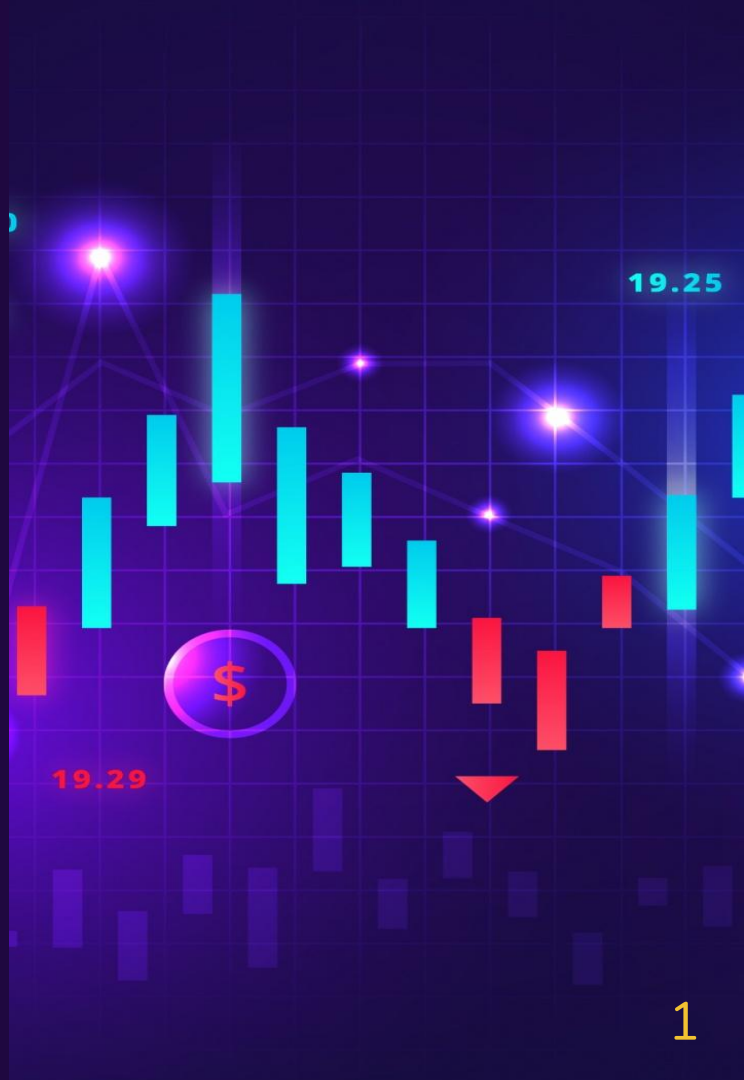
ENGIE

Profiling du code d'algorithme de trading



Louis Boulay
Maxime Clergé
Pacôme Lefebvre

Projet 04
Encadrant : Axel Breuer

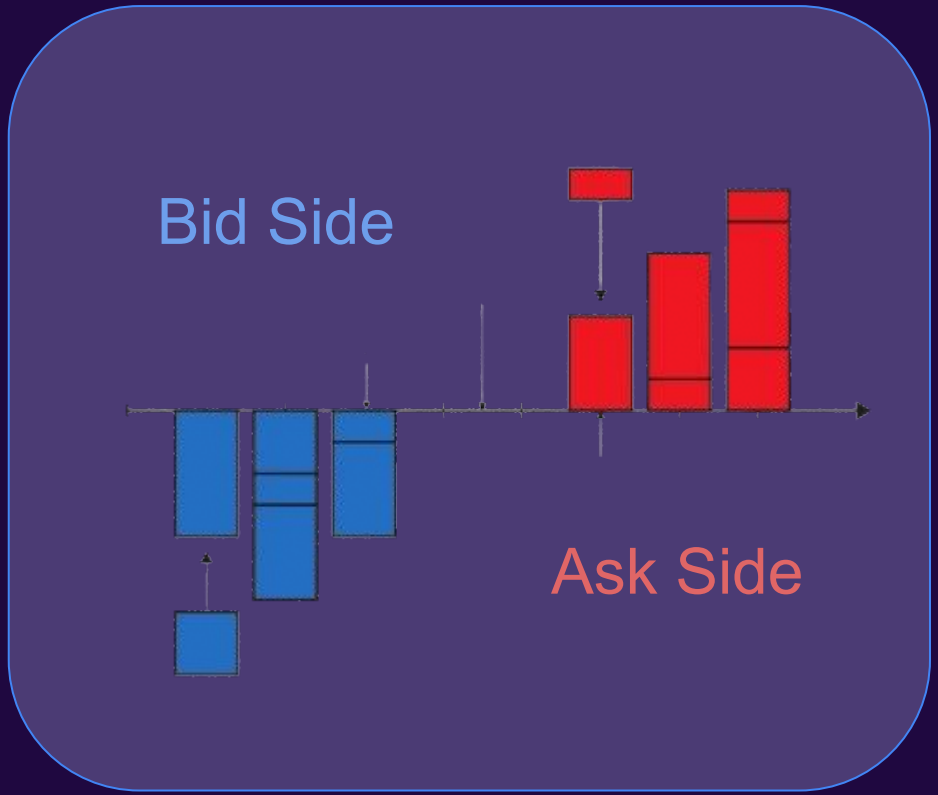
Introduction

Qui est ENGIE ?

- Un des leaders de la production et distribution d'énergie
- 100.000 employés
- 83 Mds de chiffre d'affaire

- > C'est sur la partie trading de ressources que nous allons travailler

Limit Order Book (LOB)



Concepts clés :

- Bids : priorité au plus haut
- Asks : priorité au plus bas
- Spread : écart entre le plus grand bid et le plus petit ask

Composition de nos fichiers ?

