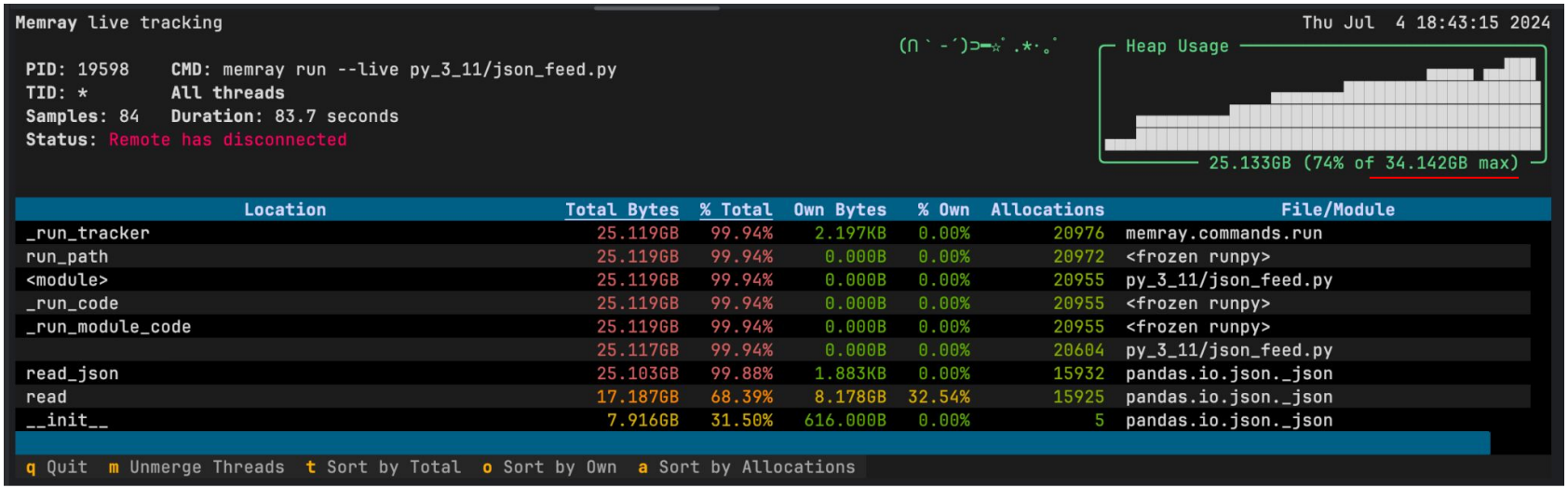
pycon.ru

Память, данные, python



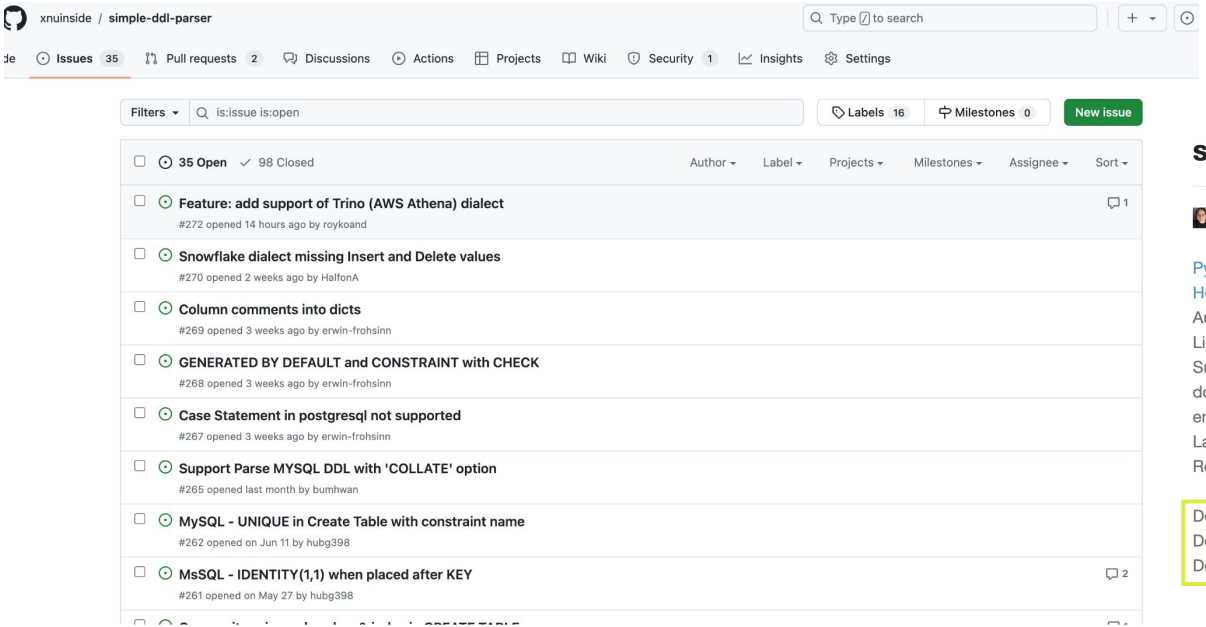
speak.er

> Юля Волкова

https://github.com/xnuinside

DataLead @ CODE SCORING

(оффтоп)



simple-ddl-parser

REMOVE from my packages

PyPI page
Home page
Author: Iuliia Volkova
License: MIT
Summary: Simple DDL Parser to parse SQL & dialects like HQL, TSQL (MSS
ddl files to json/python dict with full information about columns: types, defa
entities from ddl.
Latest version: 1.5.1
Required dependencies: dataclasses | ply

Downloads last day: 9,757
Downloads last week: 87,410
Downloads last month: 274,228

Про память

проблемки

- утечки
- swap
- OOM
- Неоптимальное использование/ неожиданные размеры

агенда

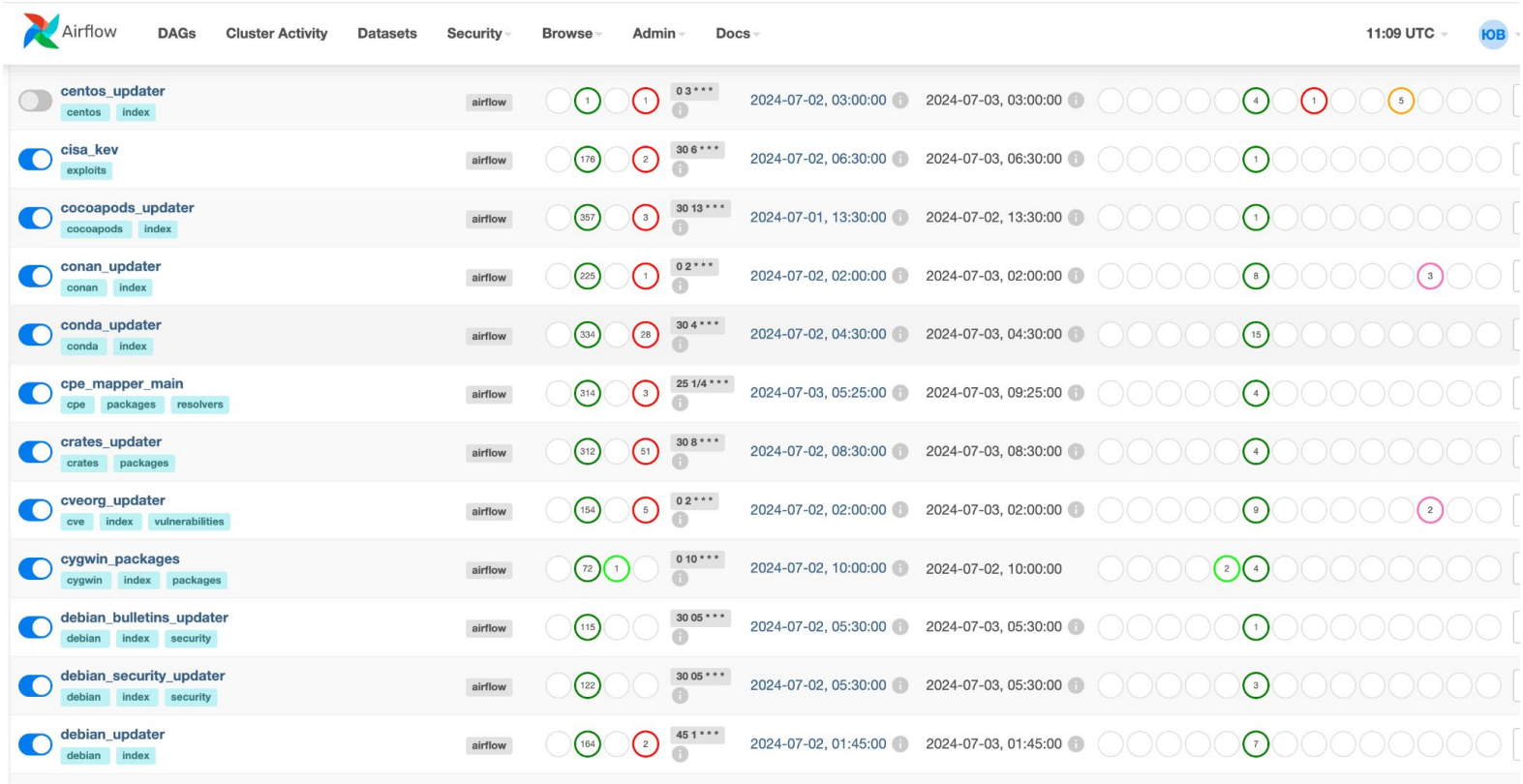
- расскажу про свой контекст
- пооптимизируем
- посмотрим на библиотеки и их поведение
- сравним разные версии питона
- вздохну про профайлеры
- проговорим очевидные вещи вслух

