MAKE THE SWITCH

POLARS

LIGHTNING-FAST DATAFRAMES

TOBIAS

THOMAS

DENIS

HOPES, FEARS & DECISIONS

PERFORMANCE

SIMPLICITY

SIMPLE
CONSISTENT
GRAMMAR

7 VERBS GET MOST JOBS DONE

```
# select/slice columns
select

# create/transform/assign columns
with_columns

# filter/slice/query rows
filter

# join/merge another dataframe
join

# group dataframe rows
groupby

# aggregate groups
agg

# sort dataframe
sort
```

```
articles = pl.read_parquet("articles.parquet")
sales = pl.read_parquet("sales.parquet")
monthly_best_sellers_2019 = (
    sales
    .with_columns([
        pl.col("date").dt.year().alias("year"),
        pl.col("date").dt.month().alias("month"),
    ])
    .filter(pl.col("year") == 2019)
    .join(articles, on="article_id")
    .groupby(["product_code", "month"])
    .agg(pl.col("price").sum().alias("total_sales"))
    .filter(
        pl.col("total_sales") ==
        pl.col("total_sales").max().over("month")
    )
    .select(["month", "product_code"])
    .sort("month")
)
```

PANDAS

```python
# deprecated
df.ix[10:11, ["foo", "bar"]]
# label based
df.loc[["a", "b"], ["foo", "bar"]]
# position based
df.iloc[[10, 11], [2, 3]]
# multi-index
df.loc[pd.IndexSlice[:, ["a", "b"]], ["foo", "bar"]]
# reset index
df.reset_index().groupby("foo", as_index=False)
```

POLARS

```python
# there is
# no index
```

EXPRESSIONS

CUSTOM COMPOSABLE EXPRESSIONS

```python
# convert a date to a yearmonth number
def as_year_month(date: str | pl.Expr) -> pl.Expr:
    date = get_col_expr(date)
    return date.dt.year() * 12 + (date.dt.month() - 1)

# calculate the difference in months between two dates
def month_diff(end: str | pl.Expr, start: str | pl.Expr) -> pl.Expr:
    return as_year_month(end) - as_year_month(start)
```

```python
from utils.polars import month_diff

avg_product_months_when_sold = (
    sales
    .join(articles, on="article_id")
    .groupby("product_code")
    .agg(month_diff(start=pl.col("date").min(), end="date").mean().alias("avg_product_months"))
)
```

LAZY EVALUATION

LAZY/SCAN → COLLECT → FASTER QUERIES ☺

```python
# eager
articles = pl.read_parquet("articles.parquet")
avg_articles_per_product = (
    articles
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
)
```

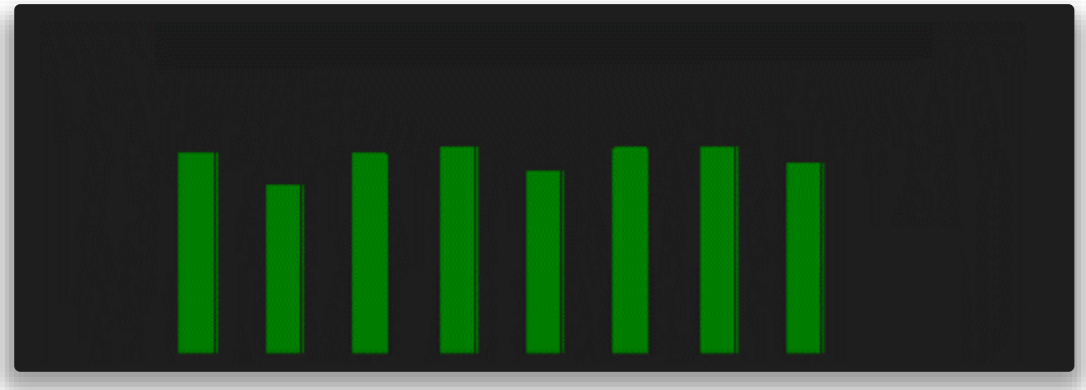
```python
# eager loading, lazy query
articles = pl.read_parquet("articles.parquet")
avg_articles_per_product = (
    articles
    .lazy()
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
    .collect()
)

# lazy loading & query
articles = pl.scan_parquet("articles.parquet")
avg_articles_per_product = (
    articles
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
    .collect()
)
```

