Experimentação em Engenharia de Software

On the Performance of the Python Language

PG 54232
PG 55972
PG 57539

Content

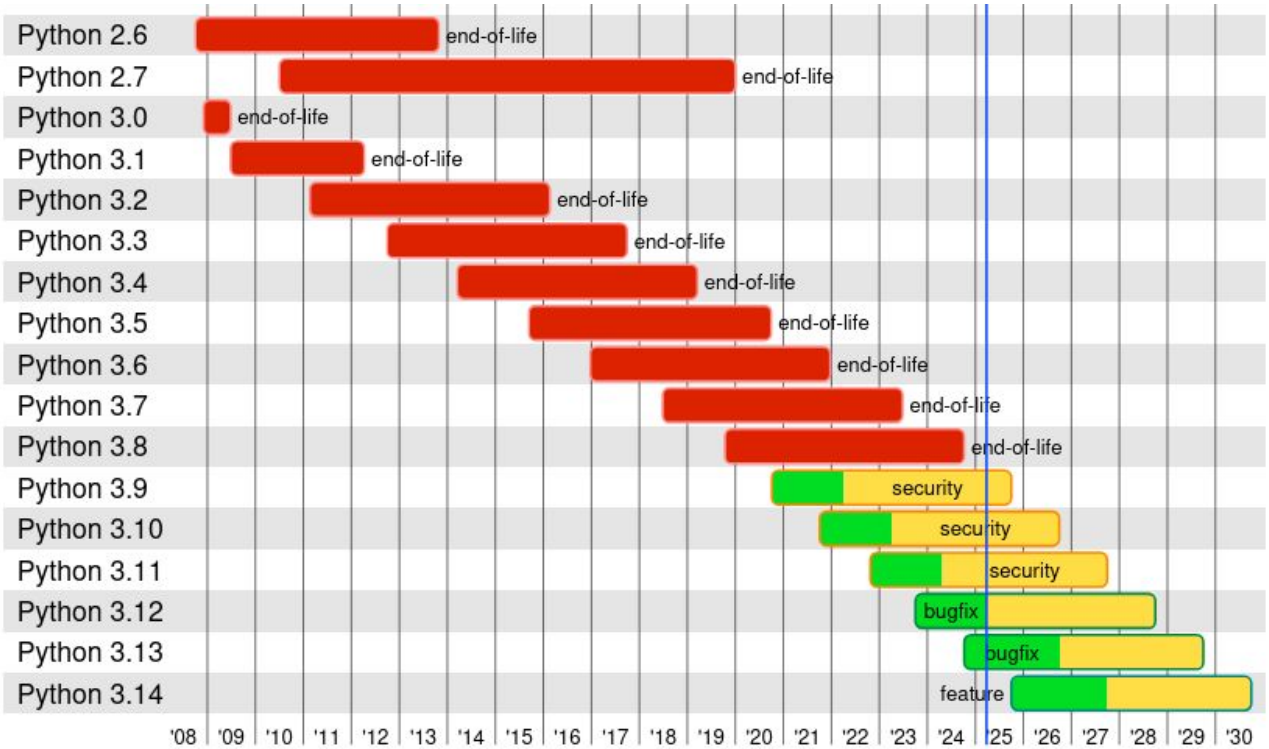
- Python
  - Lifespan
  - Chosen versions
- Methodology
- Environment
- Test Suitcase
  - Fibonacci
  - Data querying
- Authors Notes
  - Threats to validity
  - Future work
  - Conclusion

Python

Python – Lifespan

https://devguide.python.org/versions/#versions

https://endoflife.date/python



Python – Chosen Versions

Version	Release Date	Maintenance status
3.13.2	Feb. 4, 2025	Bugfix
3.13.1	Dec. 3, 2024	Bugfix
3.12.9	Feb. 4, 2025	Bugfix
3.12.8	Dec. 3, 2024	Bugfix
3.12.3	April 9, 2024	Bugfix
3.11.11	Dec. 3, 2024	Security
3.10.16	Dec. 3, 2024	Security
3.9.21	Dec. 3, 2024	Security
2.0.1	June 22, 2001	End-of-life

https://www.python.org/downloads/

Python – 2.0.x

Release: October 16, 2000

New features: Regular expressions w/ **re** module
List Comprehensions
String Methods

Curiosities:
No Boolean data type
No **datetime** module!

Python – 2.0.x :: List Comprehensions

Traditional Loop:

```
squares = [ ]

for x in range(10):
    squares.append(2**x)

#  [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

List Comprehension:

```
squares = [2**x for x in range(10)]

#  [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

Inspired by Haskell

Provide a concise and often more readable way to create lists.

Combine a for loop, a optional if condition and the list creation into a single line.

Python – 2.7

Release: June 22, 2001

New features:

Ordered dictionaries

Imported from 3.1:

io module rewritten in C

Python – 2.7 :: Order dictionaries

```
>>> from collections import OrderedDict
>>> d = OrderedDict( [
…              ('A', 1) ,
…              ('C', 3) ,
...              ('B', 2)
…     ] )

>>> d.items()
[('A', 1), ('B', 2), ('C', 3)]
```

Python 2.7 introduces a new **OrderedDict** class in the **collections** module.

Python – 3.9.x

Release: Dec. 3, 2024

Features:

Dictionary Merge
Update Operator

Curiosities:

Last version of python 3 with full backward 2.x compatibility

Python – 3.9.x :: Dictionary Merge & Update Operators

```
>>> x = {"key1": "A", "key2": "B"}
>>> y = {"key2": "100", "key3": "200"}

>>> y | x
>>> y
{
        'key2': '100',
        'key3': '200',
        'key1': 'A'
}

>>> y |= x
>>> y
 {
         'key2': 'B',
         'key3': '200',
         'key1': 'A'
}
```

Merge (|) and update (|=) operators have been added to the built-in dict class.

