

Introducing FireDucks: A High-performance Compiler-accelerated DataFrame Library

July 25, 2024

Sourav Saha (NEC)

Quick Introduction!



SOURAV SAHA – Research Engineer @ NEC Corporation

<https://www.linkedin.com/in/sourav-%E3%82%BD%E3%82%A6%E3%83%A9%E3%83%96-saha-%E3%82%B5%E3%83%8F-a5750259/>

<https://twitter.com/SouravSaha97589>

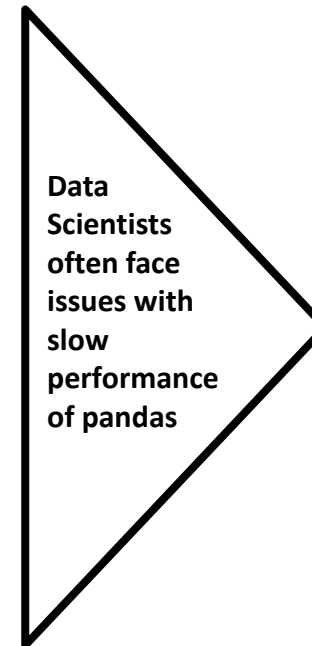
Hello, I am a software professional with 11+ years of working experience across diverse areas of **HPC, Vector Supercomputing, Distributed Programming, Big Data and Machine Learning**. Currently, my team at NEC R&D Lab, Japan, is researching various data processing-related algorithms. Blending the mixture of different niche technologies related to compiler framework, high-performance computing, and multi-threaded programming, we have developed a Python library named FireDucks with highly compatible pandas APIs for DataFrame-related operations.



Mr. Kazuhisa Ishizaka
(Primary Author)

we wanted to develop some library using compiler technology

we wanted to speed-up python



User Program

pandas API

FireDucks

groupby

join

dropna

filter

sort

corr

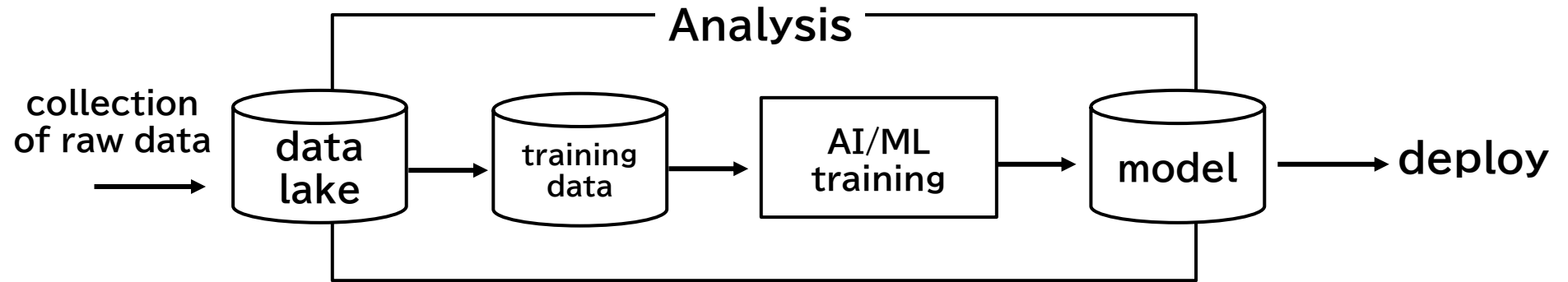
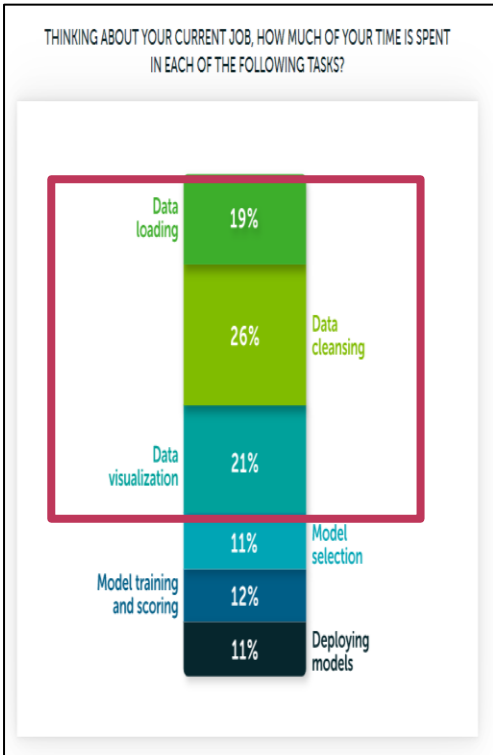
compiler technologies



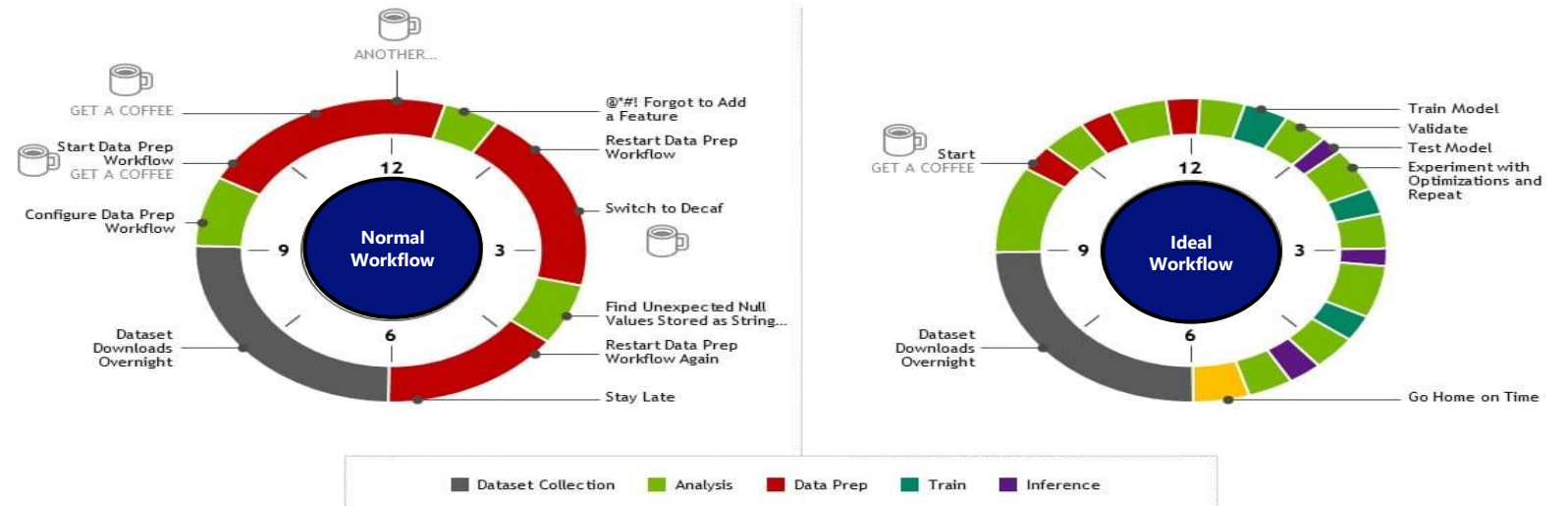
<https://www.nec.com/en/global/solutions/hpc/sx/index.html>

Workflow of a Data Scientist

almost 70% of efforts of a Data Scientist



DAY IN THE LIFE OF A DATA SCIENTIST



Anaconda:
The State of Data Science 2020

src: <https://blogs.nvidia.com/blog/accelerated-data-science-hpc/>

Background: What is DataFrame?