PERFORMANCE

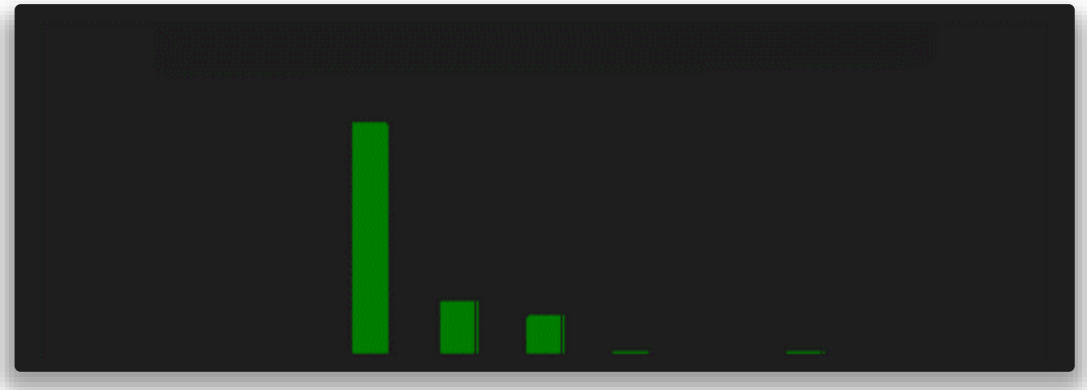
POLARS

```python
#  32m sales transactions
# 105k articles, 47k products
first_product_sales = (
  sales
  .join(articles, on="article_id")
  .groupby("product_code")
  .agg(
    pl.col("week").min().alias("prd_1st_wk"),
    pl.col("yearday").min().alias("prd_1st_yd"),
  )
)
```



PANDAS

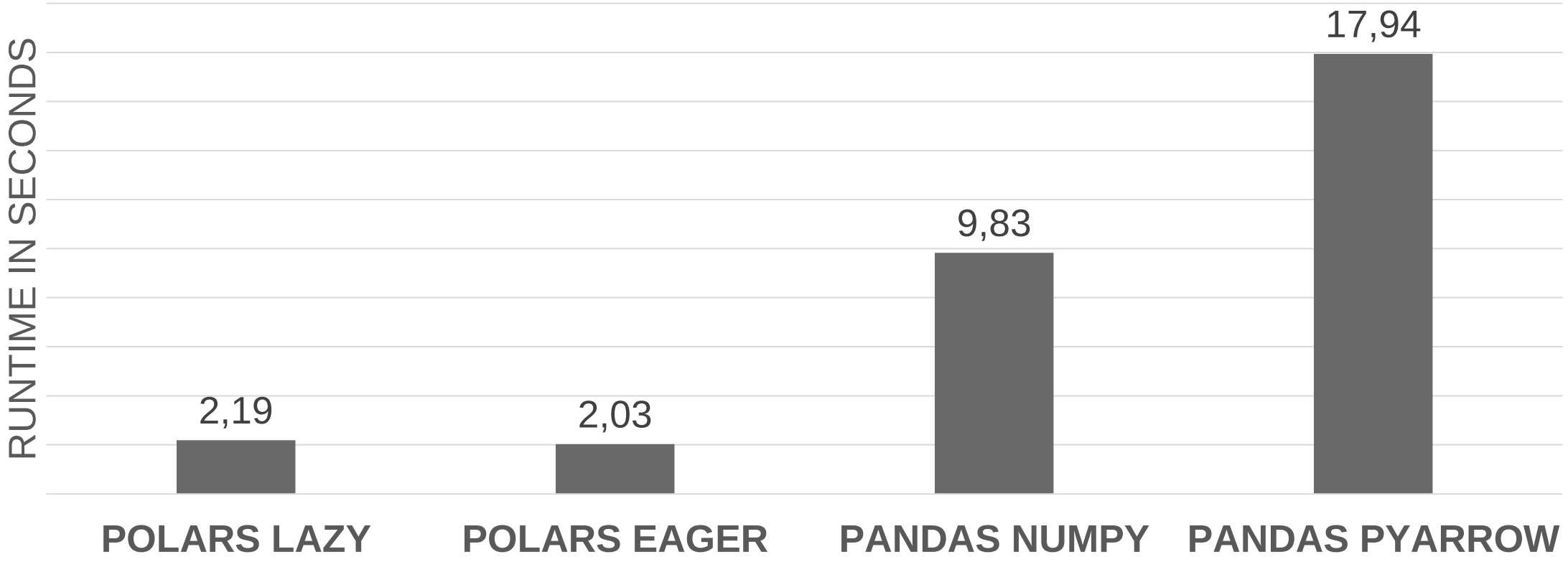
```python
#  32m sales transactions
# 105k articles, 47k products
first_product_sales = (
  sales
  .merge(articles, on="article_id")
  .groupby("product_code")
  .agg(
    prd_1st_wk=("week", "min"),
    prd_1st_yd=("yearday", "min")
  )
)
```