Parquet

Optimisé : lecture et écriture

Écriture en colonne : plus lisible qu'en lignes

Big Data : adapté aux donnés volumineuses

Trades

Date (à la seconde) : trace de l'heure de l'échange

Prix : prix d'une unité de l' échange

Volume : nombre d'unité échangée

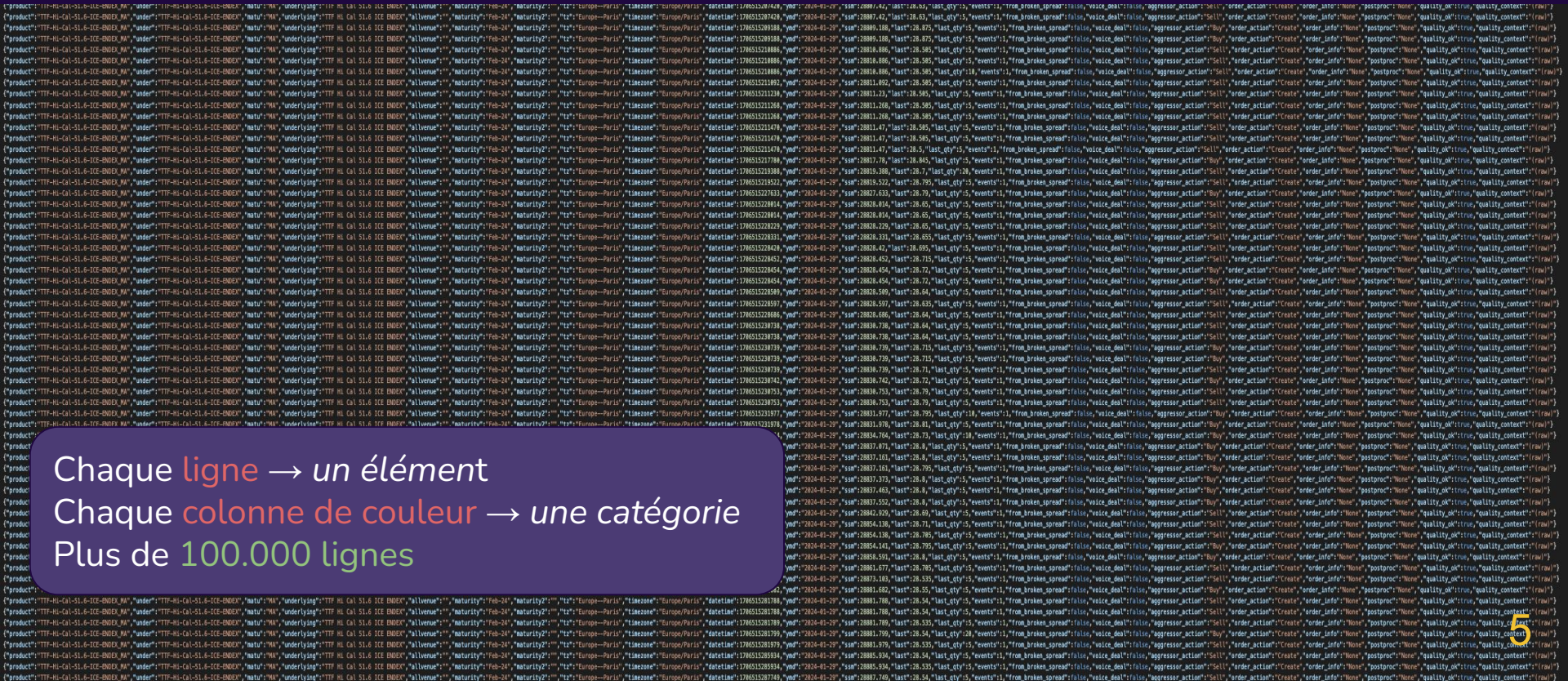