- ◆ DataFrame is a tabular data structure used for **table data analysis**.
- ◆ Table data is often used in data analytics to solve business problems.

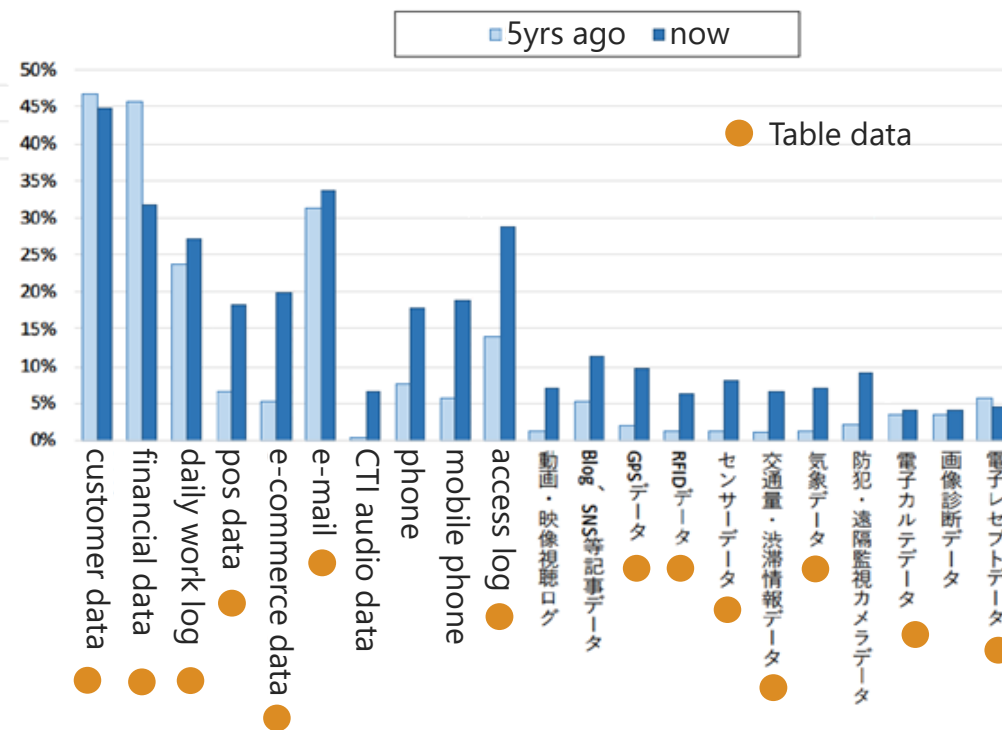
	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1325317920	4.39	4.39	4.39	4.39	0.455581	2.000000	4.390000
1	1325317980	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1325318040	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1325318100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1325318160	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
4857372	1617148560	58714.31	58714.31	58686.00	58686.00	1.384487	81259.372187	58692.753339
4857373	1617148620	58683.97	58693.43	58683.97	58685.81	7.294848	428158.146640	58693.226508
4857374	1617148680	58693.43	58723.84	58693.43	58723.84	1.705682	100117.070370	58696.198496
4857375	1617148740	58742.18	58770.38	58742.18	58760.59	0.720415	42332.958633	58761.866202
4857376	1617148800	58767.75	58778.18	58755.97	58778.18	2.712831	159417.751000	58764.349363

4857377 rows × 8 columns

Bitcoin historical prices

<https://www.kaggle.com/datasets/mczielinski/bitcoin-historical-data>

Data used in data analytics

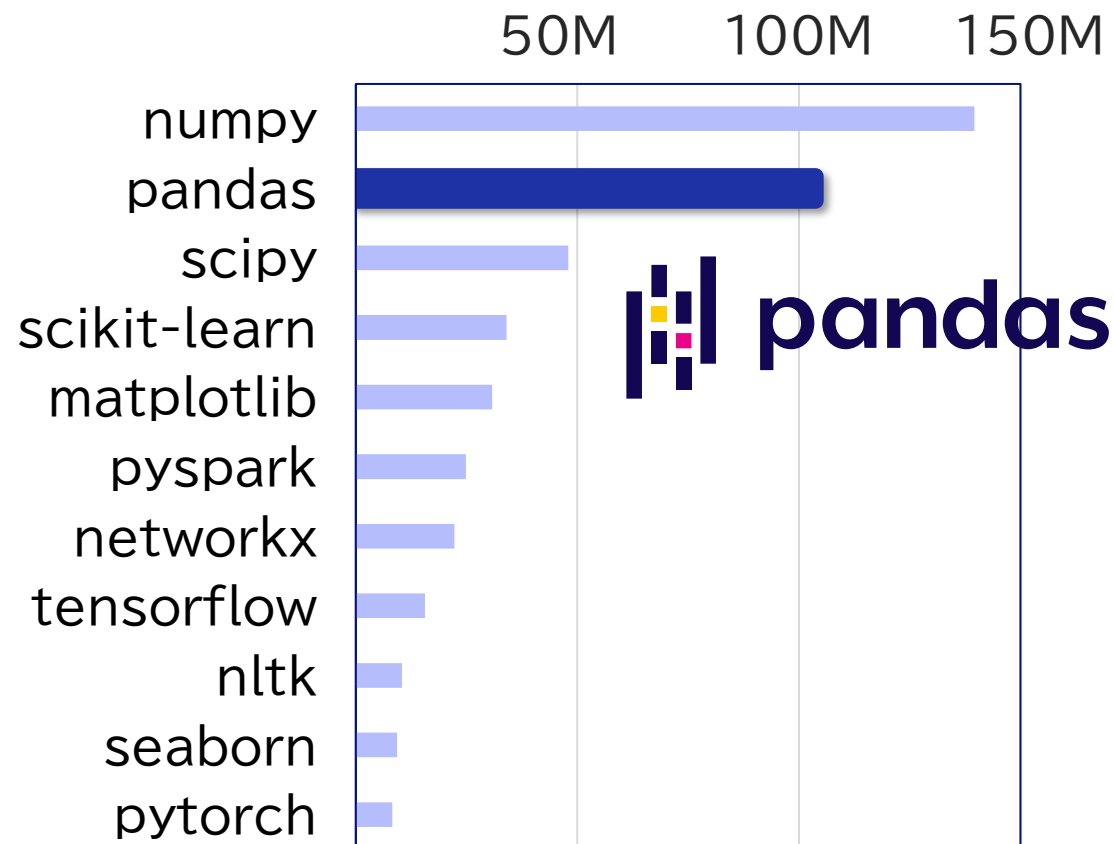


Ministry of Internal Affairs and Communications, Japan, 2020

<https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/html/ne220000.html>
(partially edited by NEC)

Background: What is pandas?