Python – 3.10.16

Release: Dec. 3, 2024

New features:

Better error messages

Python – 3.10.16 :: Better error messages

```
expected = {9: 1, 18: 2, 19: 2, 27: 3, 28: 3, 29: 3, 36: 4, 37: 4,
                38: 4, 39: 4, 45: 5, 46: 5, 47: 5, 48: 5, 49: 5, 54: 6,
some_other_code = foo()
```

Previous:

```
File "example.py", line 3
    some_other_code = foo()
                      ^
SyntaxError: invalid syntax
```

Current:

```
File "example.py", line 1
expected = {9: 1, 18: 2, 19: 2, 27: 3, 28: 3, 29: 3, 36: 4, 37: 4,
           ^
SyntaxError: '{' was never closed
```

Python – 3.12.9

Release: Feb. 4, 2025

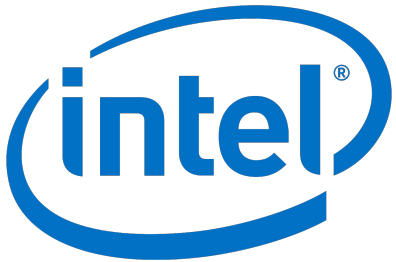
New features:

Thread locking mechanism – GIL

Methodology

Benchmarking Methodology

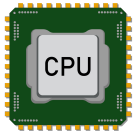
- Run 10x *via* RAPL
  - Delete min and max outliers
    - Mean



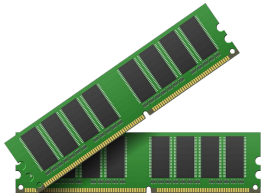
Environment

Inspects

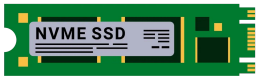
- CPU
  - Intel(R) Core(TM) i3-9100F CPU @ 3.60GHz
    - 4 Cores - 4 Threads



- RAM
  - DDR4 @ 2400 MHz
    - 8GB Dual Band (16 GB)



