

Polars

Hermilo

4 de Septiembre 2025

Polars : Query Engine for DataFrames



Polars

Polars : Query Engine for DataFrames

Polars es una biblioteca desarrollada en Rust que integra las siguientes características por diseño:

- **Velocidad** : Rust es un lenguaje de system programming conocido por su desempeño y seguridad.
- **Paralelismo** : Aprovecha arquitecturas multicore e implementa algoritmos paralelos de work stealing.
- **Eficiencia de memoria**:
 - Polars utiliza evaluaciones lazy, lo que significa que una operación no es realizada hasta que esta es necesitada.
 - Las consultas pueden ser encadenadas y optimizadas antes de su ejecución, lo que se traduce en ejecuciones más eficientes de queries.
- **Almacenamiento eficiente de datos** : Polars utiliza Apache Arrow como modelo de almacenamiento en memoria. Es decir, utiliza un formato columnar del almacenamiento de datos, lo cual resulta más eficiente que el tradicional almacenamiento basado en filas (como el utilizado por Pandas).
- Diseñado para **out-of-core processing** (más grande que la RAM).

Polars : Query Engine for DataFrames



Polars

Polars : Query Engine for DataFrames



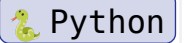
Polars

Aceleración impulsada por NVIDIA-RAPIDS

Consideraciones de diseño :

- Misma semántica.
- Optimizador.
- Parte de la biblioteca de Polars (no es necesario `import cudf_polars as pl`)

```
1 query = (  
2     transaction.groupby("cust_id").agg(  
3         pl.cum("amount")  
4     ).sort(by = "amount", descending = True)  
5     .head()  
6 )  
7  
8 # Run on the CPU  
9 result_cpu = query.collect()  
10  
11 # Run on the GPU  
12 result_gpu = query.collect(engine="gpu")  
13  
14 # assert both result are equal  
15 pl.testing.assert_frame_equal(result_cpu,  
                                result_gpu)
```

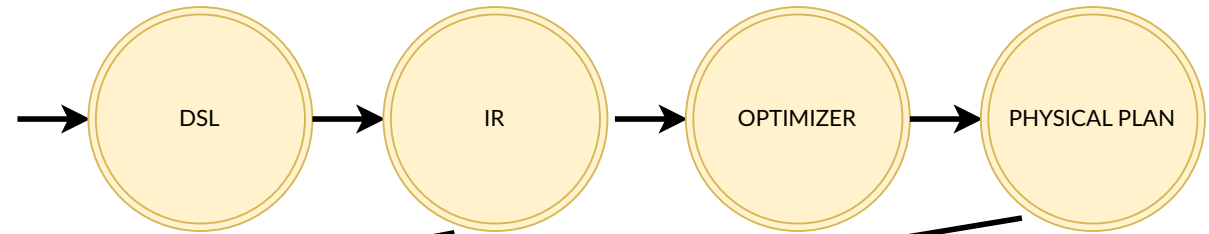


Roadmap de un query

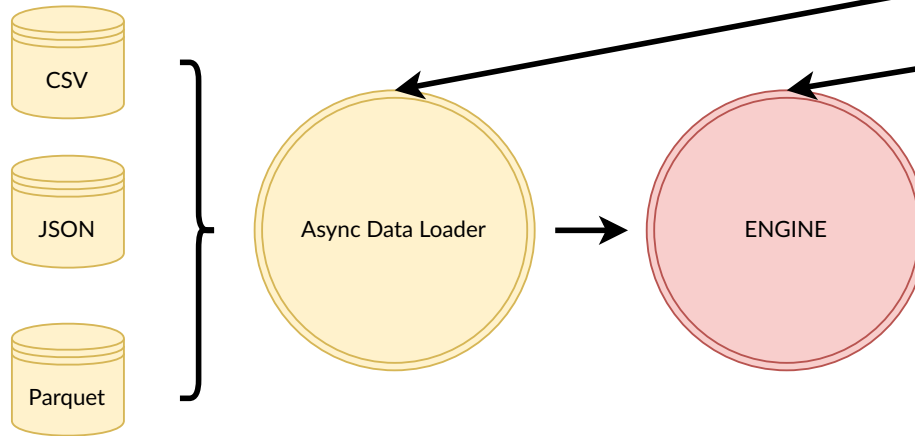
Logical

```
1 (  
2   pl.scan_parquet("my_file.parquet")  
3   .filter(pl.col("dogs")== "Retriever")  
4   .select("dogs", "origins", "weight")  
5   .head(10)  
6   .collect(engine="gpu")  
7 )
```