◆ Most popular Python library for data analytics.



Monthly download from pypi.org (Data Analytics Libraries)



Books



Courses

<https://www.udemy.com/ja/topic/pandas/>

Challenges in Data Manipulation with pandas

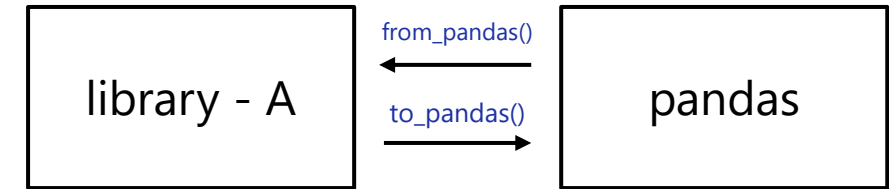


- It (mostly) doesn't support parallel computation.
- It doesn't have any auto-optimization feature.
- The choice of API heavily impacts the performance of a pandas application.
- Very slow execution reduces the efficiency of a data analyst.
- Long-running execution
 - produces higher cloud costs
 - attributes to higher CO2 emission

Challenges in Migration from pandas

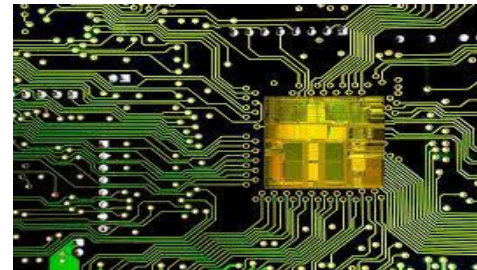
High Migration Cost:

- Needs to learn new library and their interfaces
- Manual fallback to pandas when target library doesn't support a method used in an existing pandas application
- Performance can be evaluated, and result can be tested after the migration completes.



High Hardware Cost:

- Needs to upgrade the existing execution system to leverage high-spec CPU, GPU etc.
- High-spec system incurs additional cost.



Introducing FireDucks

FireDucks (**F**lexible **IR** Engine for DataFrame) is a high-performance compiler-accelerated DataFrame library with highly compatible pandas APIs.

Speed: significantly faster than pandas

- FireDucks is multithreaded to fully exploit modern processor
- FireDucks optimizes user program at runtime by embedded **runtime compiler**

Ease of use: drop-in replacement of pandas

- FireDucks is highly compatible with pandas API
- No extra learning is required
- No code modification is required



Usage of FireDucks

1. Explicit Import

easy to import

```
# import pandas as pd
import fireducks.pandas as pd
```

simply change the import statement

2. Import Hook

FireDucks provides command line option to automatically replace “**pandas**” with “**fireducks.pandas**”

```
$ python -m fireducks.pandas program.py
```

zero code modification

```
import mod_A
import mod_B
import mod_C
import pandas as pd
:
```

program.py

```
import pandas as pd
:
```

mod_A.py

```
import pandas as pd
:
```

mod_B.py

```
import pandas as pd
:
```

mod_C.py

3. Notebook Extension

FireDucks provides simple import extension for interactive notebooks.

```
%load_ext fireducks.pandas
import pandas as pd
```

simple integration in a notebook

Demo

```
pd.read_csv("data.csv").rolling(60).mean()["Close"].tail(1000).plot()
```

pandas the difference is only in the import **FireDucks**

The image shows two JupyterLab notebooks side-by-side. The left notebook, titled 'demo1p', uses the standard pandas import: `import pandas as pd`. The right notebook, titled 'demo1f', uses the FireDucks wrapper: `import fireducks.pandas as pd`. Both notebooks execute the same code: `pd.read_csv("data.csv").rolling(60).mean()["Close"].tail(1000).plot()`. The left notebook's execution time is 4.06s, while the right notebook's is 275ms. Both notebooks display a line plot of Bitcoin historical data. A red arrow points to the 'Run' button in the left notebook. A callout box highlights the import statement in each notebook.