- ROM
  - M.2 2280 NVMe SSD
    - R 2,400 MB/s



- OS
  - Ubuntu 24.04.2 LTS
    - Linux 6.11.0-19-generic x86_64

C Language Compiler

- GCC
  - V 13.3.0

RAPL

Core Temperatures

Core 0: 25.0°C

Core 1: 25.0°C

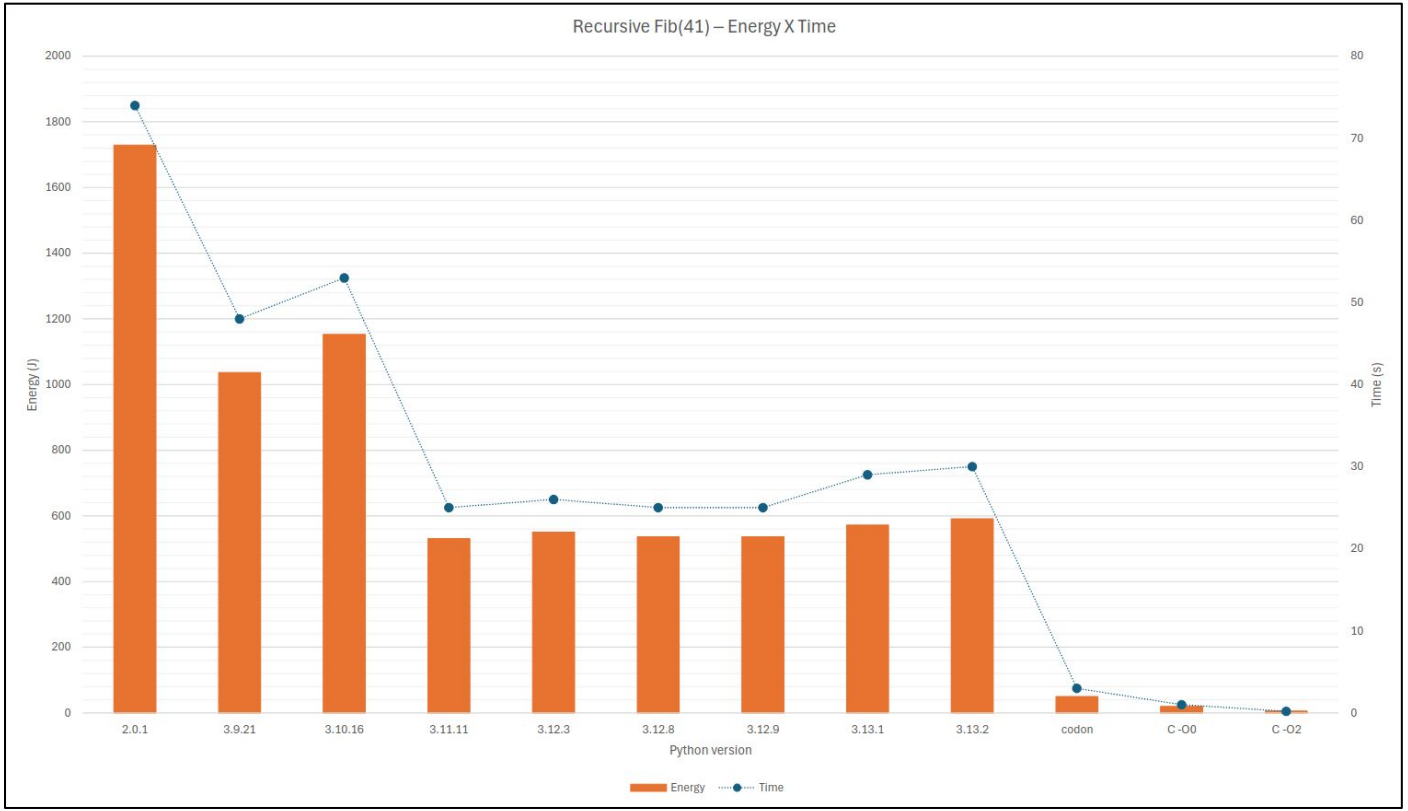
Core 2: 25.0°C

Core 3: 25.0°C

Average Temperature: 25.0°C

Fibonacci Recursive

Fibonacci Recursive (41) – Energy X Time



Fibonacci Recursive (41) – Energy X Time

Version	Energy (J)	Time (s)
2.0.1	1 729.51	74.1
3.9.21	1 036.47	48.6
3.10.16	1 153.91	53.8
3.11.11	531.16	25.5
3.12.3	550.93	26.3
3.12.8	536.45	25.2
3.12.9	536.49	25.3
3.13.1	572.52	29.1
3.13.2	591.31	29.4
codon	50.4	2.4
C -O0	20.79	1.1
C -O2	5.58	0.3

Fibonacci Recursive (41) – Powerup X Speedup X Greenup

Version	Powerup	Speedup	Greenup
2.0.1	1	1	1
3.9.21	0.9	1.5	1.7
3.10.16	0.9	1.4	1.5
3.11.11	0.9	2.9	3.3
3.12.3	0.9	2.8	3.1
3.12.8	0.9	2.9	3.2
3.12.9	0.9	2.9	3.2
3.13.1	0.8	2.6	3.0
3.13.2	0.9	2.5	2.9
codon	0.9	30.6	34.3
C -O0	0.8	63.4	83.2
C -O2	0.8	248.9	309.9

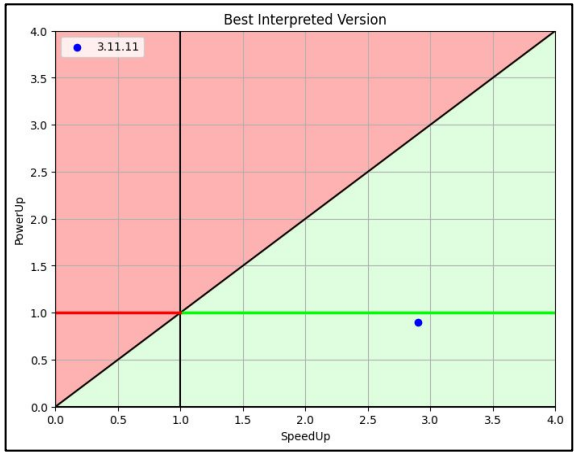
Fibonacci Recursive (41) – Powerup X Speedup X Greenup

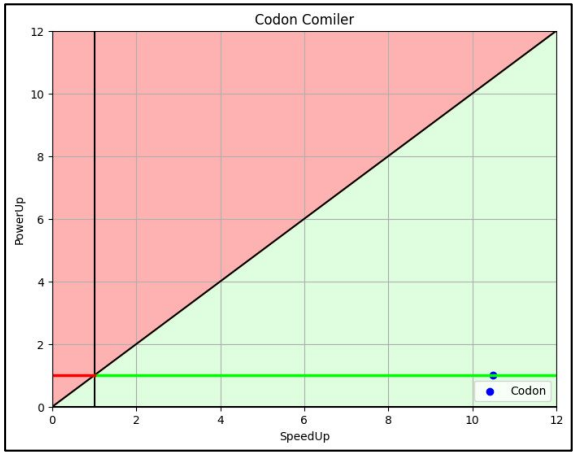
Version	Powerup	Speedup	Greenup
2.0.1	1	1	1
3.9.21	0.9	1.5	1.7
3.10.16	0.9	1.4	1.5
3.11.11	0.9	2.9	3.3
3.12.3	0.9	2.8	3.1
3.12.8	0.9	2.9	3.2
3.12.9	0.9	2.9	3.2
3.13.1	0.8	2.6	3.0
3.13.2	0.9	2.5	2.9
codon	0.9	30.6	34.3
C -O0	0.8	63.4	83.2
C -O2	0.8	248.9	309.9

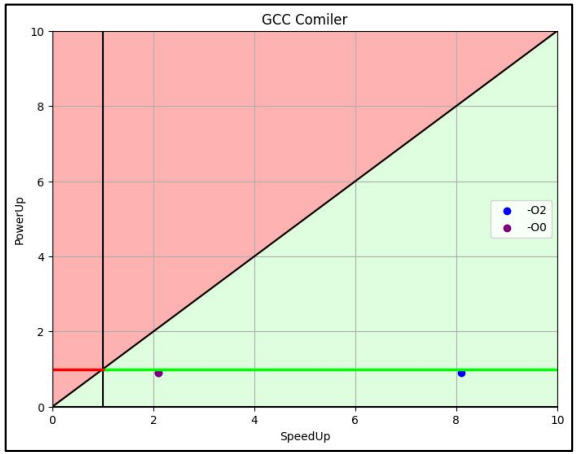
Version	Powerup	Speedup	Greenup
3.11.11	1	1	1
codon	1.0	10.5	10.5
C -O0	0.9	21.8	25.5
C -O2	0.9	85.7	95.2

