

Calcul asynchrone en Python

Utilisation de la librairie `dask`



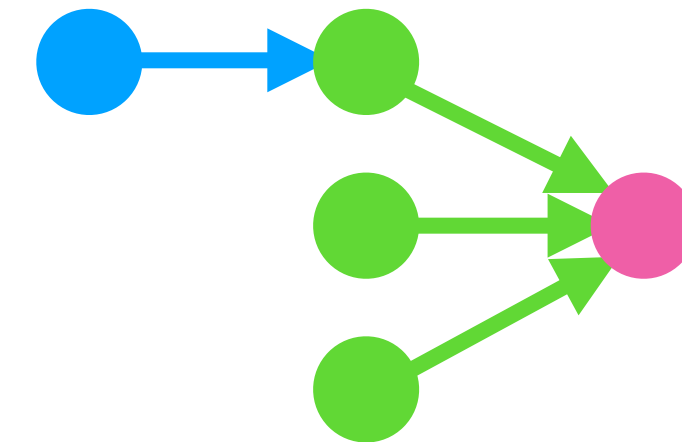
R. Tavenard & M. Ben

Calcul asynchrone

- Principe : lancer plusieurs calculs en parallèle sans attendre la fin de l'un pour débiter le suivant
- Map-reduce est un cas particulier
- Il existe des cas plus complexes

Exemple :

$$(3 + 7) * (2 + 8) * ((2 * 3) - 1)$$



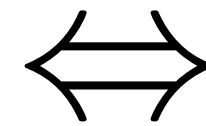
Calcul asynchrone en Python

Utilisation de la librairie `dask`

- Décoration de fonctions

```
import dask

@dask.delayed
def ma_fonction(x, y=1):
    # Faire des trucs qui
    # prennent du temps ici
```



```
def f(x, y=1):
    # Faire des trucs qui
    # prennent du temps ici

ma_fonction = dask.delayed(f)
```

- Déclare que la fonction sera exécutée en **asynchrone**

Calcul asynchrone en Python

Utilisation de la librairie `dask`

```
import dask

@dask.delayed
def ma_fonction(x, y=1):
    # Faire des trucs qui
    # prennent du temps ici

x = ma_fonction(3)
y = ma_fonction(x)
z = x + y
valeur_z = z.compute()
```

- Ici, le calcul ne commence qu'à la ligne `z.compute()`
- Les lignes précédentes ne sont pas bloquantes
- Dask se charge de construire le graphe des tâches à effectuer
- Possible de le visualiser avec `z.visualize()`

Calcul asynchrone en Python

Utilisation de la librairie `dask`

```
import dask

@dask.delayed
def ma_fonction(x, y=1):
    # Faire des trucs qui
    # prennent du temps ici

x = ma_fonction(3)
y = ma_fonction(x)
z = x + y
valeur_z = z.compute()
```

- Le type de `x`, `y`, `z` est modifié
 - Permet le `.compute()`
 - Opérations usuelles associées au type de base restent (ex : `x + y`)

Calcul asynchrone en Python

Utilisation de la librairie `dask`

```
import dask

@dask.delayed
def ma_fonction(x, y=1):
    # Faire des trucs qui
    # prennent du temps ici

x = ma_fonction(3)
y = ma_fonction(x)
z = x + y
z_val, x_val = dask.compute(z, x)
```

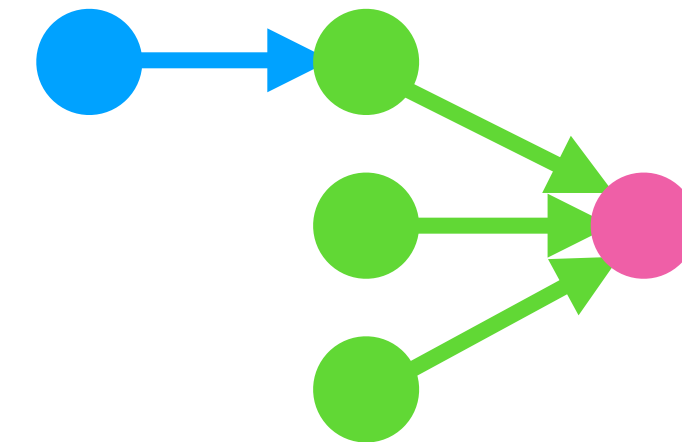
- Pour récupérer des valeurs de plusieurs variables :
`dask.compute()`