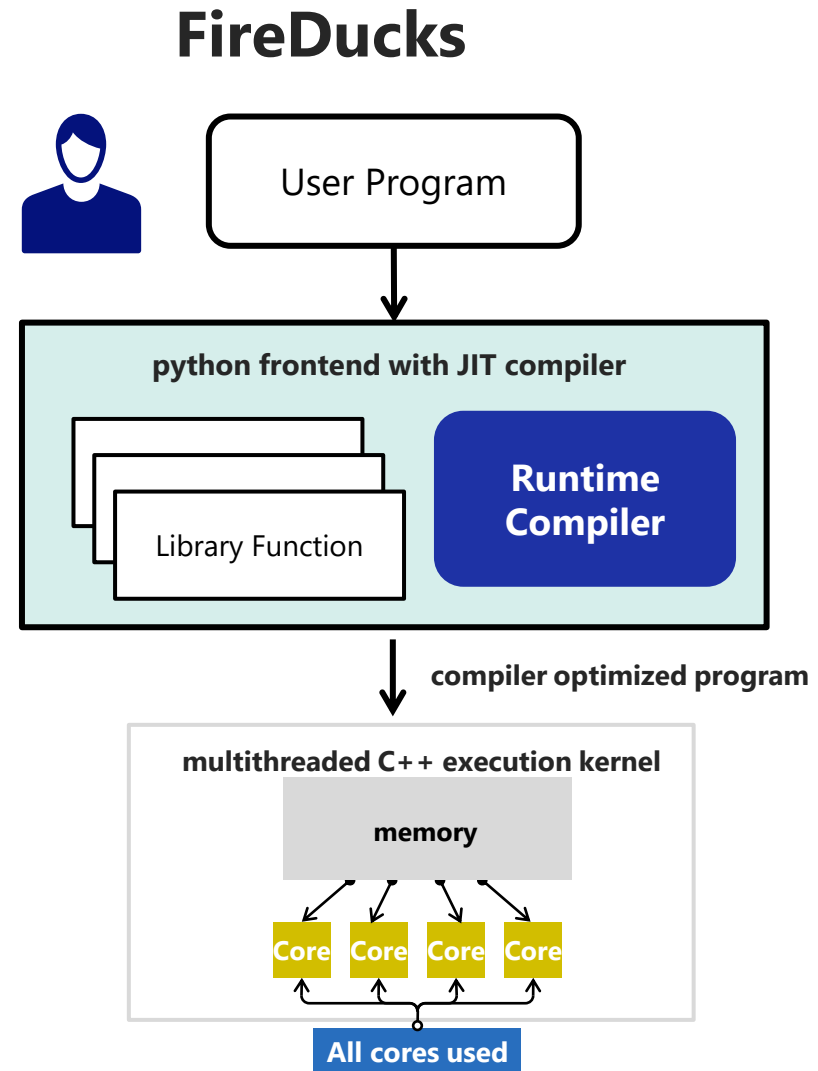
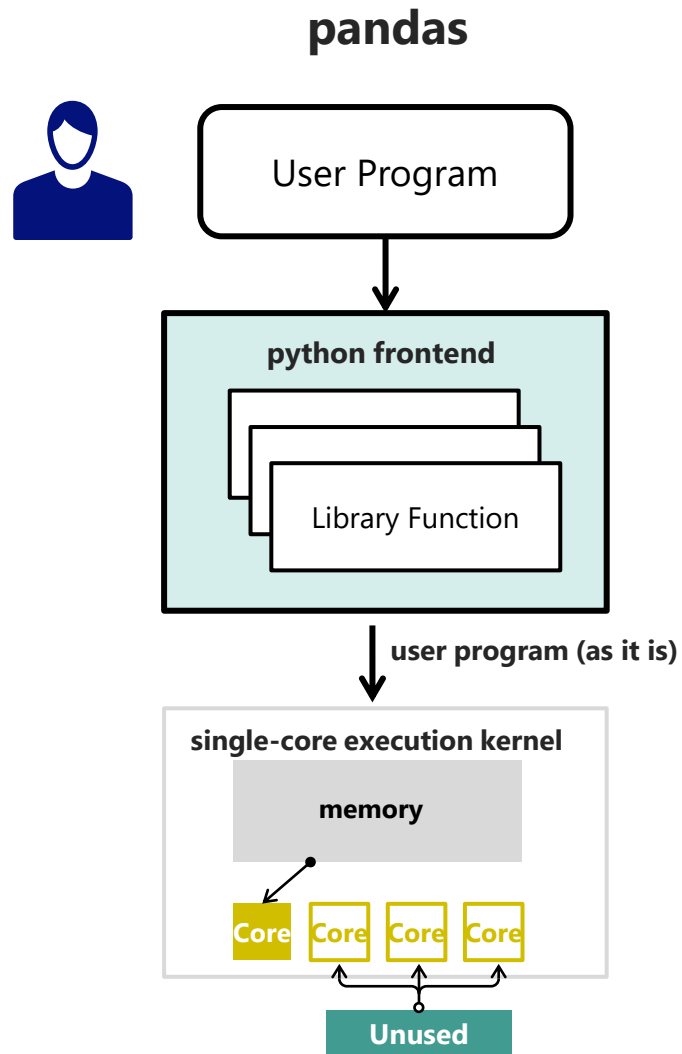
Program to calculate moving average

button to start execution

pandas: 4.06s
↓ ~15x
FireDucks: 275ms

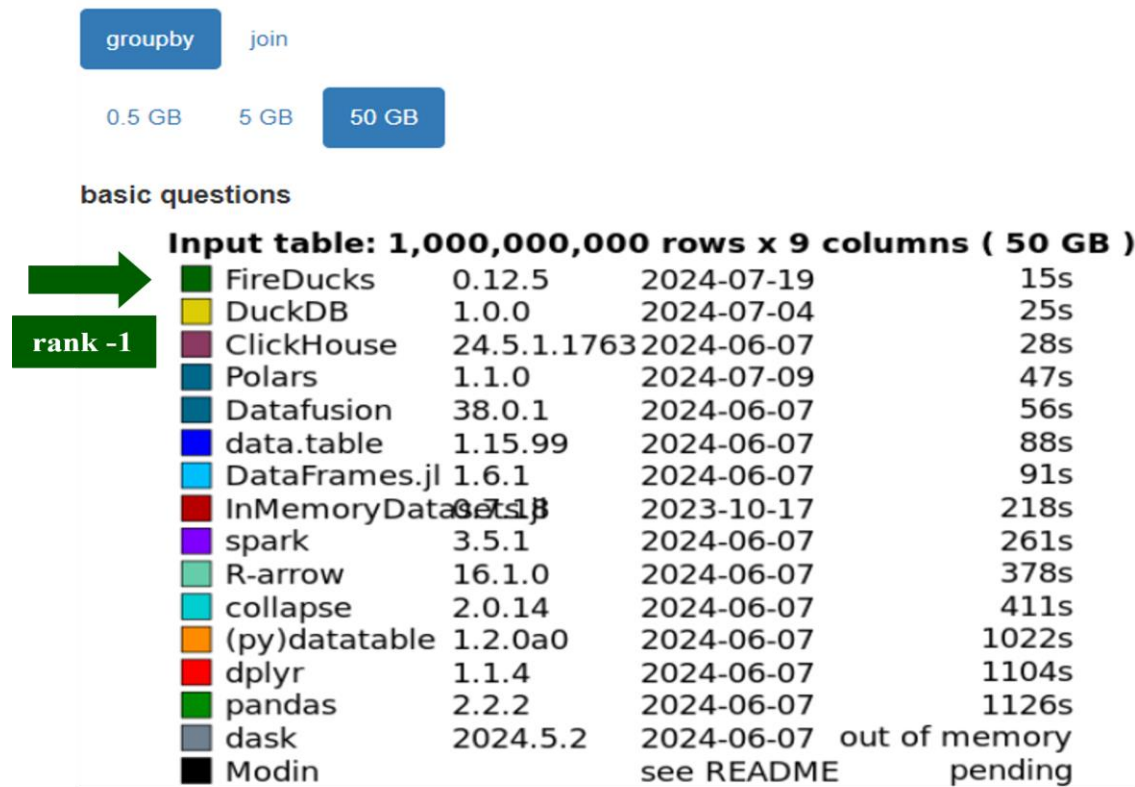
data.csv:
[Bitcoin Historical Data](#)

Why FireDucks is faster?