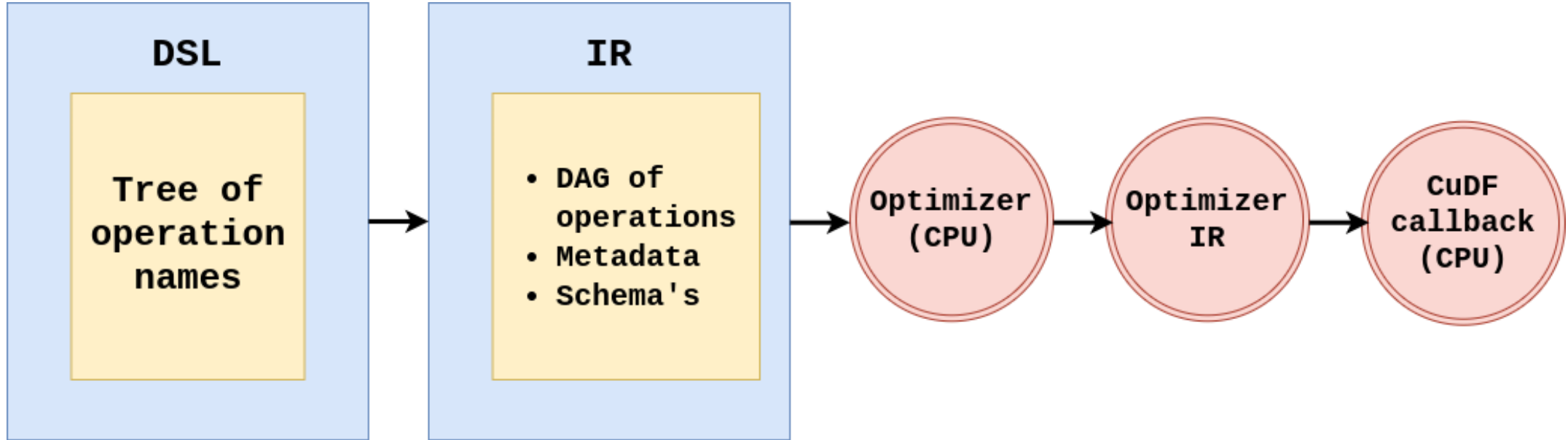
Python



Physical



Fase Lógica




Optimizaciones

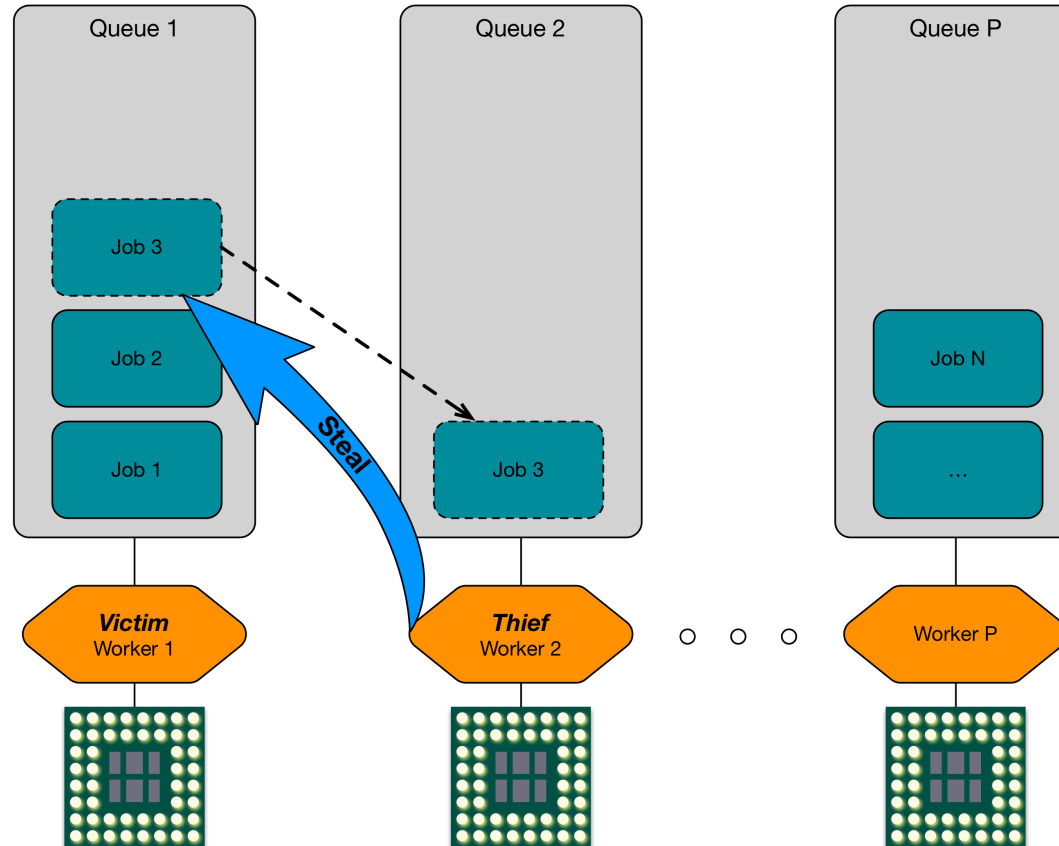
Vectorización ofrecida por NumPy:

- SI : Performance de C.
- NO : Optimizaciones.

```
1  import polars as pl
2
3  q = (
4      pl.scan_csv('../datos/flights.csv')
5      .select(['MONTH', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT'])
6      .filter(
7          (pl.col('MONTH') == 5) &
8          (pl.col('ORIGIN_AIRPORT') == 'SFO') &
9          (pl.col('DESTINATION_AIRPORT') == 'SEA'))
10 )
11 %time
12 df = q.collect(no_optimization=True)
13 CPU times: user 3 µs, sys: 0 ns, total: 3 µs
14 Wall time: 5.48 µs
15 %time
16 df = q.collect()
17 CPU times: user 5 µs, sys: 0 ns, total: 5 µs
18 Wall time: 15.7 µs
```

 Python

Paralelismo : Work Stealing



Polars Expressions

¿Qué es una expresión?

Una expresión es un árbol de operaciones que describen cómo construir una o más Series.

Polars Expressions

```
In [9]: (  
...: (  
...:     pl.lit(3).add(5)  
...:     /  
...:     pl.lit(1).add(5)  
...: )  
...: .meta  
...: .tree_format()  
...: )
```

