan complexe et un API frontale.
- - : complexe à écrire et à déployer. Probablement pas libre. Introduis un intermédiaire entre l'utilisateur et les données.

Concevoir un nouveau format, optimisé pour le calcul distribué

- + : rapide, intuitif et moderne
- - : N'est pas une norme communautaire

Revues des outils existants « Big Data »

RASDAMAN, THREDDS, SciDB, etc : versions commerciales chères, réponds partiellement à notre besoin.

SPARK/YARN/HADOOP : Pas suffisamment de souplesse pour faire face à la diversité et à la complexité des cas réels d'utilisation des géosciences.

Zarr est une alternative intéressantes à NetCDF4 pour le stockage interne. Solution pure avec un stockage transparent clé-valeur. Concept d'avenir intéressant pour le stockage de données dans de grandes bases de données distribuées.

N5/Z5 solutions similaires à Zarr implémentées en Java et C++

Revues des outils existants « Big Data »

Les spécifications de Zarr et N5 sont assez similaires (Z5 est un portage de N5 en C++/Python).

Cette uniformité involontaire est un bon signe: les deux groupes de développeurs ont convergé vers une solution commune qui semble raisonnable.

Zarr

Tableaux multidimensionnels simples et découpés en morceaux.

Chaque morceau est un fichier séparé sur le système de fichiers (ou une clé dans un stockage clé-valeur).

Compression configurable

Spécification simple

Implémentation de référence en Python.

Implémentations indépendantes

- Z5 / N5
- ndarray.scala (JVM + Javascript via Scala)

Très récent

En cours d'adoption par « Human Cell Atlas » après une analyse comparative approfondie de différents candidats

Zarr

Metadata (.zarray)

```
{
    "shape": [ 20, 20 ],
    "chunks": [ 10, 10 ],
    "compressor": {
        "id": "zlib",
        "level": 1
    },
    "dtype": "<i4",
    "fill_value": 42,
    "filters": null,
    "order": "C",
    "zarr_format": 2
}
```

Metadata (.zttrs)

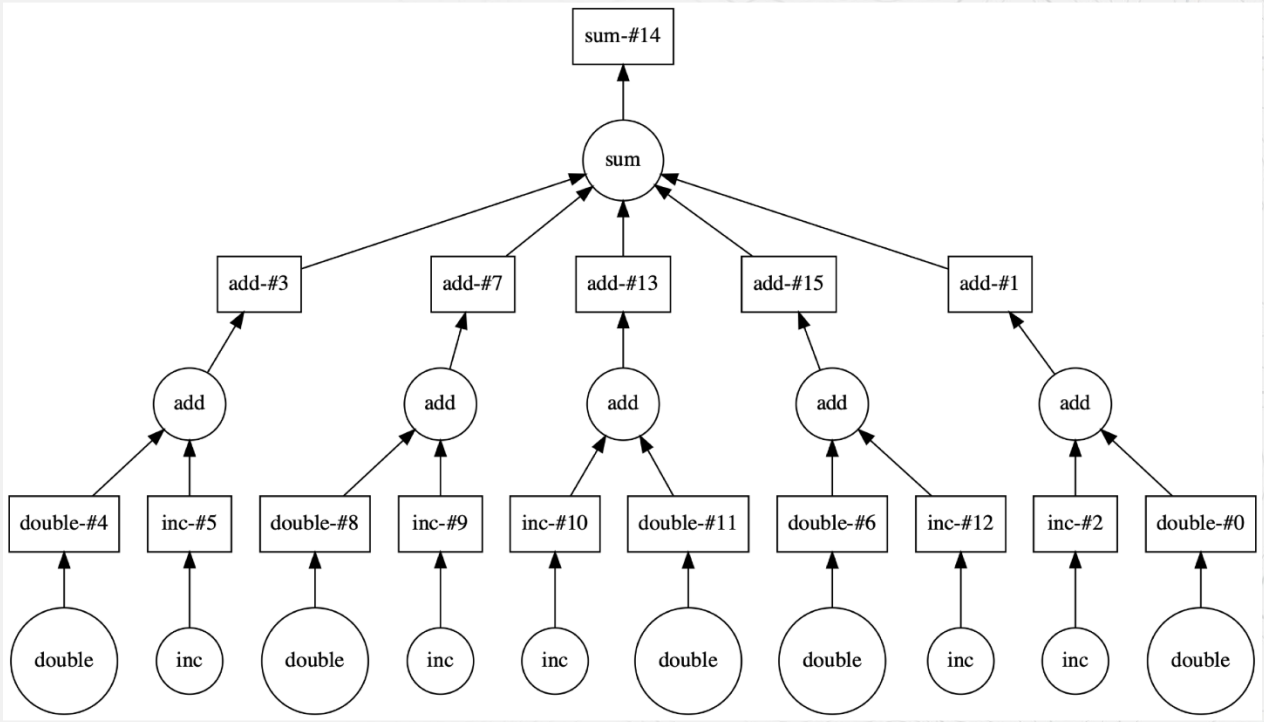
```
{
  "_ARRAY_DIMENSIONS": [
    "time",
    "face",
    "j",
    "i"
  ],
  "coordinates": "iters
dtime"
}
```

Empreinte sur le FS

```
Eta
├── 0.0.0.0
├── 0.1.0.0
├── 0.10.0.0
├── 0.11.0.0
├── 0.12.0.0
├── 0.2.0.0
├── 0.3.0.0
├── 0.4.0.0
├── 0.5.0.0
├── 0.6.0.0
├── 1.0.0.0
```

Dask

	8	8	8
5	('x', 0, 0)	('x', 0, 1)	('x', 0, 2)
5	('x', 1, 0)	('x', 1, 1)	('x', 1, 2)
5	('x', 2, 0)	('x', 2, 1)	('x', 2, 2)
5	('x', 3, 0)	('x', 3, 1)	('x', 3, 2)



Dask

Dask ajoute deux fonctionnalités majeures à NumPy :

La parallélisation

Opérations de streaming des tableaux

Dask permet de paralléliser sur un cluster, ou sur une seule machine et d'exécuter en séquentiel.

Xarray – Objectifs du design

- Pandas pour tableaux multidimensionnels
- Construits sur Pandas, NumPy et Dask
- Chaque opération dans Xarray est parallélisée avec Dask.
- Copie de l'API Pandas
- Utilise le modèle de données HDF
- Motivé par des cas d'utilisation sur des données météo et climatiques.

Xarray

Les opérations utilisent des noms par des nombres

```
# xarray style
>>> ds.sel(time="2012-11-28").max(dim="station")

# numpy style
>>> array[[0, 1, 2, 3], :, :].max(axis=2)
```

Xarray - apply_ufunc

```
def spearman_correlation(x, y, dim):
    return xarray.apply_ufunc(
        spearman_correlation_gufunc, x, y,
        input_core_dims=[[dim], [dim]],
        dask='parallelized',
        output_dtypes=[float])
```

Fonction qui prend en charge la diffusion NumPy

Dimensions sur lesquelles s'effectue le calcul

Parallélisation automatique avec Dask