Quotes

Même infos que Trades (mais en demande)

Ask : meilleur prix à la vente

Ask Quantity : quantité proposée au meilleur prix

Exemple d’un *.parquet*



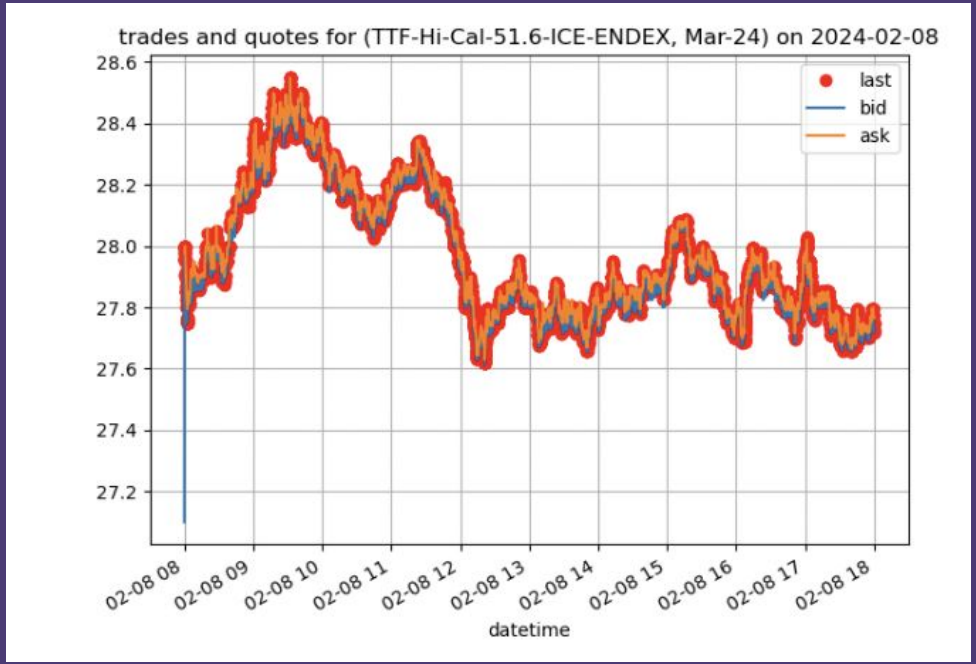
Chaque ligne → *un élément*
Chaque colonne de couleur → *une catégorie*
Plus de 100.000 lignes