контекст

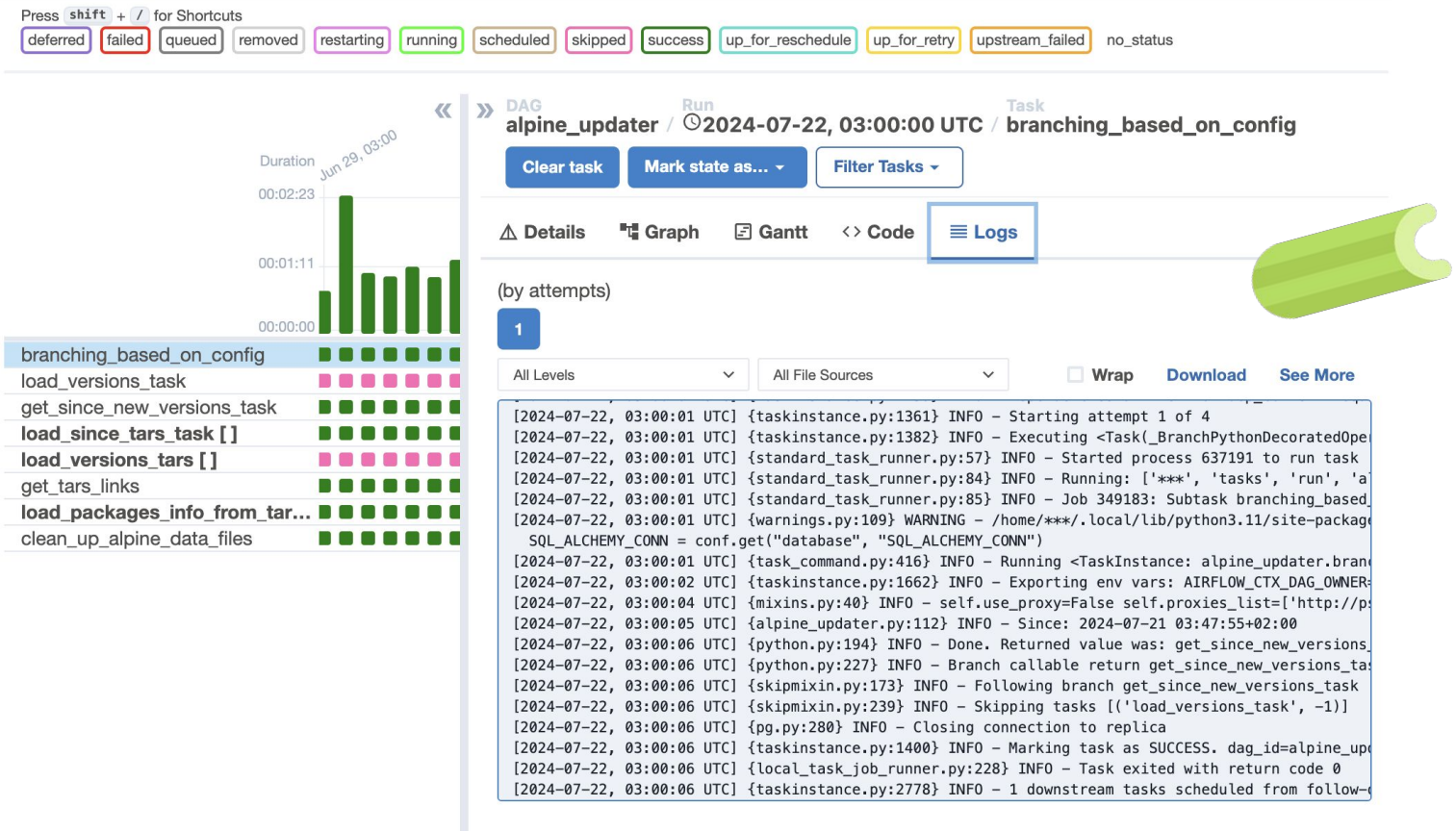
CodeScoring

Платформа композиционного анализа программного обеспечения

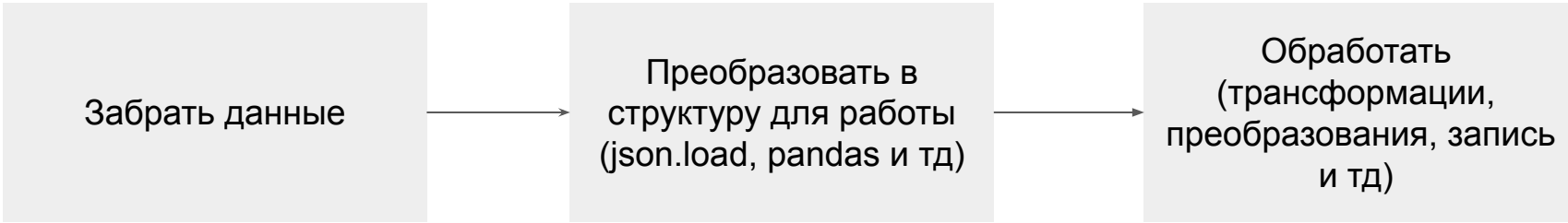
apache airflow



сельдерей


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
Jun 29, 03:00
00:02:23
00:01:11
00:00:00
branching_based_on_config
load_versions_task
get_since_new_versions_task
load_since_tars_task [ ]
load_versions_tars [ ]
get_tars_links
load_packages_info_from_tar...
clean_up_alpine_data_files
DAG
alpine_updater
Run
2024-07-22, 03:00:00 UTC
Task
branching_based_on_config
Clear task
Mark state as...
Filter Tasks
Details
Graph
Gantt
Code
Logs
(by attempts)
1
All Levels
All File Sources
Wrap
Download
See More
[2024-07-22, 03:00:01 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 4
[2024-07-22, 03:00:01 UTC] {taskinstance.py:1382} INFO - Executing <Task(_BranchPythonDecoratedOpe
[2024-07-22, 03:00:01 UTC] {standard_task_runner.py:57} INFO - Started process 637191 to run task
[2024-07-22, 03:00:01 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'a
[2024-07-22, 03:00:01 UTC] {standard_task_runner.py:85} INFO - Job 349183: Subtask branching_based_
[2024-07-22, 03:00:01 UTC] {warnings.py:109} WARNING - /home/***/.local/lib/python3.11/site-packag
SQL_ALCHEMY_CONN = conf.get("database", "SQL_ALCHEMY_CONN")
[2024-07-22, 03:00:01 UTC] {task_command.py:416} INFO - Running <TaskInstance: alpine_updater.bran
[2024-07-22, 03:00:02 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER=
[2024-07-22, 03:00:04 UTC] {mixins.py:40} INFO - self.use_proxy=False self.proxies_list=['http://p
[2024-07-22, 03:00:05 UTC] {alpine_updater.py:112} INFO - Since: 2024-07-21 03:47:55+02:00
[2024-07-22, 03:00:06 UTC] {python.py:194} INFO - Done. Returned value was: get_since_new_versions_
[2024-07-22, 03:00:06 UTC] {python.py:227} INFO - Branch callable return get_since_new_versions_ta
[2024-07-22, 03:00:06 UTC] {skipmixin.py:173} INFO - Following branch get_since_new_versions_task
[2024-07-22, 03:00:06 UTC] {skipmixin.py:239} INFO - Skipping tasks [('load_versions_task', -1)]
[2024-07-22, 03:00:06 UTC] {pg.py:280} INFO - Closing connection to replica
[2024-07-22, 03:00:06 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=alpine_up
[2024-07-22, 03:00:06 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2024-07-22, 03:00:06 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-

каждый dag …


Забрать данные
Преобразовать в структуру для работы (json.load, pandas и тд)
Обработать (трансформации, преобразования, запись и тд)