Version	Powerup	Speedup	Greenup
codon	1	1	1
C -O0	0.9	2.1	2.4
C -O2	0.9	8.1	9.0

Fibonacci Recursive (41) – GPS







Querying

Querying – Data

USER		
Field	**Type**	**Constraint**
username	VARCHAR(75)	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
pay_method	VARCHAR(16)	ENUM
account_status	VARCHAR(8)	ENUM

RIDE		
Field	**Type**	**Constraint**
id	VARCHAR(75)	PK
date	DATE	–
driver	INTEGER	FK
user	VARCHAR(75)	FK
city	VARCHAR(64)	–
distance	INTEGER	–
score_user	FLOAT	–
score_driver	FLOAT	–
tip	FLOAT	–
comment	TEXT	–

DRIVER		
Field	**Type**	**Constraint**
id	INTEGER	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
account_status	VARCHAR(8)	ENUM
car_class	VARCHAR(8)	ENUM
license_plate	VARCHAR(8)	–
city	VARCHAR(64)	–

Querying – Data

Relationships

USER		
Field	**Type**	**Constraint**
username	VARCHAR(75)	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
pay_method	VARCHAR(16)	ENUM
account_status	VARCHAR(8)	ENUM

RIDE		
Field	**Type**	**Constraint**
id	VARCHAR(75)	PK
date	DATE	–
driver	INTEGER	FK
user	VARCHAR(75)	FK
city	VARCHAR(64)	–
distance	INTEGER	–
score_user	FLOAT	–
score_driver	FLOAT	–
tip	FLOAT	–
comment	TEXT	–

DRIVER		
Field	**Type**	**Constraint**
id	INTEGER	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
account_status	VARCHAR(8)	ENUM
car_class	VARCHAR(8)	ENUM
license_plate	VARCHAR(8)	–
city	VARCHAR(64)	–

Querying – Process

Load	Validation	Structure	Query
1.38 GB	1 000 001 + 100 001 + 10 000 001 = 11 100 003	Index by **‘username’** Index by **‘driver_id’** Index by *fit*	Access structures Calculate

Querying – Process

Load → Validation → Structure → Query

1.38 GB

1 000 001
+
100 001
+
10 000 001

=
11 100 003

Index by **‘username’**

Index by **‘driver_id’**

Index by *fit*

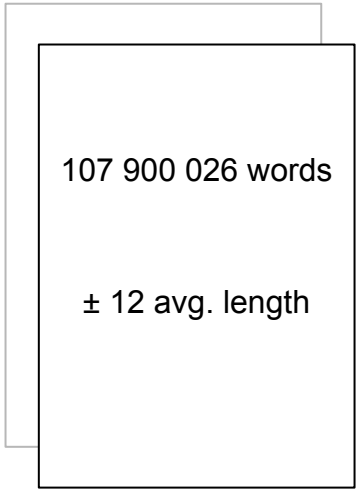
Access structures

Calculate

Querying – Process

Load | Validation | Structure | Query

1.38 GB



Index by **‘username’**

Index by **‘driver_id’**

Index by *fit*

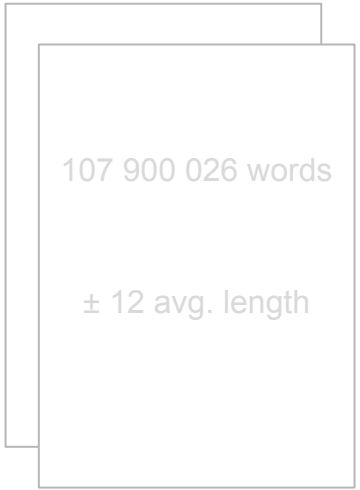
Access structures

Calculate

Querying – Process

Load → Validation → **Structure** → Query

1.38 GB



Index by **‘username’**

Index by **‘driver_id’**

Index by *fit*

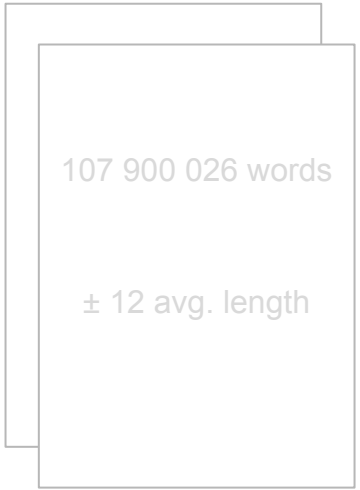
Access structures

Calculate

Querying – Process

Load → Validation → Structure → **Query**

1.38 GB



Index by **‘username’**

Index by **‘driver_id’**

Index by *fit*

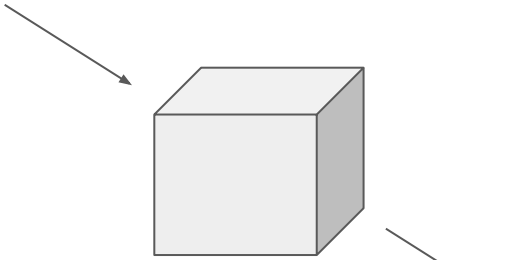
Access structures

Calculate

User Profiling

Querying – User profiling

username



- ★ Nome
- ★ Género
- ★ Idade
- ★ Avaliação média
- ★ Número de viagens
- ★ Total gasto €

Querying – User profiling

Structure

USER

Field	Type	Constraint
username	VARCHAR(75)	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
pay_method	VARCHAR(16)	ENUM
account_status	VARCHAR(8)	ENUM