Principe du Backtester

Utilisation de fichiers *.parquet*

Fichiers trades et quotes en format parquet

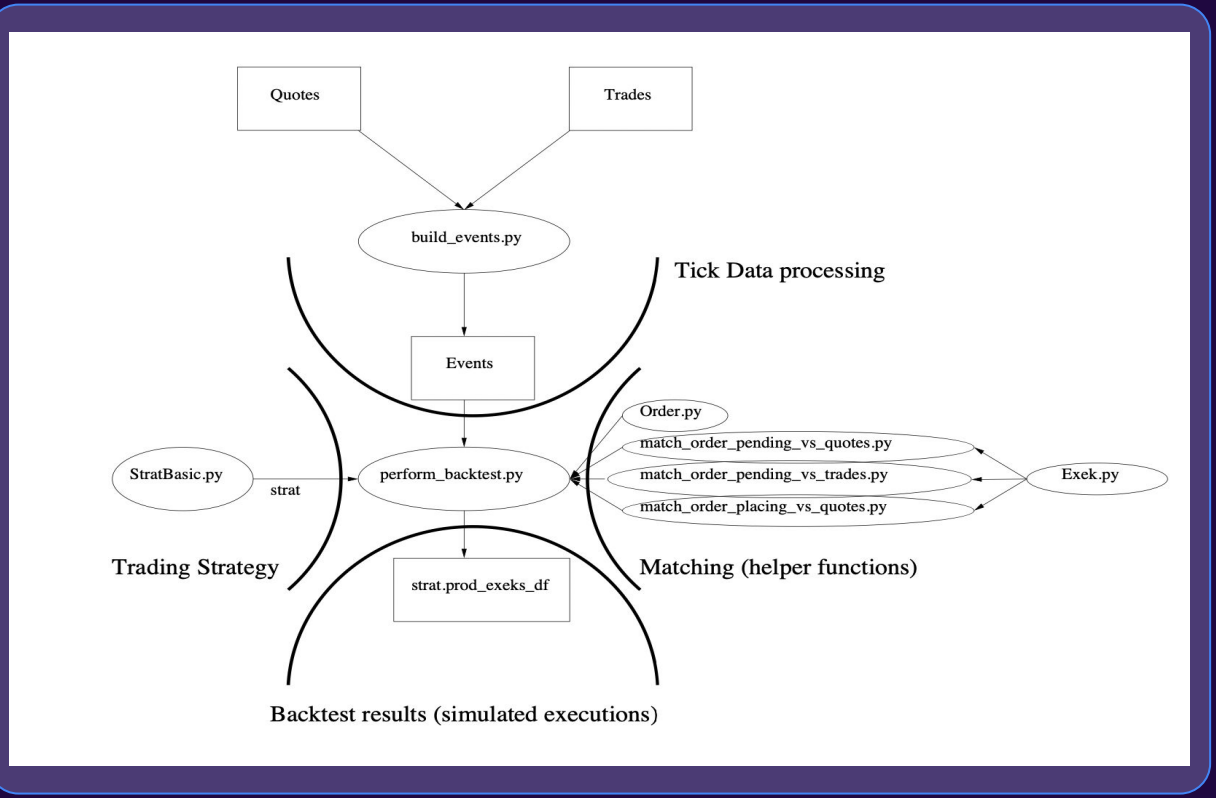
Transformation en dataframe



Backtester d'Engie

15 fichiers Python

Utilisation de dataframe avec pandas



Backtester d’Engie

Objectif : tester une stratégie

Stratégie : Vendre 40 unités à un prix unitaire de 28.51€

Backtester

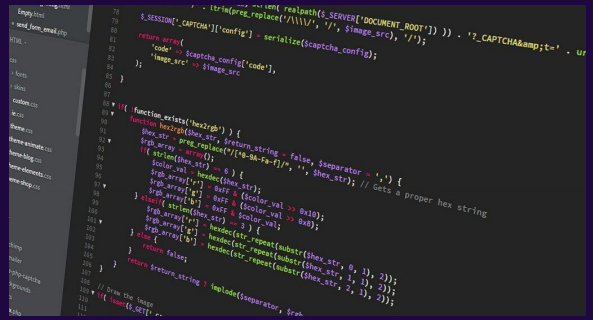
Trades et quotes modifiés

Temps d'exécution