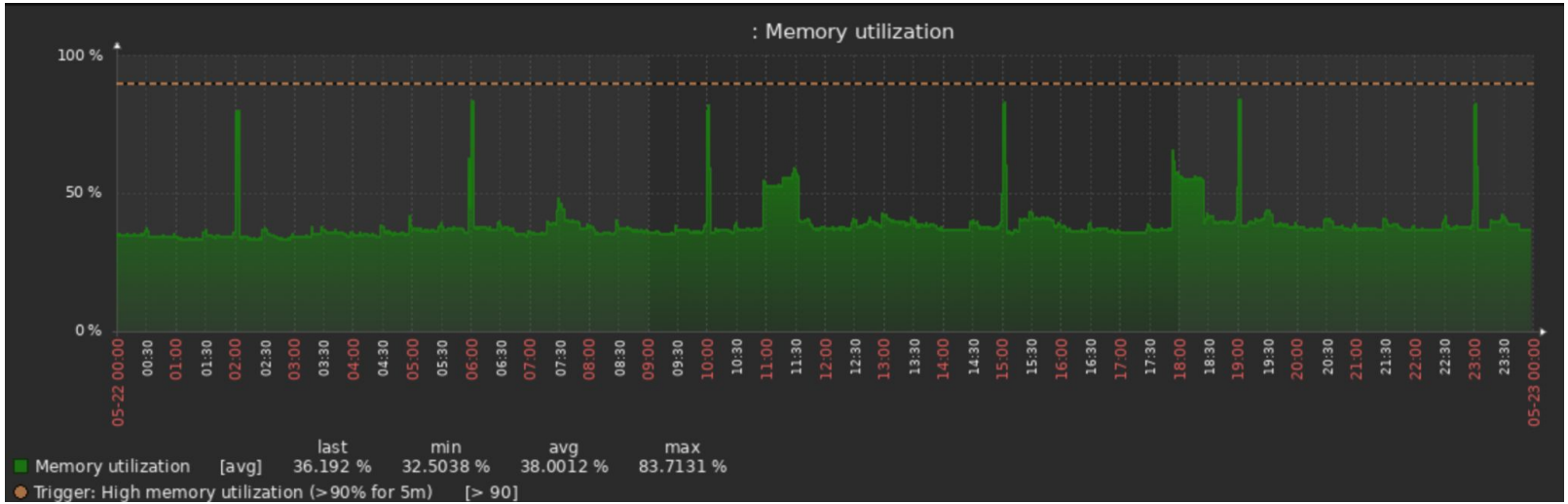
разнообразно

diff data

< 50 Mib

питончика достаточно

но иногда…


: Memory utilization
100 %
50 %
0 %
05-22 00:00
00:30
01:00
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30
05-23 00:00
last
min
avg
max
Memory utilization [avg] 36.192 % 32.5038 % 38.0012 % 83.7131 %
Trigger: High memory utilization (>90% for 5m) [> 90]

привет, oom!

```
ndard_task_runner.py:57} INFO - Started process 1199650 to run task
ndard_task_runner.py:84} INFO - Running: ['***', '
ndard_task_runner.py:85} INFO - Job 315100: Subtas
nings.py:109} WARNING - /home/***/.local/lib/python3.11/site-packages/***/settings.py:1
'database", "SQL_ALCHEMY_CONN")
k_command.py:416} INFO - Running <TaskInstance
kinstance.py:1662} INFO - Exporting env vars:
ders.py:56} INFO - Start loading
ders.py:69} INFO - Current version 2024-06-18T18:39:07.967
ders.py:70} INFO - Last version 2024-06-18T22:53:33.457
ders.py:79} INFO - Load vulnerabilities
al_task_job_runner.py:228} INFO - Task exited with return code -9
kinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

привет, oom!

```
ndard_task_runner.py:57} INFO - Started process 1199650 to run task
ndard_task_runner.py:84} INFO - Running: ['***', '
ndard_task_runner.py:85} INFO - Job 315100: Subtas
nings.py:109} WARNING - /home/***/.local/lib/python3.11/site-packages/***/settings.py:1
'database", "SQL_ALCHEMY_CONN")
k_command.py:416} INFO - Running <TaskInstance
kinstance.py:1662} INFO - Exporting env vars:
ders.py:56} INFO - Start loading
ders.py:69} INFO - Current version 2024-06-18T18:39:07.967
ders.py:70} INFO - Last version 2024-06-18T22:53:33.457
ders.py:79} INFO - Load vulnerabilities
al_task_job_runner.py:228} INFO - Task exited with return code -9
kinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-o
```

Если для вас это проблема - помните про ограничение памяти процессам

что могло пойти не так

питон иногда ооооооочень расслабляет

```
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))

        vulnerabilities = []
        vulnerable_packages = set()

        for file_ in zip_file.filelist:
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

шаг 1

```
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))

        vulnerabilities = []
        vulnerable_packages = set()

        for file_ in zip_file.filelist:
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

тут что-то загрузили

шаг 2

```
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))

        vulnerabilities = []
        vulnerable_packages = set()

        for file_ in zip_file.filelist:
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