Ride Index

user

RIDE

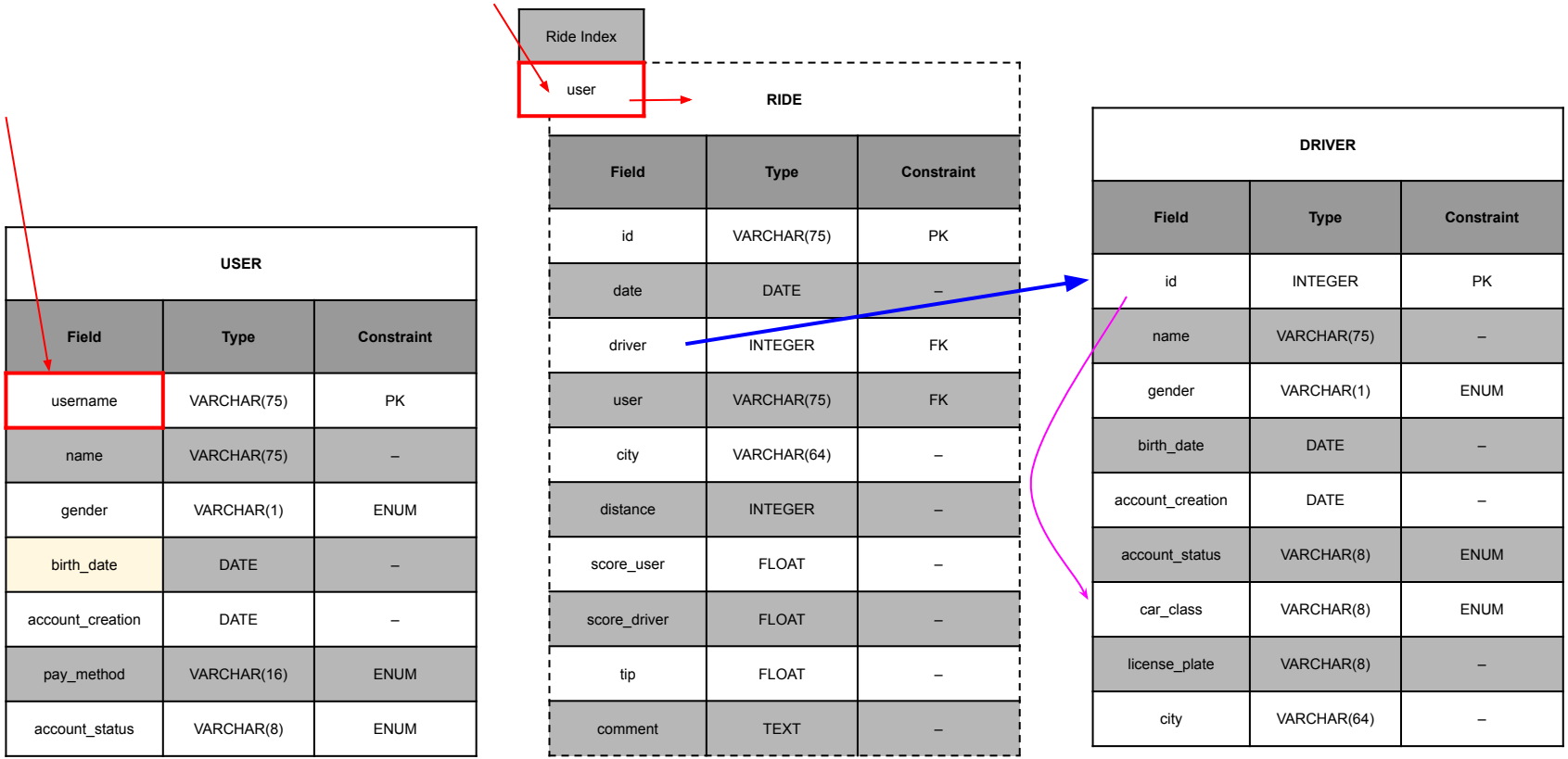
Field	Type	Constraint
id	VARCHAR(75)	PK
date	DATE	–
driver	INTEGER	FK
user	VARCHAR(75)	FK
city	VARCHAR(64)	–
distance	INTEGER	–
score_user	FLOAT	–
score_driver	FLOAT	–
tip	FLOAT	–
comment	TEXT	–

DRIVER

Field	Type	Constraint
id	INTEGER	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
account_status	VARCHAR(8)	ENUM
car_class	VARCHAR(8)	ENUM
license_plate	VARCHAR(8)	–
city	VARCHAR(64)	–