8 CORE LAPTOP, 16 GB RAM, PANDAS 2.0.0, POLARS 0.17.2

FIND FIRST PRODUCT SALES FROM 32M TRANSACTIONS

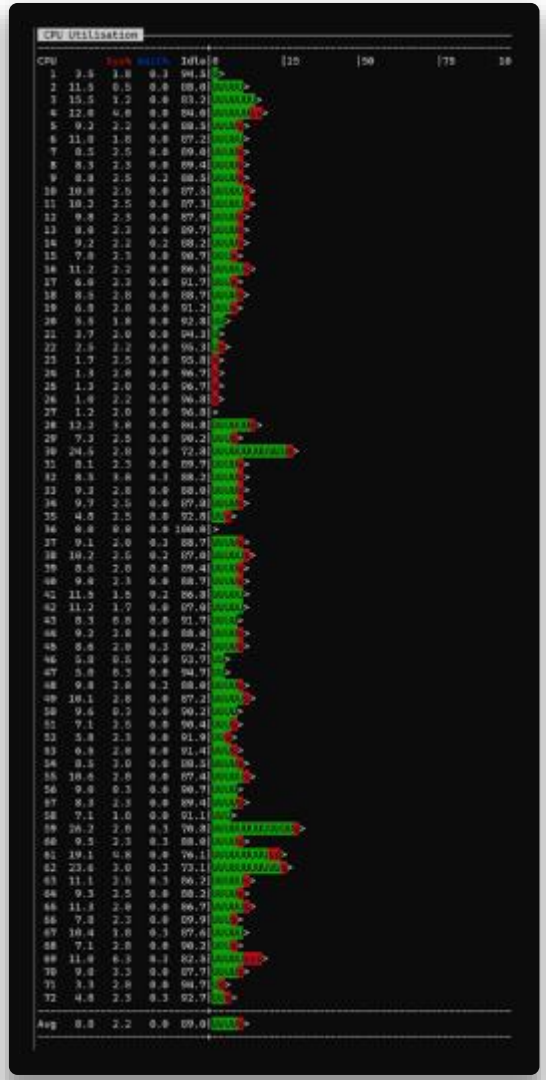
MANUALLY LIMITED LOADED COLUMNS



8 CORE LAPTOP, 16 GB RAM, PANDAS 2.0.0, POLARS 0.17.2

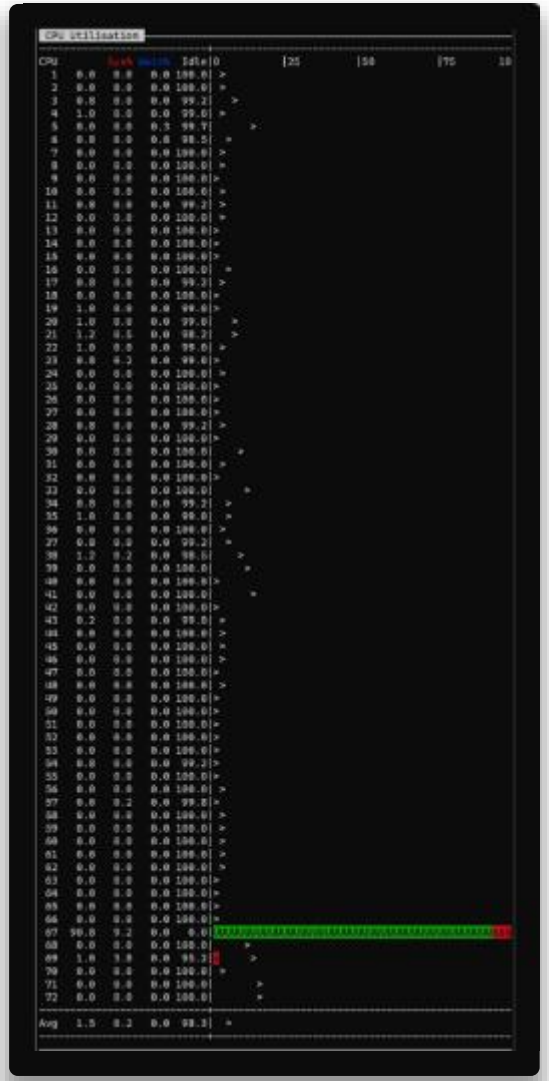
72 CORES
640 GB RAM

POLARS LAZY



POLARS EAGER

PANDAS

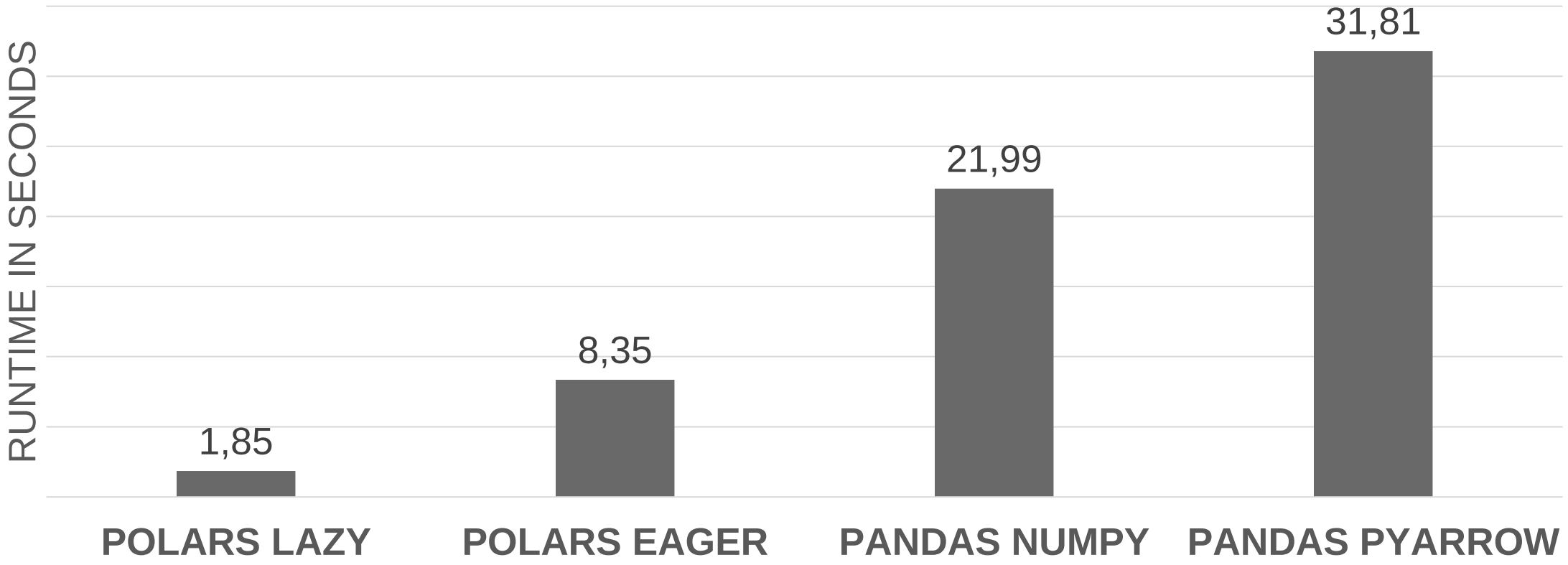


72 CORE SERVER, 640 GB RAM, PANDAS 2.0.0, POLARS 0.17.2

BettercallxPaul

FIND FIRST PRODUCT SALES FROM 32M TRANSACTIONS

COLUMNS NOT MANUALLY LIMITED



72 CORE SERVER, 640 GB RAM, PANDAS 2.0.0, POLARS 0.17.2

LARGER THAN LIFE RAM?

OUT OF CORE PROCESSING!

LARGER THAN RAM? STREAMING=TRUE!

```
# stream to memory
articles = pl.scan_parquet("articles.parquet")
avg_articles_per_product = (
    articles
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
    .collect()
)