тут бахнули все в память

шаг 3

```
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))

        vulnerabilities = []
        vulnerable_packages = set()

        for file_ in zip_file.filelist:
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

а тут мы типа умные, используем orjson

профайлер

memory_profiler

профайлер

Pympler

memory_profiler

профайлер

Pympler

resource.getrusage()

psutil.virtual_memory()

memory_profiler

профайлер

Pympler

TRACEMALLOC

resource.getrusage()

GUPPY

psutil.virtual_memory()

memory_profiler

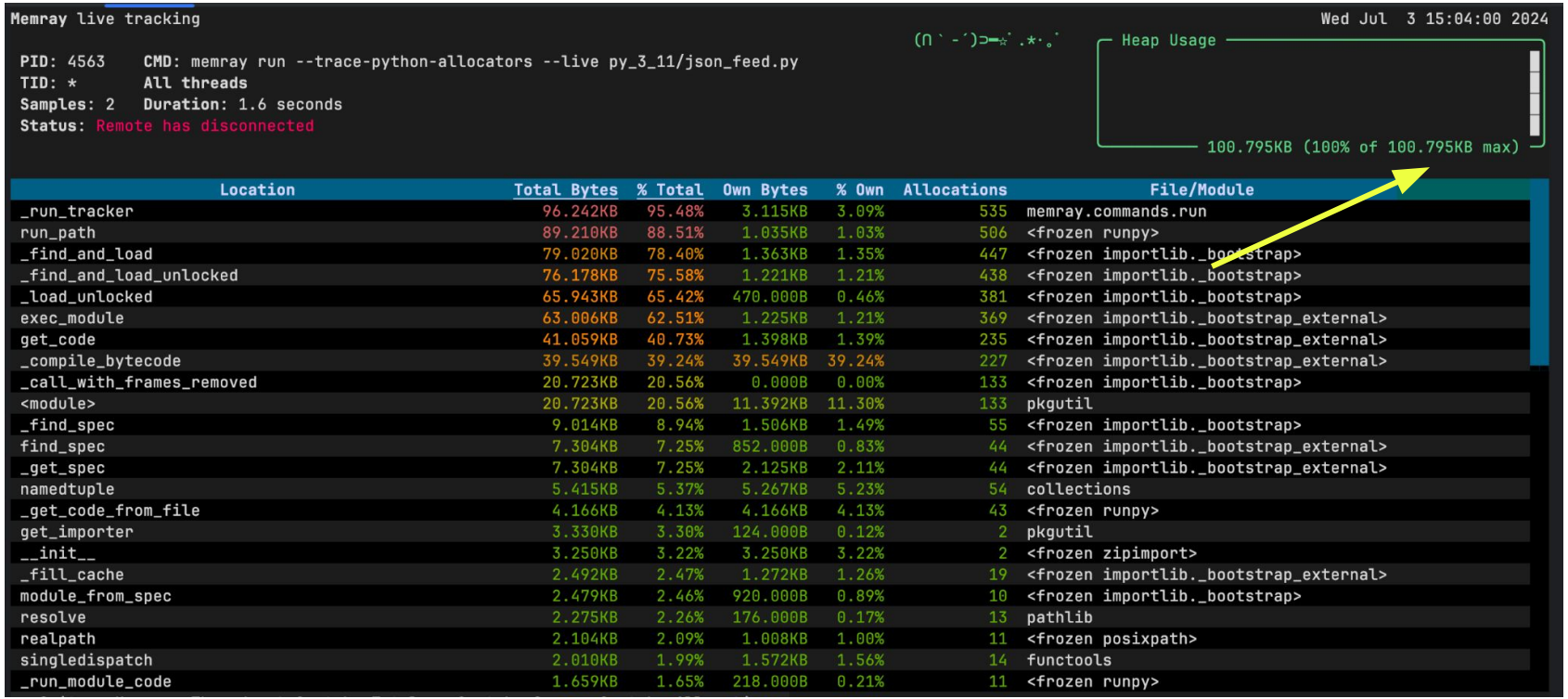
профайлер

конфигурация

- M1 Pro
- 32 Гб
- Sonoma 14.4.1

Python 3.11

почти пустой файл

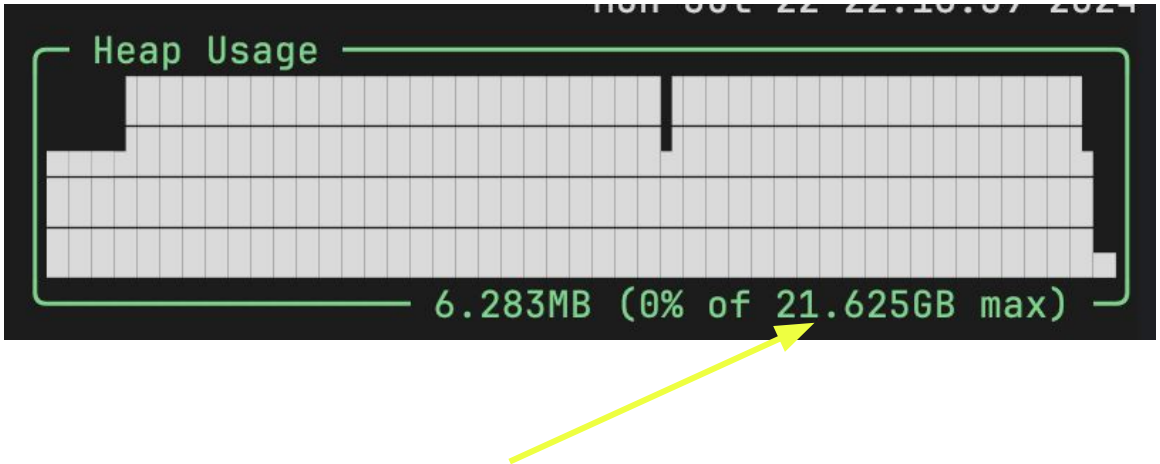


Location	Total Bytes	% Total	Own Bytes	% Own	Allocations	File/Module
_run_tracker	96.242KB	95.48%	3.115KB	3.09%	535	memray.commands.run
run_path	89.210KB	88.51%	1.035KB	1.03%	506	<frozen runpy>
_find_and_load	79.020KB	78.40%	1.363KB	1.35%	447	<frozen importlib._bootstrap>
_find_and_load_unlocked	76.178KB	75.58%	1.221KB	1.21%	438	<frozen importlib._bootstrap>
_load_unlocked	65.943KB	65.42%	470.000B	0.46%	381	<frozen importlib._bootstrap>
exec_module	63.006KB	62.51%	1.225KB	1.21%	369	<frozen importlib._bootstrap_external>
get_code	41.059KB	40.73%	1.398KB	1.39%	235	<frozen importlib._bootstrap_external>
_compile_bytecode	39.549KB	39.24%	39.549KB	39.24%	227	<frozen importlib._bootstrap_external>
_call_with_frames_removed	20.723KB	20.56%	0.000B	0.00%	133	<frozen importlib._bootstrap>
<module>	20.723KB	20.56%	11.392KB	11.30%	133	pkgutil
_find_spec	9.014KB	8.94%	1.506KB	1.49%	55	<frozen importlib._bootstrap>
find_spec	7.304KB	7.25%	852.000B	0.83%	44	<frozen importlib._bootstrap_external>
_get_spec	7.304KB	7.25%	2.125KB	2.11%	44	<frozen importlib._bootstrap_external>
namedtuple	5.415KB	5.37%	5.267KB	5.23%	54	collections
_get_code_from_file	4.166KB	4.13%	4.166KB	4.13%	43	<frozen runpy>
get_importer	3.330KB	3.30%	124.000B	0.12%	2	pkgutil
__init__	3.250KB	3.22%	3.250KB	3.22%	2	<frozen zipimport>
_fill_cache	2.492KB	2.47%	1.272KB	1.26%	19	<frozen importlib._bootstrap_external>
module_from_spec	2.479KB	2.46%	920.000B	0.89%	10	<frozen importlib._bootstrap>
resolve	2.275KB	2.26%	176.000B	0.17%	13	pathlib
realpath	2.104KB	2.09%	1.008KB	1.00%	11	<frozen posixpath>
singledispatch	2.010KB	1.99%	1.572KB	1.56%	14	functools
_run_module_code	1.659KB	1.65%	218.000B	0.21%	11	<frozen runpy>

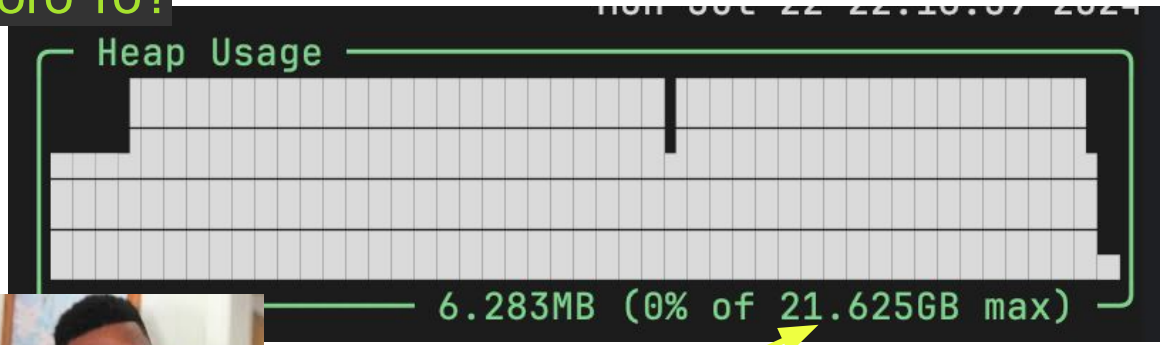
просто прочтем файл

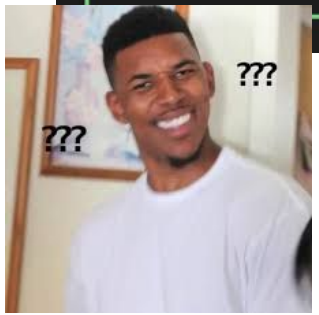
```
def read_only_file():
    with open(full_feed, 'r') as f:
        data = f.read()
```

Во сколько превратились 4,64 Гб json ?


Heap Usage
6.283MB (0% of 21.625GB max)

А чЁ так много то?





нельзя “просто” прочитать в utf-8

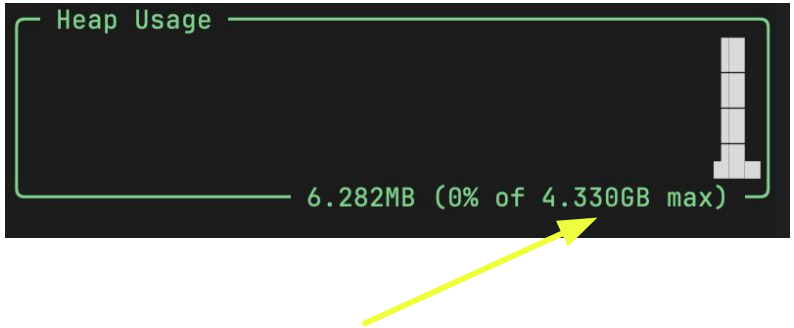
```
def read_only_file():
    with open(full_feed, 'r') as f:
        data = f.read()
```

помним про ‘b’

```
def read_only_file():
    with open(full_feed, 'rb') as f:
        data = f.read()
```

помним про ‘b’

```
def read_only_file():
    with open(full_feed, 'rb') as f:
        data = f.read()
```



помним про ‘b’

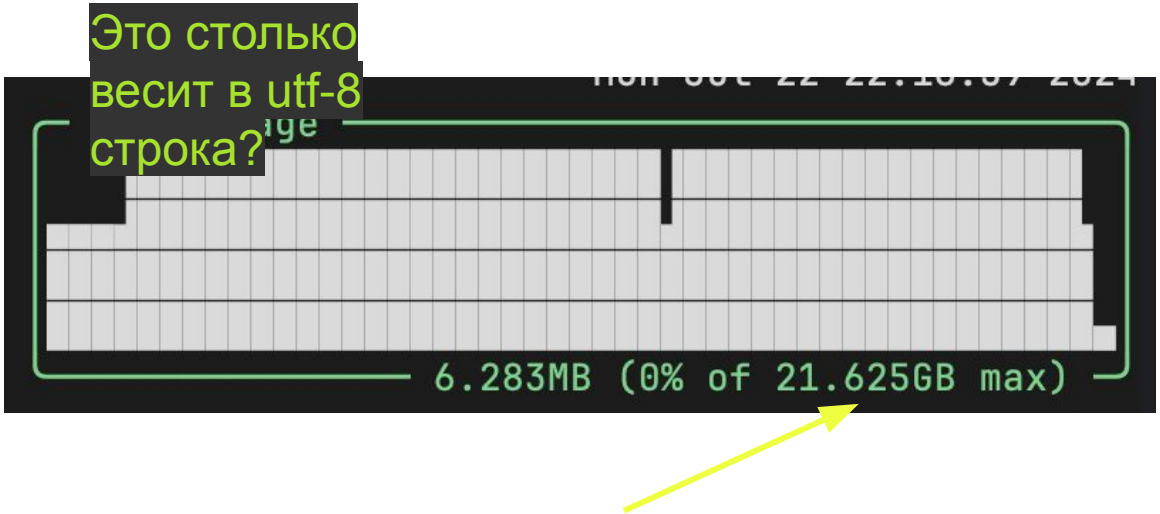
```
import sys
from pympler import asizeof


def read_only_file():
    with open(full_feed, 'rb') as f:
        data = f.read()

        print('sys.getsizeof(json_data): ',
sys.getsizeof(data) >> 20, 'mb')
        print('pympler (asizeof): ',
asizeof.asizeof(data) >> 20, 'mb')
```

помним про ‘b’