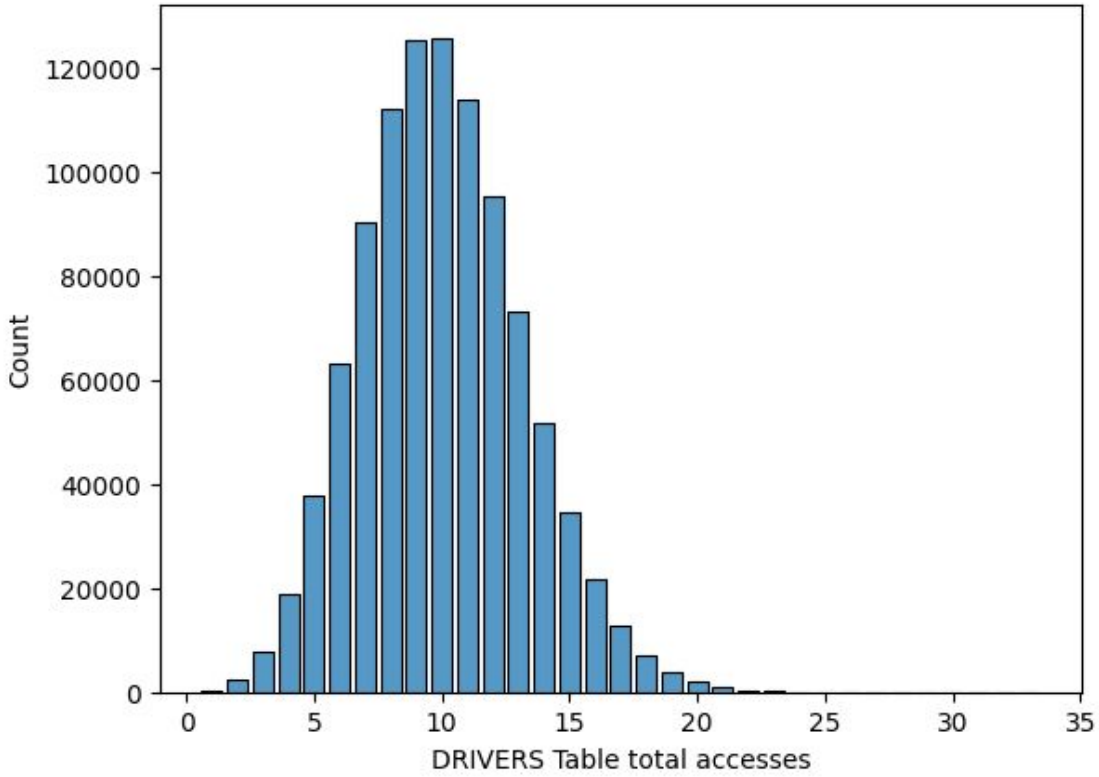
Querying – User profiling

Relating



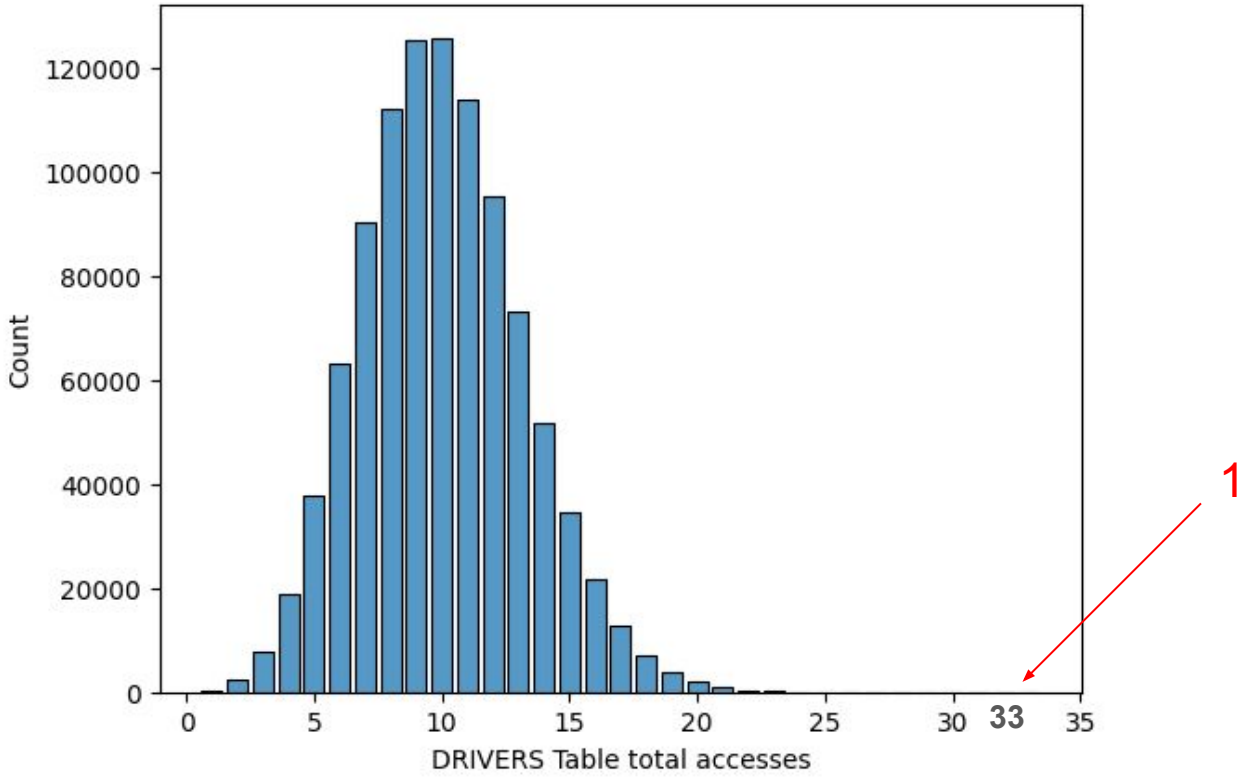
Querying – User profiling

Benchmark representativeness



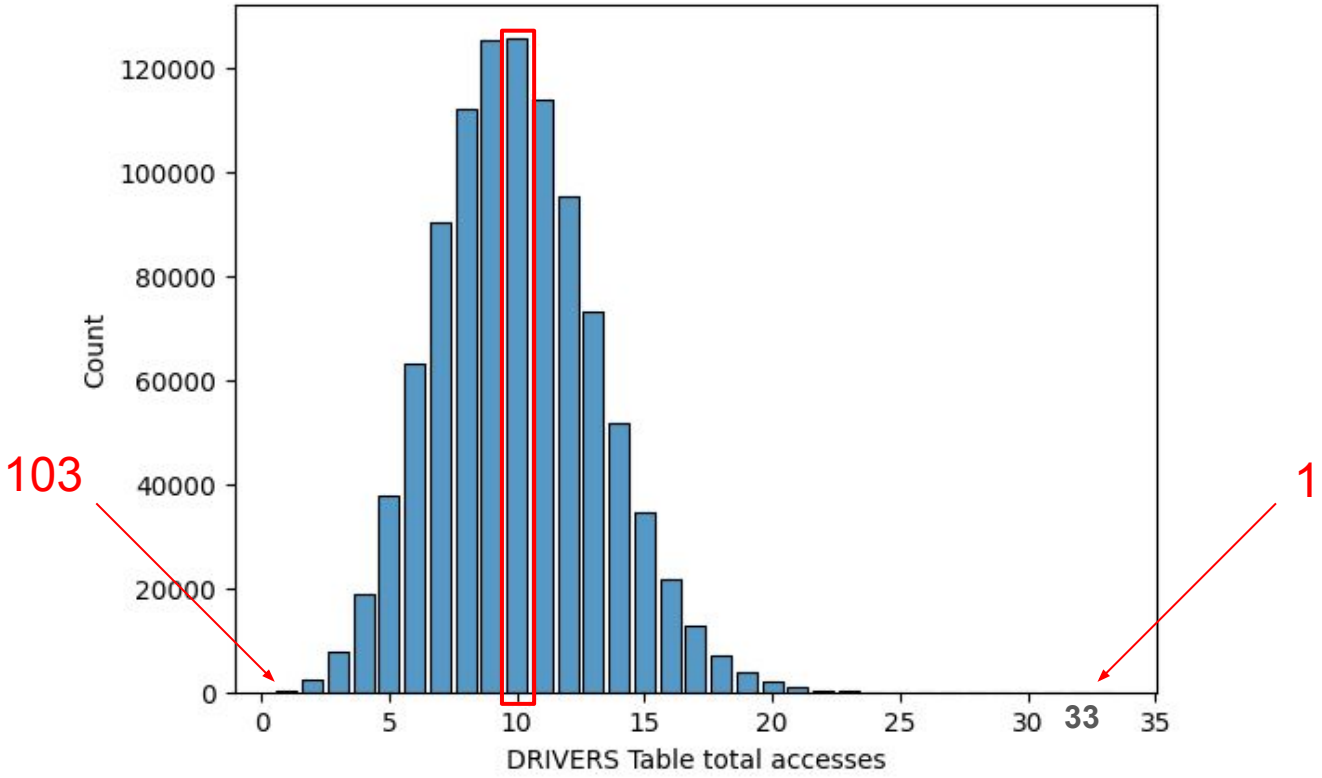
Querying – User profiling

Benchmark representativeness



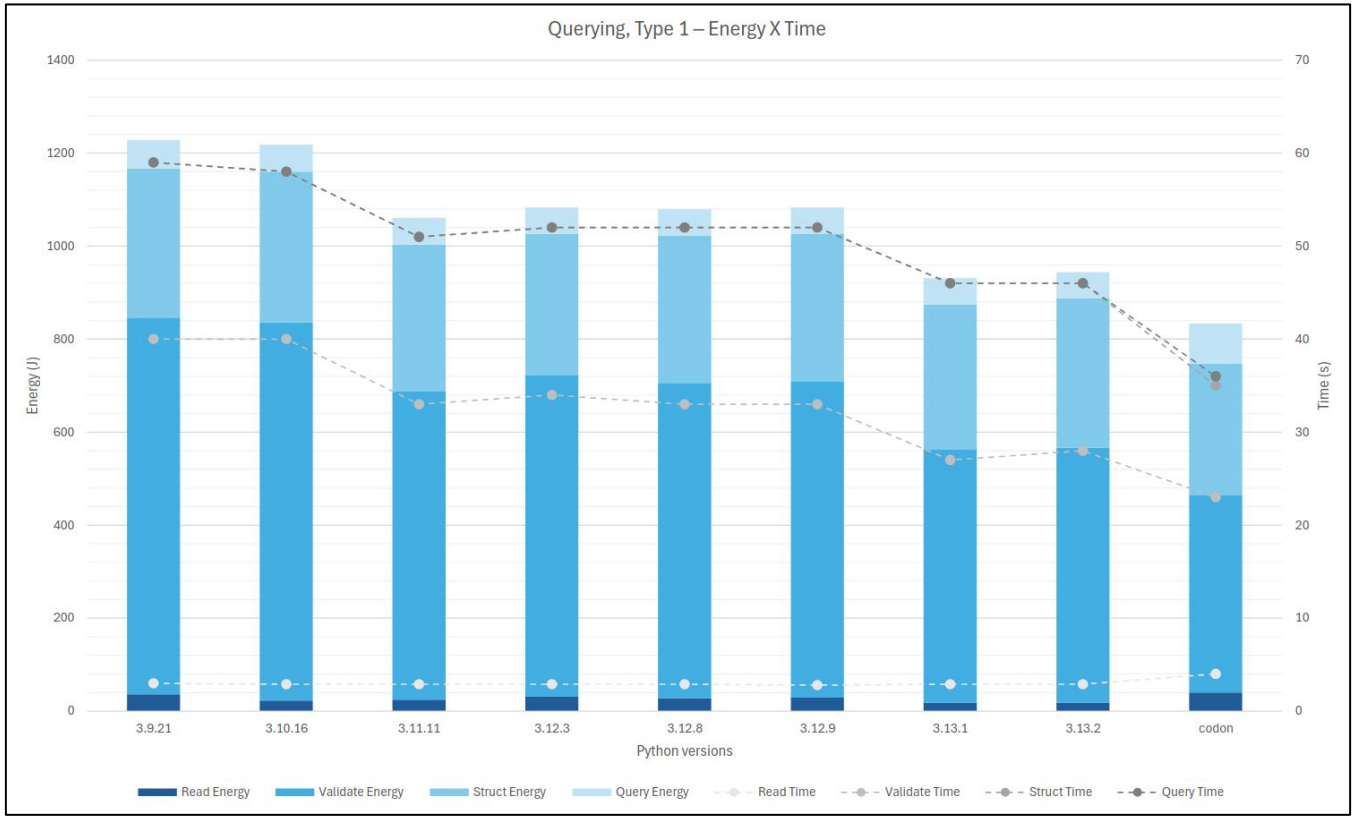
Querying – User profiling

Benchmark representativeness



Querying – Process

Energy X Time



Querying

Energy X Time

Load → Validation → Structure → **Query**

1 access

Version	Energy (J)	Time (s)
3.9.21	1 228.51	59.4
3.10.16	1 218.98	58.6
3.11.11	1 061.06	51.5
3.12.3	1 083.83	52.5
3.12.8	1 079.52	52.4
3.12.9	1 083.32	52.7
3.13.1	931.24	46.0
3.13.2	944.55	46.3
Codon	833.02	36.6

10 accesses

Version	Energy (J)	Time (s)
3.9.21	1 196.07	59.5
3.10.16	1 202.63	59.3
3.11.11	1 040.57	51.8
3.12.3	1 062.77	52.9
3.12.8	1 049.75	52.5
3.12.9	1 061.43	52.9
3.13.1	918.22	46.3
3.13.2	926.41	46.4
Codon	832.95	36.6

33 accesses

Version	Energy (J)	Time (s)
3.9.21	1 224.36	59.4
3.10.16	1 224.84	59.1
3.11.11	1 066.10	51.6
3.12.3	1 083.56	52.6
3.12.8	1 081.10	52.6
3.12.9	1 086.38	52.8
3.13.1	932.25	46.1