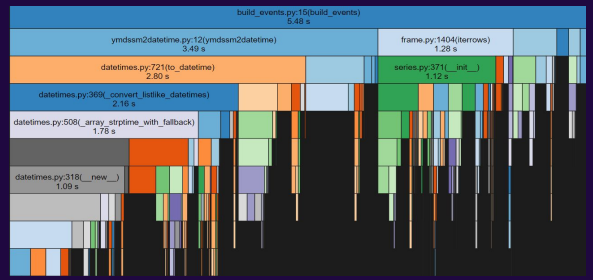
Profiling

➔ Profiling pour analyse du temps d'exécution

- Diagnostic des fonctions les plus chronophages
- Utilisation de Snakeviz pour visualiser les données.



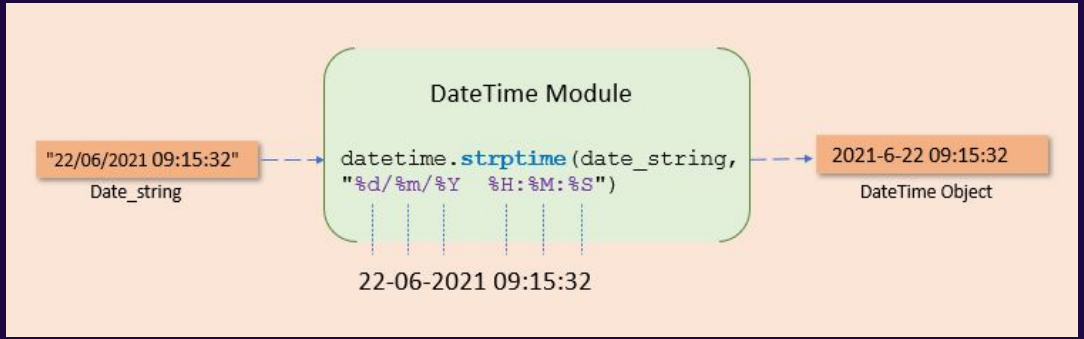
Profiling



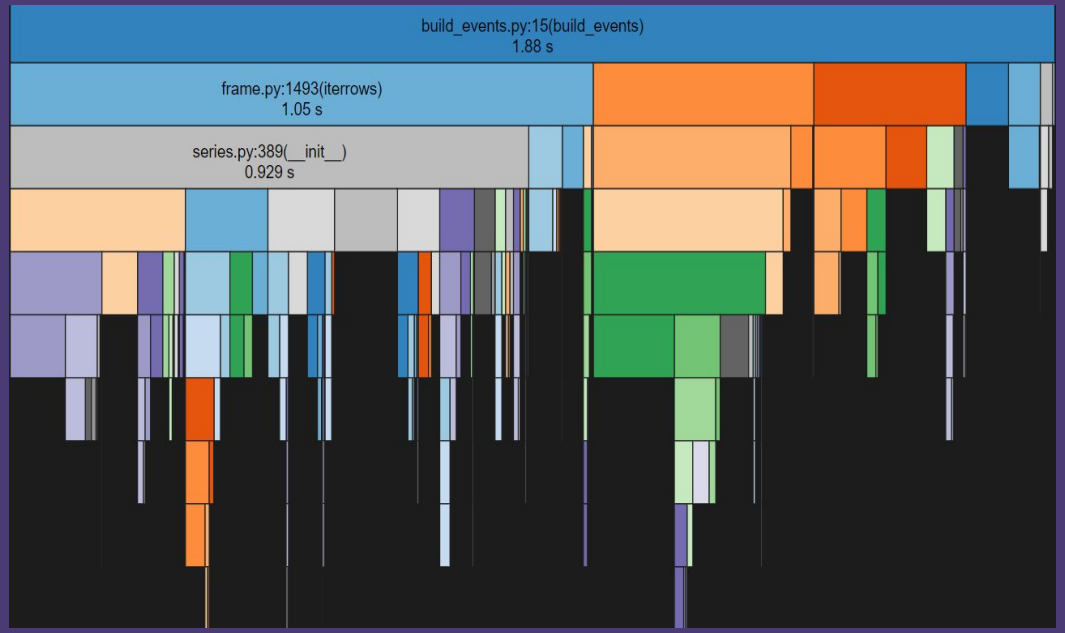
Amélioration du code

```
datetime = pd.to_datetime(ymd, format='%Y-%m-%d').to_pydatetime()
timedelta = pd.to_timedelta(ssm, unit='s').to_pytimedelta()
dt = datetime + timedelta
```