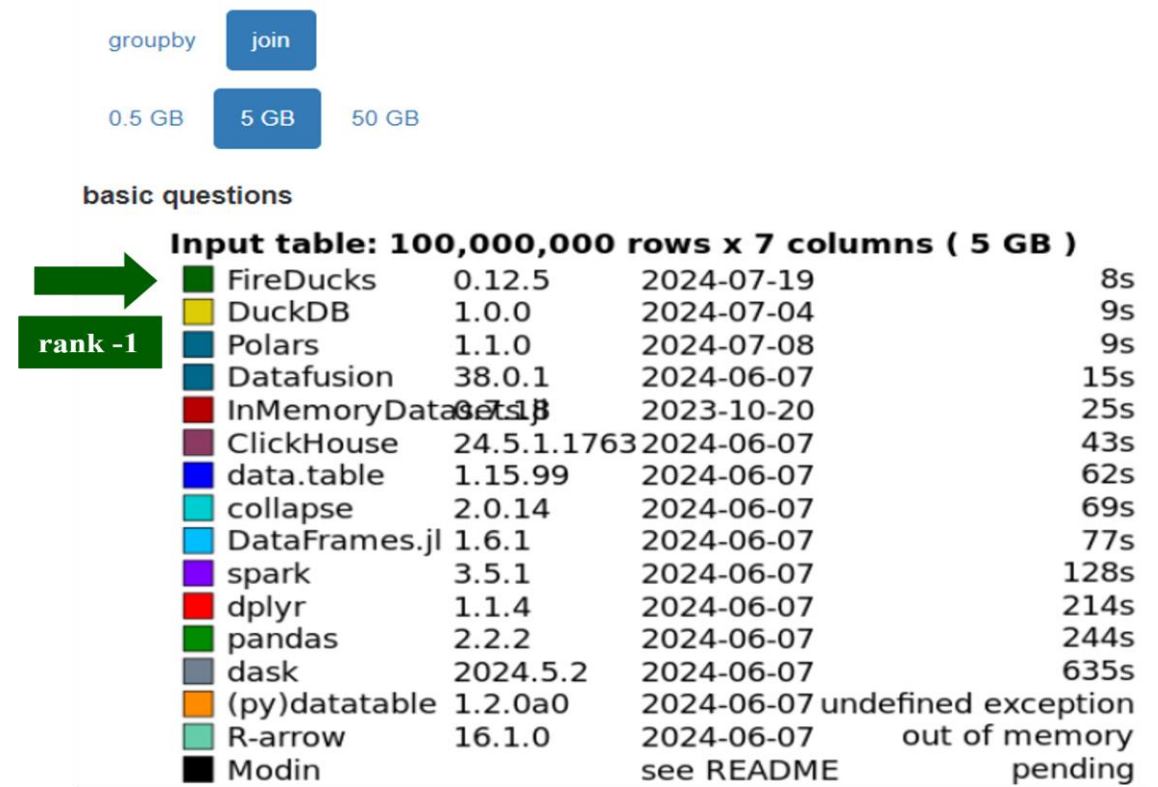


Benchmark (1): DB-Benchmark

Database-like ops benchmark (<https://duckdblabs.github.io/db-benchmark>)



groupby



join

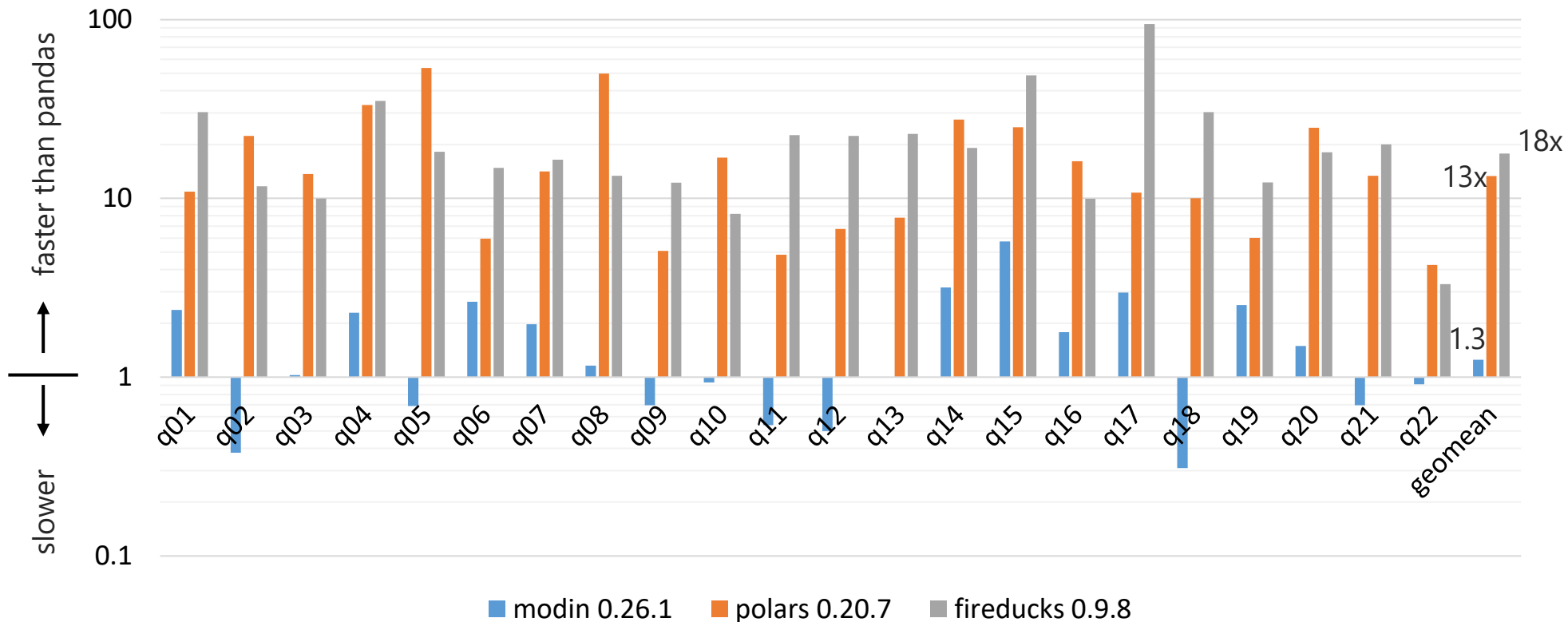
Benchmark (2): Speedup from pandas in TPC-H benchmark

FireDucks is 95x faster than pandas at max

Server

Xeon Gold 5317 x2
(24 cores), 256GB

Speedup from pandas 2.2.0 (Scale Factor=10)



Comparison of DataFrame libraries (average speedup)