```

LARGER THAN RAM? STREAMING=TRUE!

```python
# stream to memory
articles = pl.scan_parquet("articles.parquet")
avg_articles_per_product = (
    articles
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
    .collect(streaming=True)
)
```

```python
# stream to disk
articles = pl.scan_parquet("articles.parquet")
(
    articles
    .groupby("product_code")
    .agg(pl.count())
    .select(pl.col("count").mean())
    .sink_parquet("result.parquet")
)
```

WATCH OUT:
STILL QUITE ALPHA

LEARNING CURVE
COMMUNITY & SUPPORT

[polars-python]

see https://pola.rs

4 BUG + 2 FEATURE + 1 ? IN A YEAR

1. BUG 1 day
Allow join on same categorical source

2. BUG 18 days
Keep categorical type in comparison

3. BUG 2 months
Inconsistent behavior of sum().over()

4. BUG 1 day
Aggregation fails when grouped by multiple columns with one column of datatype Int8 or Int16

5. FEATURE 2.5 months
Allow expressions for date_range parameters

6. FEATURE 1 day
Support datetime.date in date_range

7. DEBATABLE open
Selecting columns by type (dtype) fails for filter

ALMOST WEEKLY RELEASES

March 15, 2023 – April 15, 2023

Period: 1 month

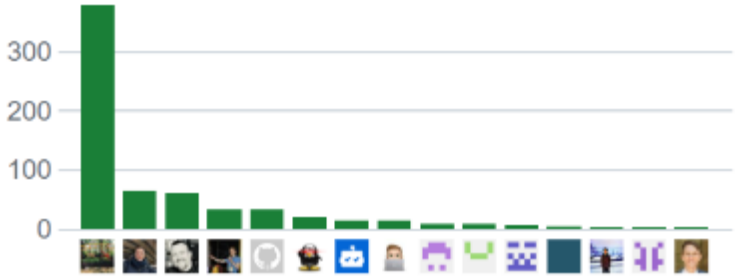
Overview

336 Active pull requests

388 Active issues

313	23	209	179
Merged pull requests	Open pull requests	Closed issues	New issues

Excluding merges, **31 authors** have pushed **313 commits** to main and **619 commits** to all branches. On main, **551 files** have changed and there have been 26,127 **additions** and 12,308 **deletions**.



ALTERNATIVES

OUR CHOICE

SINGLE COMPUTE WORKLOADS

WHATEVER WORKS FOR THE TEAM AT HAND

DISTRIBUTED WORKLOADS


THANK YOU!
MAKE THE SWITCH!?

datenzauber.ai

bcxp.de

YOUR QUESTIONS!