Calcul asynchrone en Python

Utilisation de la librairie `dask`

- Exercice : implémentez l'exemple de début du cours à l'aide de `dask`

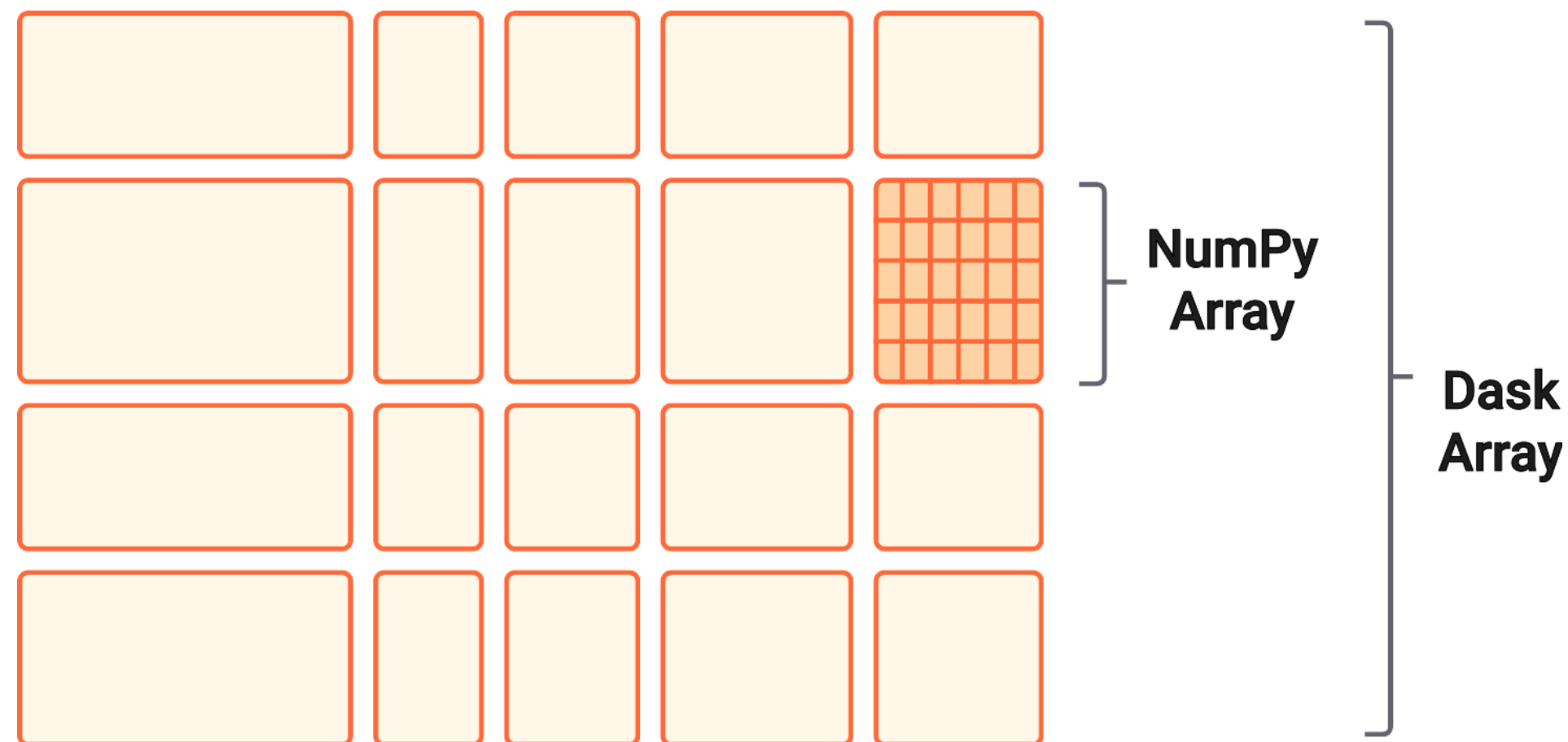
$$(3 + 7) * (2 + 8) * ((2 * 3) - 1)$$



Collections spécifiques à dask

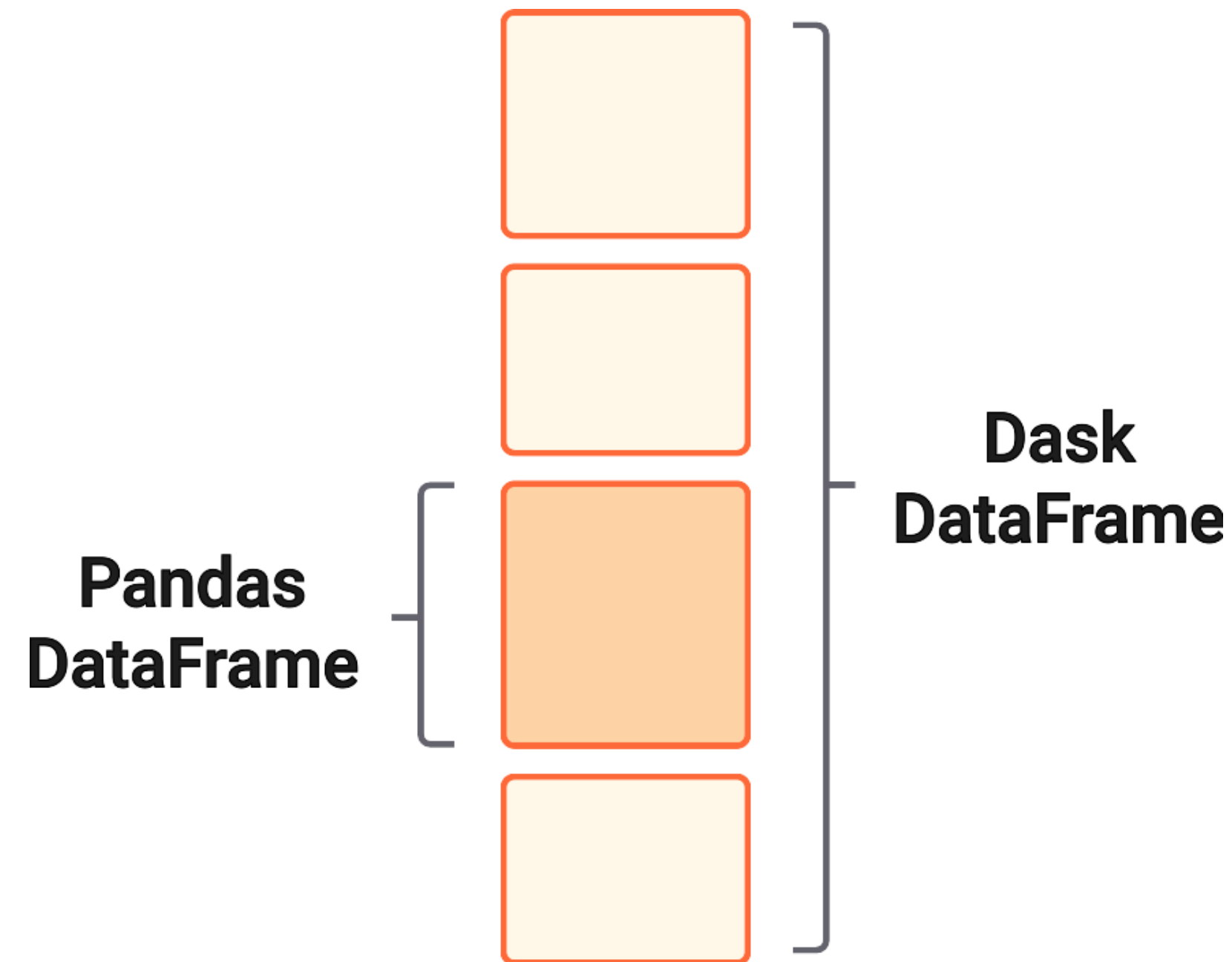
- Data Science
 - Types massivement utilisés
 - np.array
 - pd.DataFrame
- Dask propose des réimplémentations de ces types
 - Pour les cas où les données ne tiennent pas en mémoire
 - Pour paralléliser les traitements

dask.array



- Principe
 - Ensemble de blocs (découpage selon chaque dimension)
 - 1 bloc = 1 `np.array`
- Calculs parallèles
 - Utilise tous les coeurs disponibles par défaut
 - Réimplémentation des algorithmes standards par bloc
- Possible d'allouer des array plus grands que la mémoire

dask.DataFrame



- Principe
 - Ensemble de blocs
(1 bloc = 1 ensemble d'individus)
 - 1 bloc = 1 `pd.DataFrame`
- Même logique que `dask.array`
 - Calculs parallèles par bloc
 - Réimplémentation des fonctions pandas usuelles