FireDucks 18x

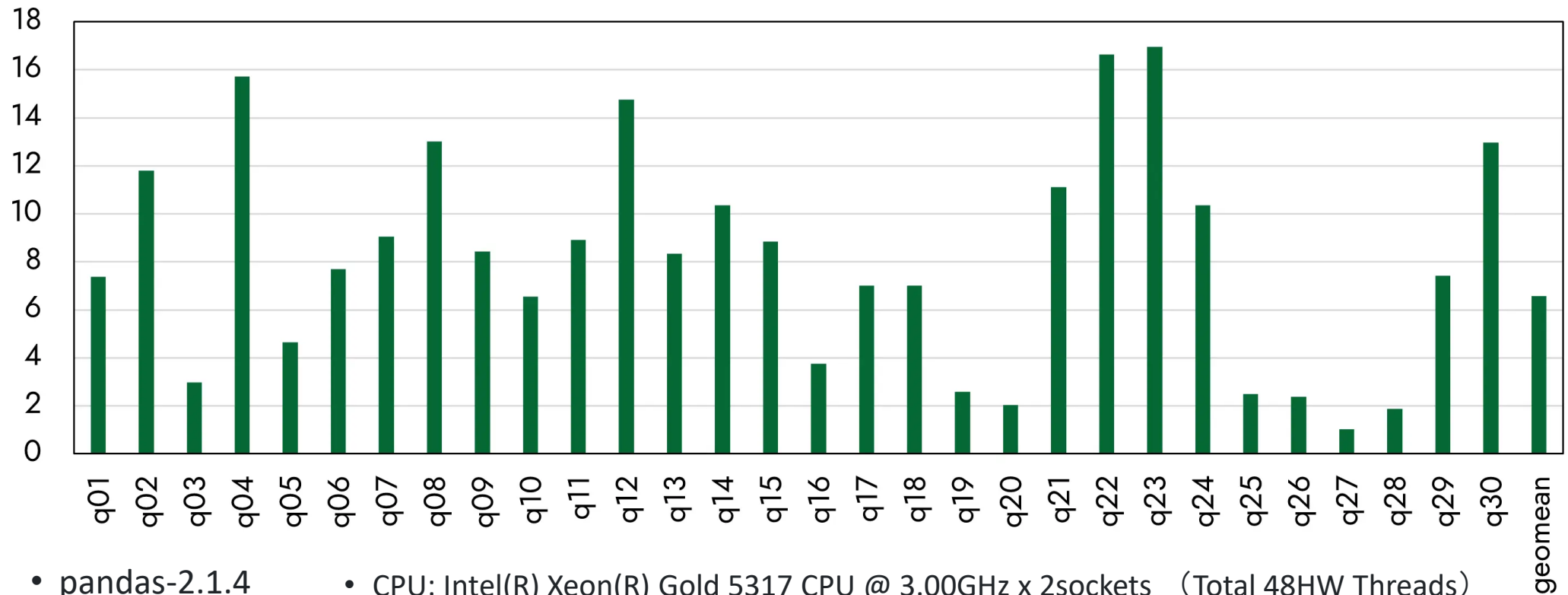
Polars 13x

Modin 1.3x

Benchmark (3): Speedup from pandas in TPCx-BB benchmark

ETL(Extract, Transform, Load) and ML Workflow

FireDucks speedup from pandas



- pandas-2.1.4
- fireducks-0.9.3

- CPU: Intel(R) Xeon(R) Gold 5317 CPU @ 3.00GHz x 2sockets (Total 48HW Threads)
- Main memory: 256GB

Resource on FireDucks

Web site (User guide, benchmark, blog)

<https://fireducks-dev.github.io/>



X(twitter) (Release information)

<https://x.com/fireducksdev>



Github (Issue report)

<https://github.com/fireducks-dev/fireducks>



slack Q/A, communication

https://join.slack.com/t/fireducks/shared_invite/zt-2j4lucmtj-IGR7AWIXO62Lu605pnBJ2w



FireDucks

Compiler Accelerated DataFrame Library for Python with fully-compatible pandas API

Get Started

```
import fireducks.pandas as pd
```

News

[Release fireducks-0.12.4 \(Jul 09, 2024\)](#)

[Have you ever thought of speeding up your data analysis in pandas with a compiler?\(blog\) \(Jul 03, 2024\)](#)

[Evaluation result of Database-like ops benchmark with FireDucks is now available. \(Jun 18, 2024\)](#)



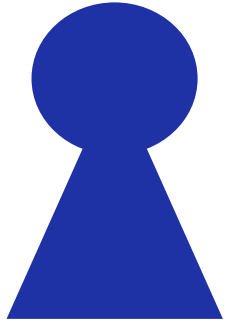
Accelerate pandas without any manual code changes

Do you have a pandas-based program that is slow? FireDucks can speed-up your programs without any manual code changes. You can accelerate your data analysis without worrying about slow performance due to single-threaded execution in pandas.

Comparison Among Several Python DataFrame Libraries

Library Name	pandas compatibility	single-node performance	multi-mode performance
FireDucks	○	○	×
Polars	×	○	×
Modin	○	△	○
Dask/Vaex	△	△	○
Pandas	○	×	×

User feedback



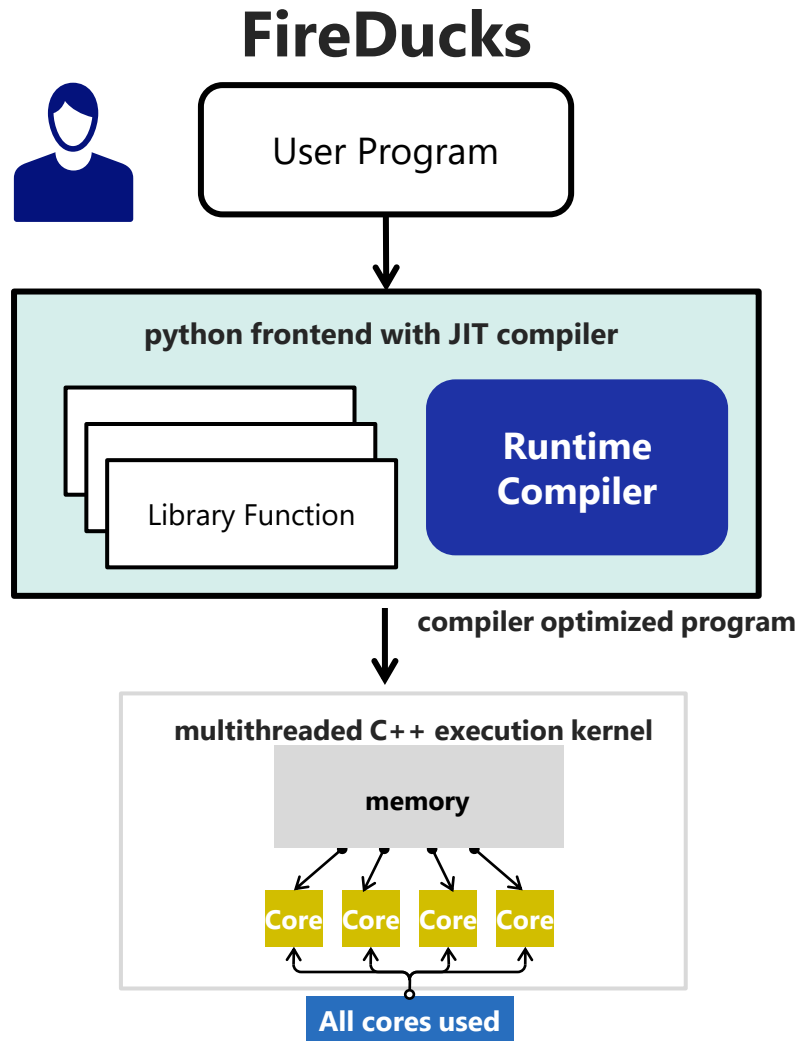
Due to a significant reduction in execution time, I can now focus more on in-depth data analysis.