```
import sys
from pympler import asizeof


def read_only_file():
    with open(full_feed, 'rb') as f:
        data = f.read()

        print('sys.getsizeof(json_data): ',
sys.getsizeof(data) >> 20, 'mb')
        print('pympler (asizeof): ',
asizeof.asizeof(data) >> 20, 'mb')
```

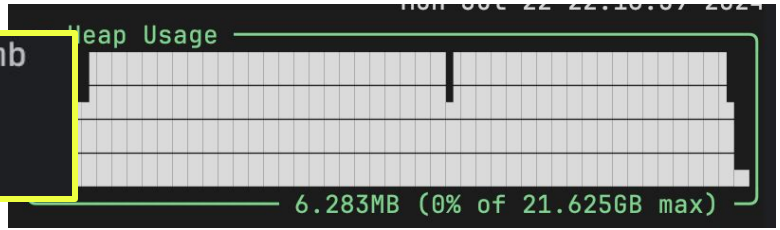
```
os.path.getsize(full_feed):  4427 mb
sys.getsizeof(data):  4427 mb
pympler (asizeof): 4427 mb
```

возвращаемся к mode = ‘r’



возвращаемся к mode = ‘r’

```
os.path.getsize(full_feed):  4427 mb
sys.getsizeof(data):  8707 mb
pympler (asizeof):  8707 mb
```



откуда max?

https://github.com/python/cpython/blob/main/Objects/unicodeobject.c#L160

```
/* Generic helper macro to convert characters of different types.
   from_type and to_type have to be valid type names, begin and end
   are pointers to the source characters which should be of type
   "from_type *".  to is a pointer of type "to_type *" and points to the
   buffer where the result characters are written to. */
```

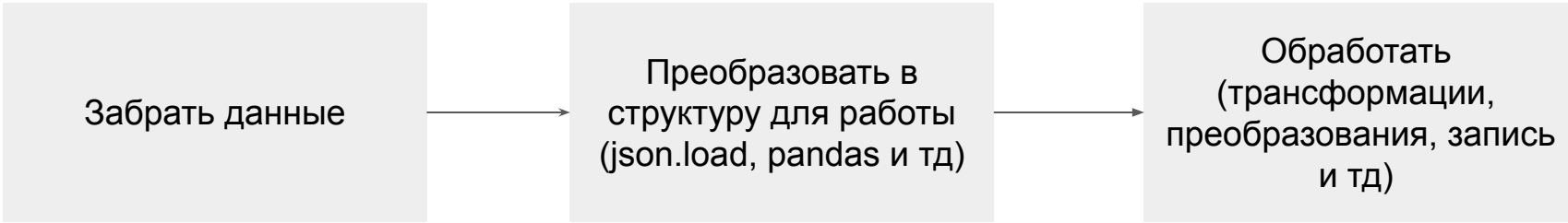
```
_PyUnicodeWriter_WriteCharInline(_PyUnicodeWriter *writer, Py_UCS4 ch);
static inline void
_PyUnicodeWriter_InitWithBuffer(_PyUnicodeWriter *writer, PyObject *buffer);
static PyObject *
unicode_encode_utf8(PyObject *unicode, _Py_error_handler error_handler,
                    const char *errors);
```

… вернемся

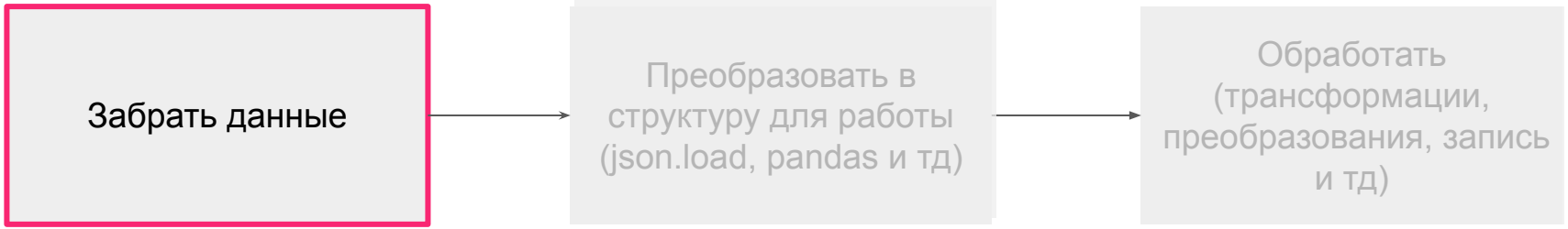
вспомнили про:

- байты
- буферизацию
- декодинг
- максимальное потребление памяти

вы уже наверно забыли …

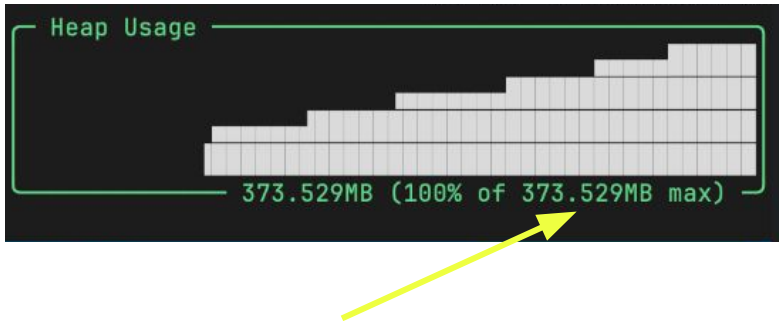


вы уже наверно забыли …



скачать-закачать

```python
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))
```



Конечно тут gzip

стриминг

https://www.python-httpx.org/quickstart/#streaming-responses

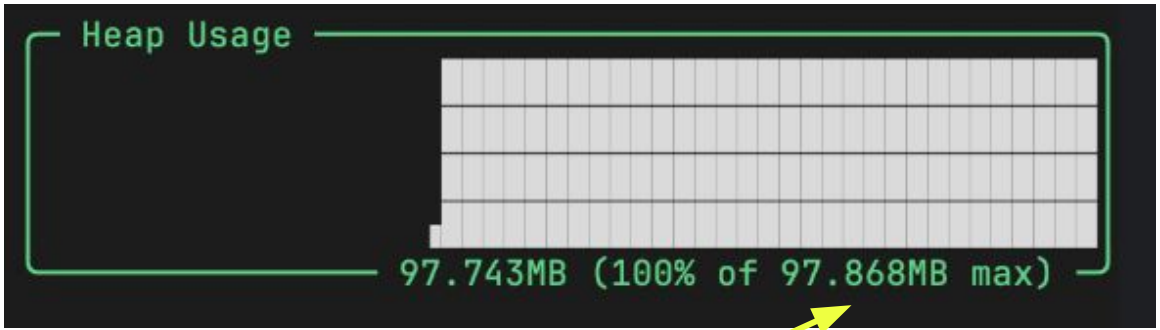
Streaming Responses

For large downloads you may want to use streaming responses that do not load the entire response body into memory at once.

You can stream the binary content of the response...