3.13.2	941.65	46.4
Codon	835.38	36.6

Querying

Powerup X Speedup X Greenup

33 accesses

Version	Powerup	Speedup	Greenup
3.9.21	1	1	1
3.10.16	1.01	1.01	1.00
3.11.11	1.00	1.15	1.15
3.12.3	1.00	1.13	1.13
3.12.8	1.00	1.13	1.13
3.12.9	1.00	1.12	1.13
3.13.1	0.98	1.29	1.31
3.13.2	0.98	1.28	1.30
Codon	1.11	1.62	1.47

33 accesses

Version	Powerup	Speedup	Greenup
3.13.1	1	1	1
Codon	1.13	1.26	1.12

Drivers Score Rank

Querying – Driver score rank

Structure

RIDE

Field	Type	Constraint
id	VARCHAR(75)	PK
date	DATE	–
driver	INTEGER	FK
user	VARCHAR(75)	FK
city	VARCHAR(64)	–
distance	INTEGER	–
score_user	FLOAT	–
score_driver	FLOAT	–
tip	FLOAT	–
comment	TEXT	–

DRIVER

Field	Type	Constraint
id	INTEGER	PK
name	VARCHAR(75)	–
gender	VARCHAR(1)	ENUM
birth_date	DATE	–
account_creation	DATE	–
account_status	VARCHAR(8)	ENUM
car_class	VARCHAR(8)	ENUM
license_plate	VARCHAR(8)	–
city	VARCHAR(64)	–

Querying

Energy X Time

Load → Validation → Structure → **Query**

Querying

Energy X Time

Load → Validation → Structure → **Query**

Top 10

Version	Energy (J)	Time (s)
3.9.21	1 228	59.3
3.10.16	1 219.12	58.6
3.11.11	1 030.53	50.1