Easy integration in an existing application in just 30 mins!

Further Technical Details

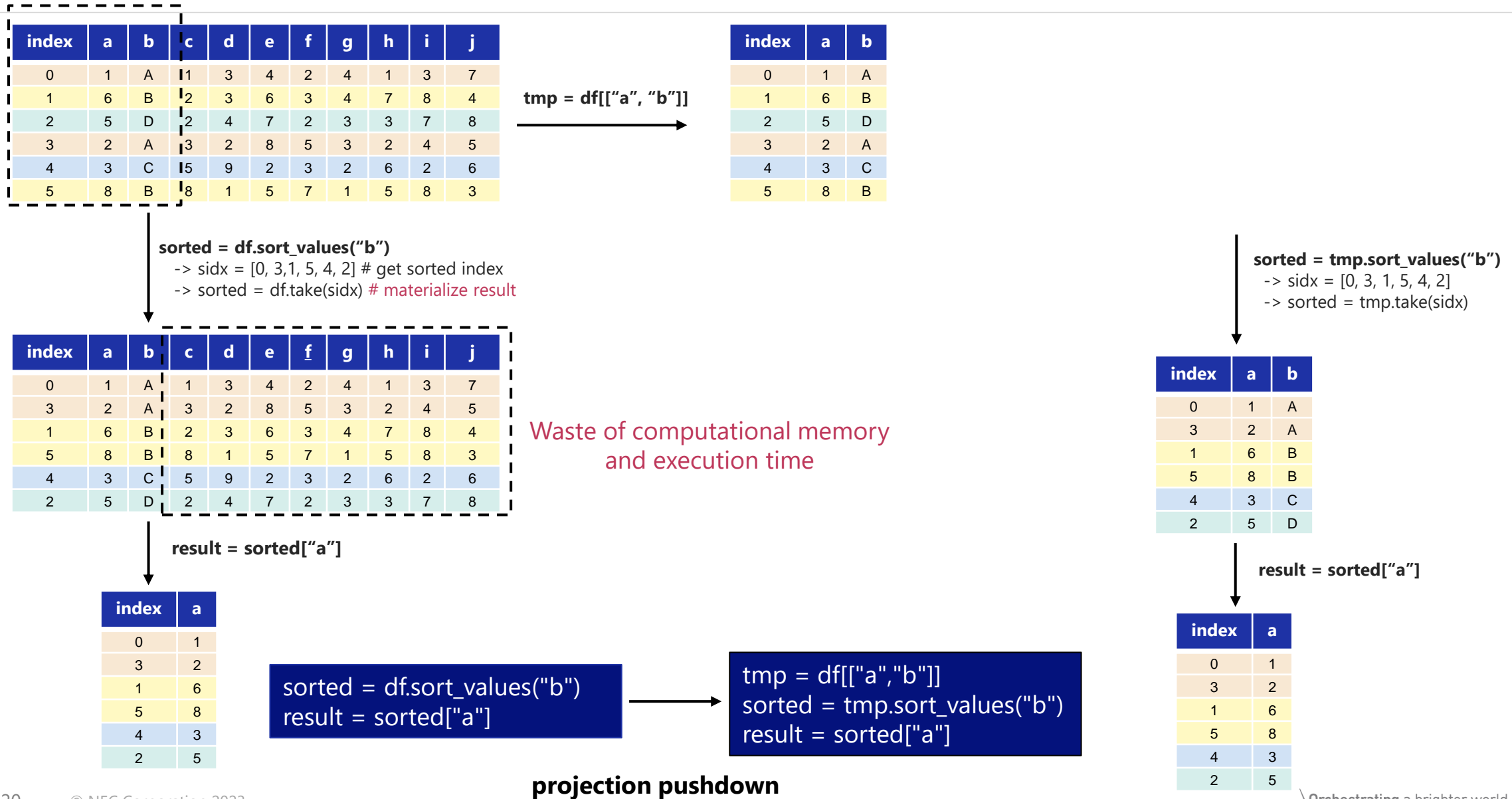
Optimization Features



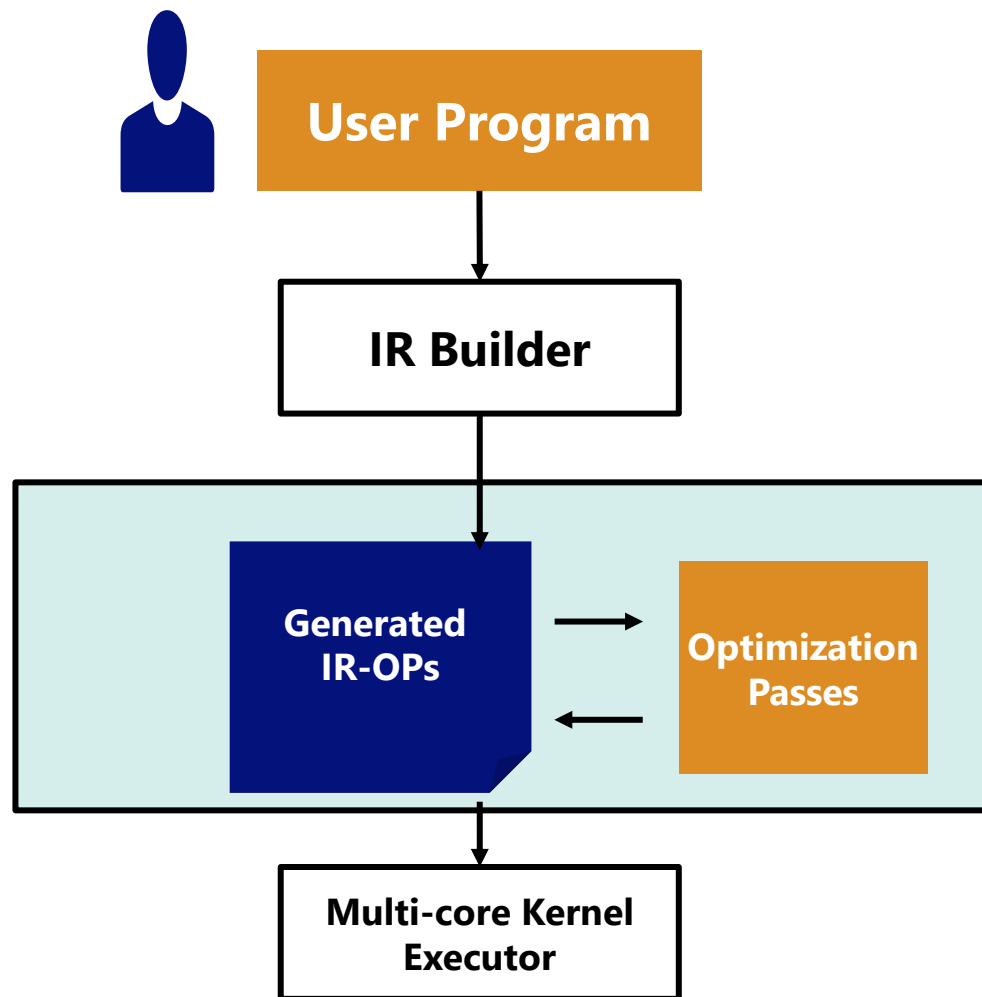
1. **Compiler Specific Optimizations:** Common Sub-expression Elimination, Dead-code Elimination, Constant Folding etc.
2. **Domain Specific Optimization:** Optimization at query-level: reordering instructions etc.
3. **Pandas Specific Optimization:** selection of suitable pandas APIs, selection of suitable parameter etc.