```
>>> with httpx.stream("GET", "https://www.example.com") as r:
...     for data in r.iter_bytes():
...         print(data)
```

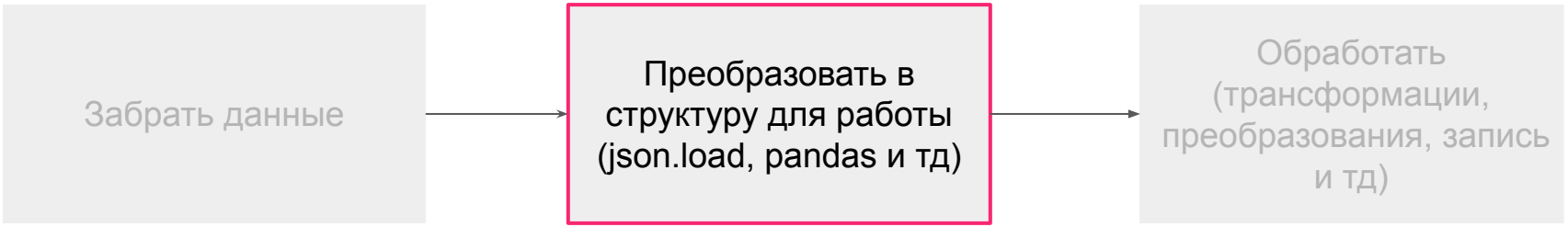
переписали на стриминг



стало: 98 MB

было: 373 MB

далее

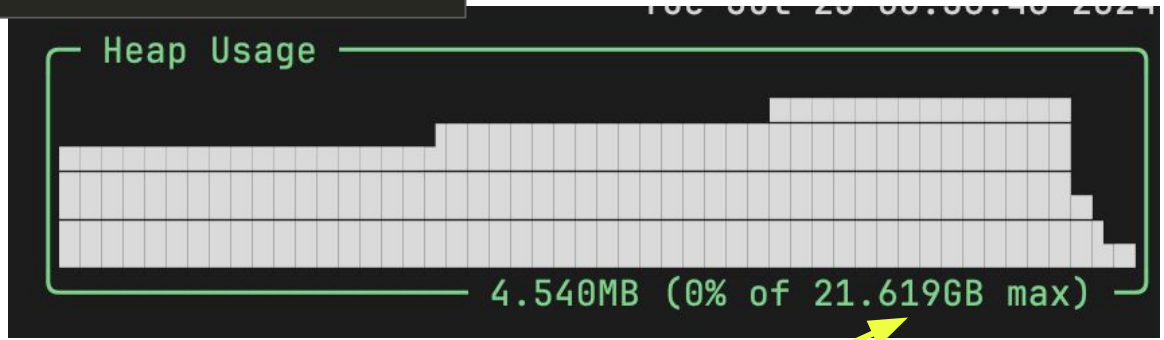

Забрать данные
Преобразовать в структуру для работы (json.load, pandas и тд)
Обработать (трансформации, преобразования, запись и тд)

json

```
import json


def load_python_pure_json():
    with open(full_feed, 'rb') as f:
        json_loaded = json.load(f)
```

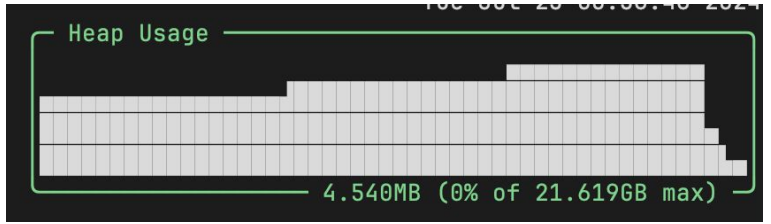
json

```python
import json


def load_python_pure_json():
    with open(full_feed, 'rb') as f:
        json_loaded = json.load(f)
```



json

```
sys.getsizeof(data):  0 mb
pympler (asizeof):  7470 mb
```



Pympler работает ооооочень медленно

но у нас же и так был orjson!

```
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

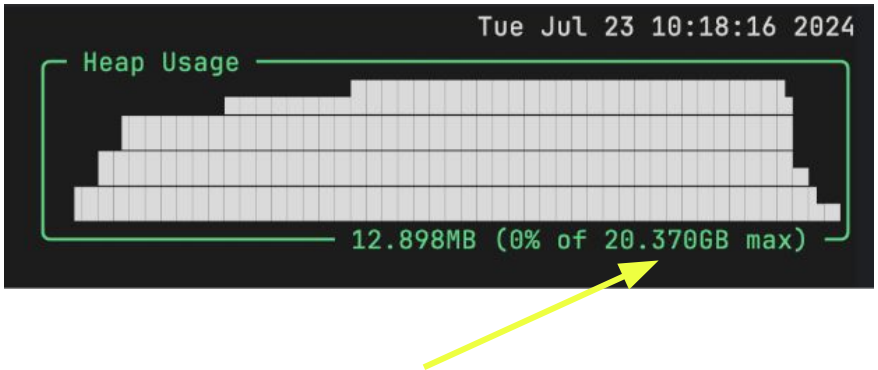
https://github.com/ijl/orjson

orjson

```python
import orjson


def load_orjson():

    with open(full_feed, 'rb') as f:
        data = f.read()
        json_data = orjson.loads(data)
```

orjson

```python
import orjson


def load_orjson():

    with open(full_feed, 'rb') as f:
        data = f.read()
        json_data = orjson.loads(data)
```



orjson + ‘r’

```
with open(full_feed, 'r') as f:
```

orjson + ‘r’

```
with open(full_feed, 'r') as f:
```

orjson: ‘rb’ vs ‘r’

‘rb’: 20.37 GB

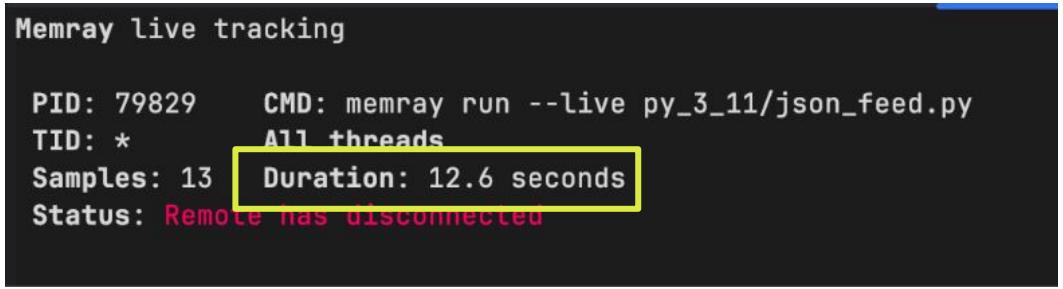
‘r’: 28.659 GB

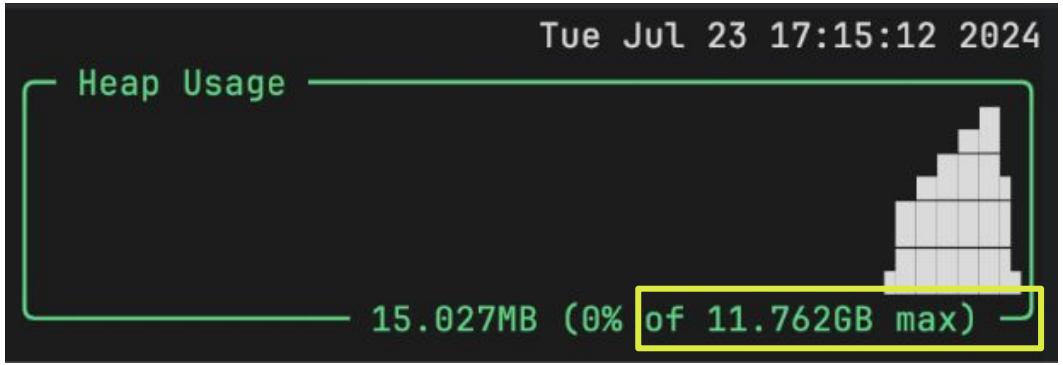
msgspec

```
import msgspec


def load_msgspec():

    with open(full_feed, 'rb') as f:
        data = f.read()
        json_data = msgspec.json.decode(data)
```

https://github.com/jcrist/msgspec

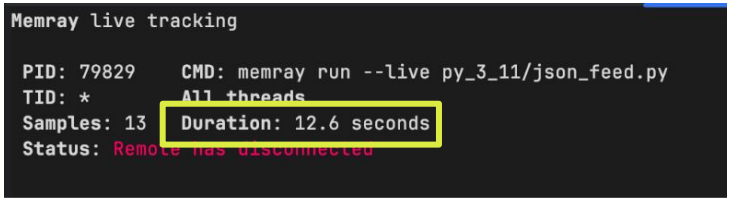
msgspec

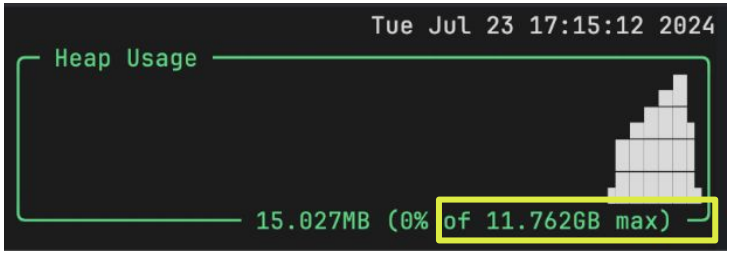

Memray live tracking
PID: 79829
CMD: memray run --live py_3_11/json_feed.py
TID: *
All threads
Samples: 13
Duration: 12.6 seconds
Status: Remote has disconnected


Tue Jul 23 17:15:12 2024
Heap Usage
15.027MB (0% of 11.762GB max)

msgspec


msgspec


Memray live tracking
PID: 79829
CMD: memray run --live py_3_11/json_feed.py
TID: *
All threads
Samples: 13
Duration: 12.6 seconds
Status: Remote has disconnected