3.12.3	1 057.81	51.4
3.12.8	1 058.46	51.3
3.12.9	1 060.90	51.5
3.13.1	914.72	45.1
3.13.2	917.76	45.3
Codon	863.69	39.5

Top 100 000

Version	Energy (J)	Time (s)
3.9.21	1 273.43	59.2
3.10.16	1 272.59	58.9
3.11.11	1 063.56	50.1
3.12.3	1 106.79	51.7
3.12.8	1 095.12	51.3
3.12.9	1 103.01	51.5
3.13.1	945.25	45.0
3.13.2	955.16	45.4
Codon	865.74	39.6

Querying

Powerup X Speedup X Greenup

Load → Validation → Structure → **Query**

Top 100 000

Version	Powerup	Speedup	Greenup
3.9.21	1	1	1
3.10.16	1.00	1.01	1.00
3.11.11	0.99	1.18	1.20
3.12.3	1.00	1.15	1.15
3.12.8	0.99	1.15	1.16
3.12.9	1.00	1.15	1.15
3.13.1	0.98	1.32	1.35
3.13.2	0.98	1.30	1.33
Codon	1.02	1.49	1.47

Top 100 000

Version	Powerup	Speedup	Greenup
3.13.1	1	1	1
Codon	1.04	1.14	1.09

Authors Notes

Threats to Validity

- Representativeness

- Energy consumption as a whole

Future Work

- Benchmark sorting algorithms
- Benchmark data structures
- Develop data analysis

Conclusion

- Optimized interpreters

- Compiled
  - Best performance

Experimentação em Engenharia de Software

On the Performance of the Python Language

PG 54232
PG 55972
PG 57539