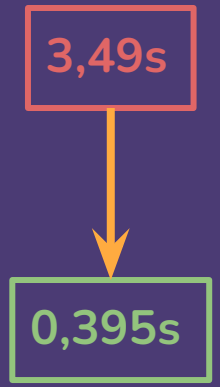
```
dt = datetime.strptime(ymd, '%Y-%m-%d') + timedelta(seconds=ssm)
```



Profiling 2.0



Pour la fonction `ymdssm2datetime` :



Division du temps d'un rapport **8,83**

Fusion : le principe

Problèmes ?

Testé sur peu de données

Problèmes d'échelle ?

Format des données peu pratique

Solution proposées :

Récupérer tous les fichiers *.parquet*

Les lire les uns après les autres ?

Les fusionner en une seule fois ?

→ Nous allons fusionner en une fois pour gagner du temps

Fusion : le code

```python
def find_all_parquet_files(base_dir):
    """
    Trouve récursivement tous les fichiers Parquet dans un répertoire de base.
    """
    parquet_files = []
    for root, dirs, files in os.walk(base_dir):
        for file in files:
            if file.endswith(".parquet"):
                parquet_files.append(os.path.join(root, file))
    return parquet_files
```

```python
# Lire les fichiers Parquet et les concaténer
nb_fichier_trouve = 0
df_list = []
for file in parquet_files:
    try:
        df = pd.read_parquet(file)
        df_list.append(df)
        nb_fichier_trouve +=1
    except Exception as e:
        print(f"Erreur lors de la lecture du fichier {file}:", e)
```

Piste d'amélioration

Réécriture du code :

- Utilisation de Cython
- C++
- Utilisation de Numba

Utilisation de meilleurs machine :

- De grand écart de temps ont été vue suivant les machine qui font tourner le code.

Conclusion

Merci de votre écoute !

Annexe

```
def ymdssm2datetime(ymd, ssm):
    """
    AIM:
    Convert time expressed in datetime
    into time expressed in Year Month Day (YYYY-MM-DD)

    INPUT:
    ymd: string (or list of strings) in yyyy-mm-dd format
    ssm: float or array of floats representing seconds

    OUTPUT:
    dt: datetime
    """
    if isinstance(ssm, (float, np.float64)):
        dt = datetime.strptime(ymd, '%Y-%m-%d') + timedelta(seconds=ssm)  # convert to datetime
    else:
        dt = [ymdssm2datetime(ymd, ssm[idx]) for idx in range(len(ssm))]

    return dt
```

Annexe

```
with cProfile.Profile() as profile:
    events = build_events(trades_df,
                          quotes_df,
                          ticks_from_hms,
                          ticks_to_hms,
                          strat_params["heartbeat_dt"],
                          strat_params["trade_from_hms"],
                          strat_params["trade_to_hms"])

    strat = StratBasic(strat_params)

    product = under + "_" + matu
    perform_backtest(strat,
                     product,
                     events,
                     verbose=True)
results = pstats.Stats(profile)
results.sort_stats(pstats.SortKey.TIME)
results.dump_stats("Hourrabis.prof")
```