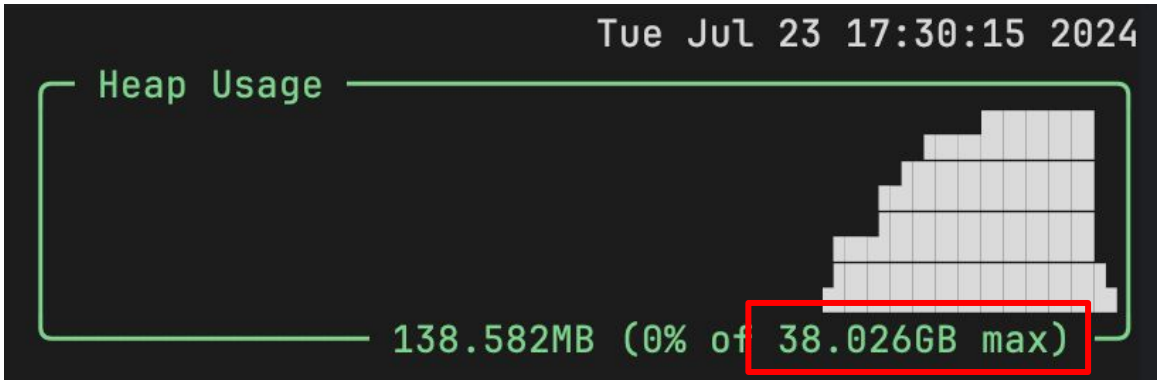

Tue Jul 23 17:15:12 2024
Heap Usage
15.027MB (0% of 11.762GB max)

а ещёёёёёё …

- polars
- pandas

угадайте насколько все плохо с pandas

pandas



самая скучная часть


Ой, а может мы не будем просто все ПИХАТЬ в память?


Ой, а может мы не будем просто все ПИХАТЬ в память?
Боже, ты такой умный!

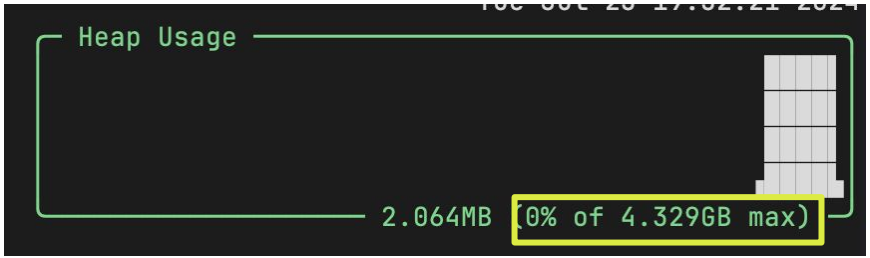
ijson

```
import ijson


def load_ijson():

    with open(full_feed, 'rb') as f:
        data = f.read()
        data_el = ijson.items(data, 'item')
        for num, item in enumerate(data_el):
            if num == 1000000:
                break
```

https://github.com/ICRAR/ijson

ijson

```python
import ijson


def load_ijson():

    with open(full_feed, 'rb') as f:
        data = f.read()
        data_el = ijson.items(data, 'item')
        for num, item in enumerate(data_el):
            if num == 1000000:
                break
```



https://github.com/ICRAR/ijson

ijson + r = disaster

А что если не json

Если этой датой управляете вы сами

Помним, что есть

- Parquet
- Avro
- Orc

и вообще BigData-мир …

и можно фиды делать по датам,

не пихая все в один json …

зафиксируем

msgspec - хорошо

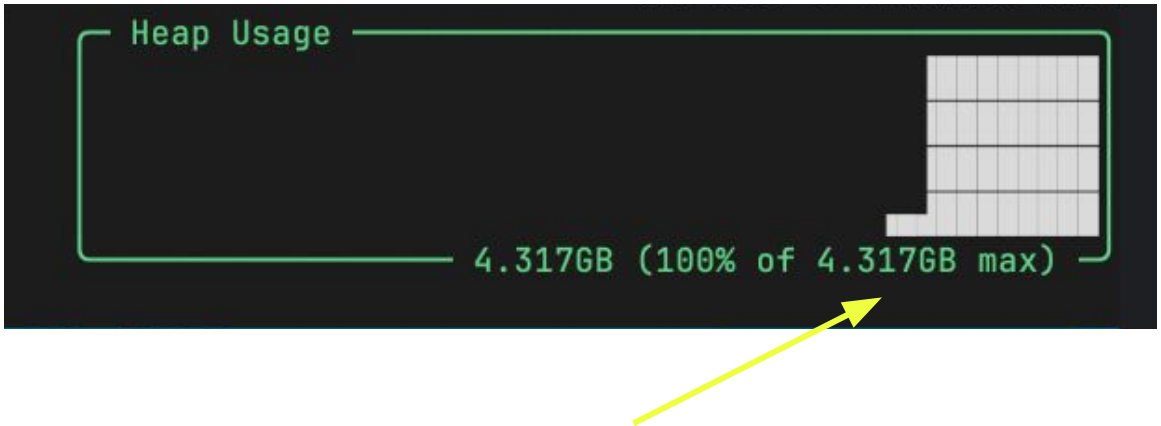
C, Rust, etc ext - байты, не декодируйте

Помним про ‘b’

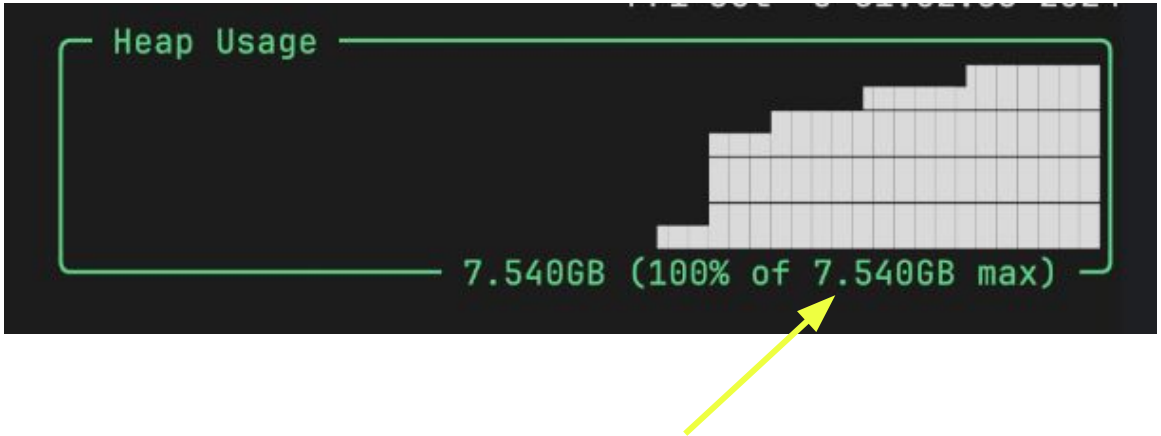
Стриминг-чтение - ответ

К реальному кейсу!

1 день - процессинг



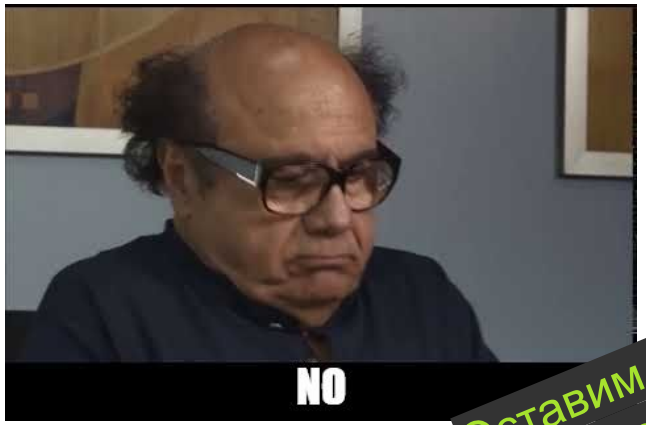
2 месяца



```
def load_vulnerabilities(self, url: str) -> None:
        self.logger.info('Load vulnerabilities')

        response = self.fetch(url=url, method='GET')
        zip_file = ZipFile(BytesIO(response.content))

        vulnerabilities = []
        vulnerable_packages = set()

        for file_ in zip_file.filelist:
            with zip_file.open(file_) as f:
                content = f.read()

                file_json = orjson.loads(content)

                for vulnerability in file_json:
                    parsed = Parser(**vulnerability)

vulnerabilities.append(tuple(parsed.model_dump().values()))
```

Тут же еще сериализация перед записью в базу

буду я с этим что-то делать?