1. **Multi-threaded Computation:** Leverage all the available computational cores.
2. **Efficient Memory Management:** Data Structures backed by Apache Arrow
3. **Optimized Kernels:** Patented algorithms for Database like kernel operations: like sorting, join, filter, groupby, dropna etc. developed in C++ from scratch.

Domain Specific Optimization (Example #1)



How does FireDucks Work?



```
sorted = df.sort_values("b")  
result = sorted["a"]
```

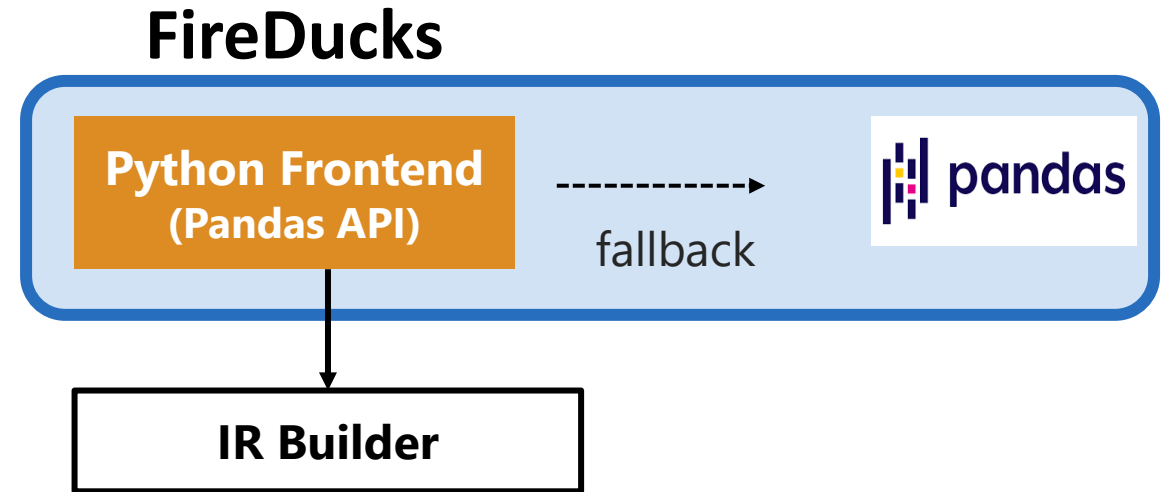
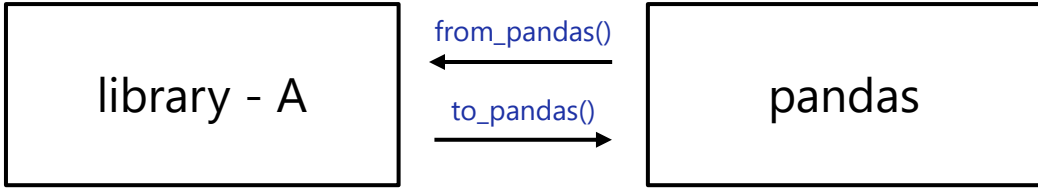
```
%v2 = "fireducks.sort_values"(%v1,"b")  
%v3 = "fireducks.project"(%v2,["a"])
```

print (result)

```
%v11 = "fireducks.project"(%v1,["a","b"])  
%v2 = "fireducks.sort_values"(%v11,"b")  
%v3 = "fireducks.project"(%v2,["a"])
```

```
tmp = df[["a","b"]]  
sorted = tmp.sort_values("b")  
result = sorted["a"]
```

Why FireDucks is highly compatible with pandas?



```
%load_ext fireducks.pandas ← notebook extension for importhook
import pandas as pd
import numpy as np
```

```
%%fireducks.profile ← notebook specific profiler
df = pd.DataFrame({
    "id": np.random.choice(list("abcdef"), 10000),
    "val": np.random.choice(100, 10000)
})

r1 = (
    df.sort_values("id")
    .groupby("id")
    .head(2)
    .reset_index(drop=True)
)


r1["val"] = r1["val"].cumsum()
r1.describe()
```

profiling-summary:: total: 42.4832 msec (fallback: 1.1448 msec)


	name	type	n_calls	duration (msec)
0	groupby_head	kernel	1	16.696805
1	sort_values	kernel	1	16.684564
2	from_pandas.frame.metadata	kernel	2	3.641694
3	to_pandas.frame.metadata	kernel	2	2.237987
4	describe	kernel	1	2.021135
5	DataFrame_repr_html_	fallback	1	1.021662
6	Series.cumsum	fallback	1	0.111802
7	setitem	kernel	1	0.010280
8	get_metadata	kernel	1	0.009650
9	reset_index	kernel	1	0.008050

When running a python script/program, you may like to set the environment variable to get fallback warning logs:
FIREDUCKS_FLAGS="-Wfallback"

Raise feature request when you encounter some expensive fallback hindering your program performance!



Directly **communicate** with us over our slack channel for any performance or API related queries!



Demo

<https://colab.research.google.com/drive/1qpej-X7CZsleOqKuhBg4kq-cbGuJf1Zp?usp=sharing>



\Orchestrating a brighter world

NEC creates the social values of safety, security, fairness and efficiency to promote a more sustainable world where everyone has the chance to reach their full potential.

\ Orchestrating a brighter world

NEC