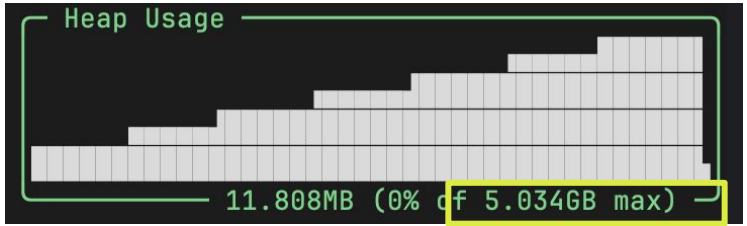
Оставим эту историю для другого доклада - скоро обед

если серьезно

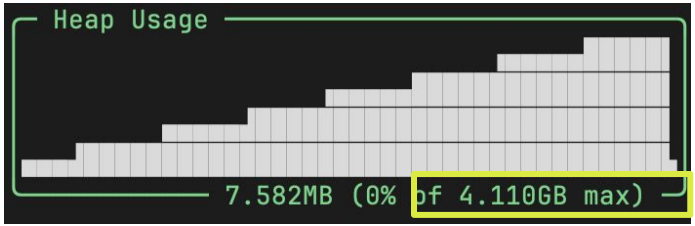
```python
class Model(BaseModel):

    id: str
    severity: str
    description: None | str = None
    affected_packages: None | list[dict] = None
    modified_at: None | datetime = None
    created_date: None | date = None
```

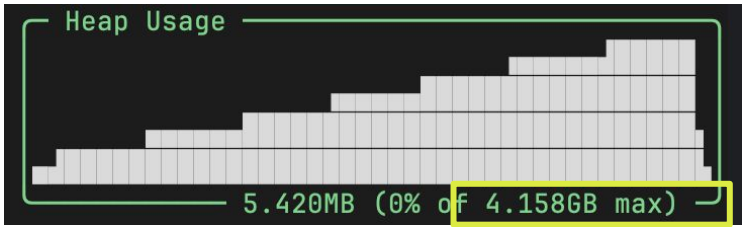
```python
def generate_set_of_models():
    models = []
    for i in range(1_000_000):

models.append(Model(**generate_data_eleme
nt()))
    return models
```

pydantic

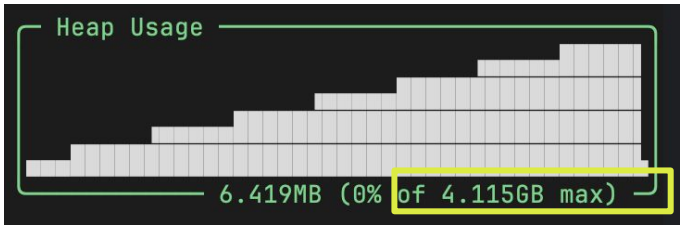


msgspec





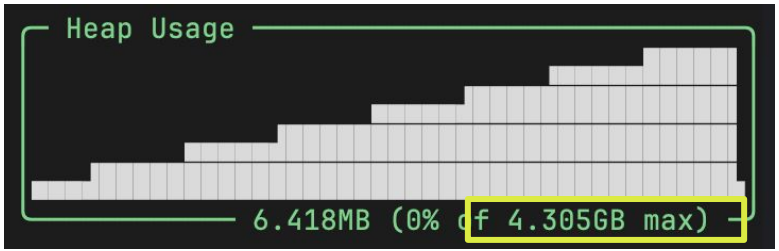
pure class



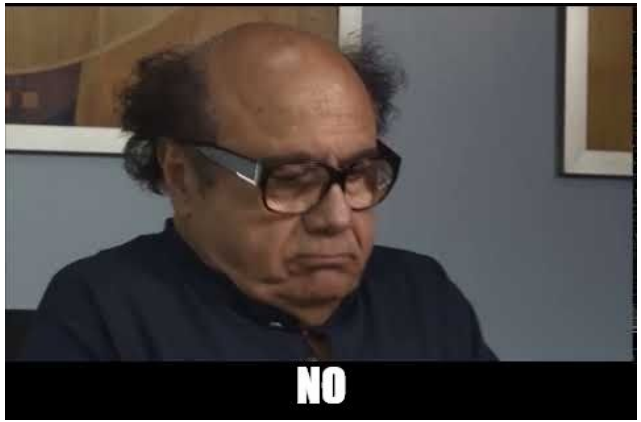
pure class + __slots__

без моделей

```
def generate_set_of_data():
    data = []
    for i in range(1_000_000):

data.append(generate_data_element())
    return data
```

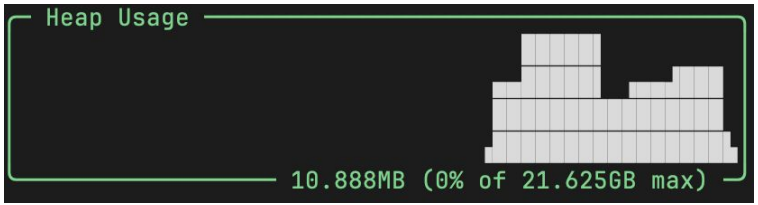


поэтому



про профайлеры

memray



memory_profiler

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    12     34.3 MiB     34.3 MiB           1   @profile
    13                                         def load_python_pure_json():
    14     34.3 MiB      0.0 MiB           1       import json
    15   8120.0 MiB      0.1 MiB           2       with open(full_feed, 'r') as f:
    16   8119.9 MiB   8085.6 MiB           1           json_loaded = json.load(f)
```

resource.getrusage()

```
resource.getrusage(resource.RUSAGE_SELF)
ru_maxrss: 16809 mb
```

Pympler

```
                types |   # objects |   total size
===================== | =========== | ============
                  str |    71103985 |      6.16 GB
                 list |     1504420 |    667.57 MB
                 dict |     1255696 |    229.19 MB
                float |       67350 |      1.54 MB
                  int |         436 |     11.92 KB
  _io.BufferedReader |           1 |      4.16 KB
   _io.TextIOWrapper |           1 |    208      B
```

memory_profiler

```
def read_only_file():
    with open(full_feed, 'r') as f:
        data = f.read()
```

```
os.path.getsize(full_feed):  4427 mb
sys.getsizeof(data):  8707 mb
pympler (asizeof):  8707 mb
```

memory_profiler

```python
def read_only_file():
    with open(full_feed, 'r') as f:
        data = f.read()
```

```
os.path.getsize(full_feed):  4427 mb
sys.getsizeof(data):  8707 mb
pympler (asizeof):  8707 mb
```

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    13     24.1 MiB     24.1 MiB           1   @profile
    14                                         def read_only_file():
    15    595.2 MiB      0.3 MiB           2       with open(full_feed, 'r') as f:
    16    594.8 MiB    570.8 MiB           1           data = f.read()
```

memory_profiler

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    13     23.4 MiB     23.4 MiB           1   @profile
    14                                         def read_only_file():
    15    399.6 MiB      0.5 MiB           2       with open(full_feed, 'r') as f:
    16    399.1 MiB    375.7 MiB           1           data = f.read()
```

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    13     23.5 MiB     23.5 MiB           1   @profile
    14                                         def read_only_file():
    15    498.7 MiB      0.4 MiB           2       with open(full_feed, 'r') as f:
    16    498.3 MiB    474.8 MiB           1           data = f.read()
```

А что там с изменениями в Python?

рубрика “золотые цитаты”

Will CPython 3.11 use more memory?

Maybe not; we don’t expect memory use **to exceed 20%** higher than 3.10.

This is offset by memory optimizations for frame objects and object dictionaries as mentioned above.

в GB

	3.8	3.11	3.12
read	4.438	4.438	4.437
json	20.818	20.668	20.405
orjson	20.617	20.499	18.922
msgspec	12.022	11.9	11.469
ijson	4.326	4.328	4.328

Python 3.8

- Removed one `Py_ssize_t` member from `PyGC_Head`. All GC tracked objects (e.g. tuple, list, dict) size is reduced 4 or 8 bytes. (Contributed by Inada Naoki in bpo-33597.)

Python 3.11

Lazily create dictionaries for plain Python objects. Objects now require less memory due to lazily created object namespaces. Their namespace dictionaries now also share keys more freely. (Contributed Mark Shannon in bpo-45340 and bpo-40116.)

с Python 3.9 по Python 3.12

PEP 623: Remove wstr from Unicode objects in Python’s C API, reducing the size of every str object by at least 8 bytes. - очень долгий путь для дропа легаси-юникод объектов https://peps.python.org/pep-0623/

Оно же:

Remove `wstr` and `wstr_length` members from Unicode objects. It reduces object size by 8 or 16 bytes on 64bit platform. (**PEP 623**) (Contributed by Inada Naoki in gh-92536.)

выводы капитана

- Память не в приоритете - кому оно надо!
- помним про байт-представление
- стримим на диск
- профайлеры - хорошо, но надо понимать что это за цифры
- если можно не грузить все в память - не грузим
- DS-библиотеки - хорошо, но только если нет больших объемов и если важна функциональность
- И тд, и тп

СПАСИБО, У МЕНЯ ВСЁ

pycon.ru

Сюда можно накидать отзывов