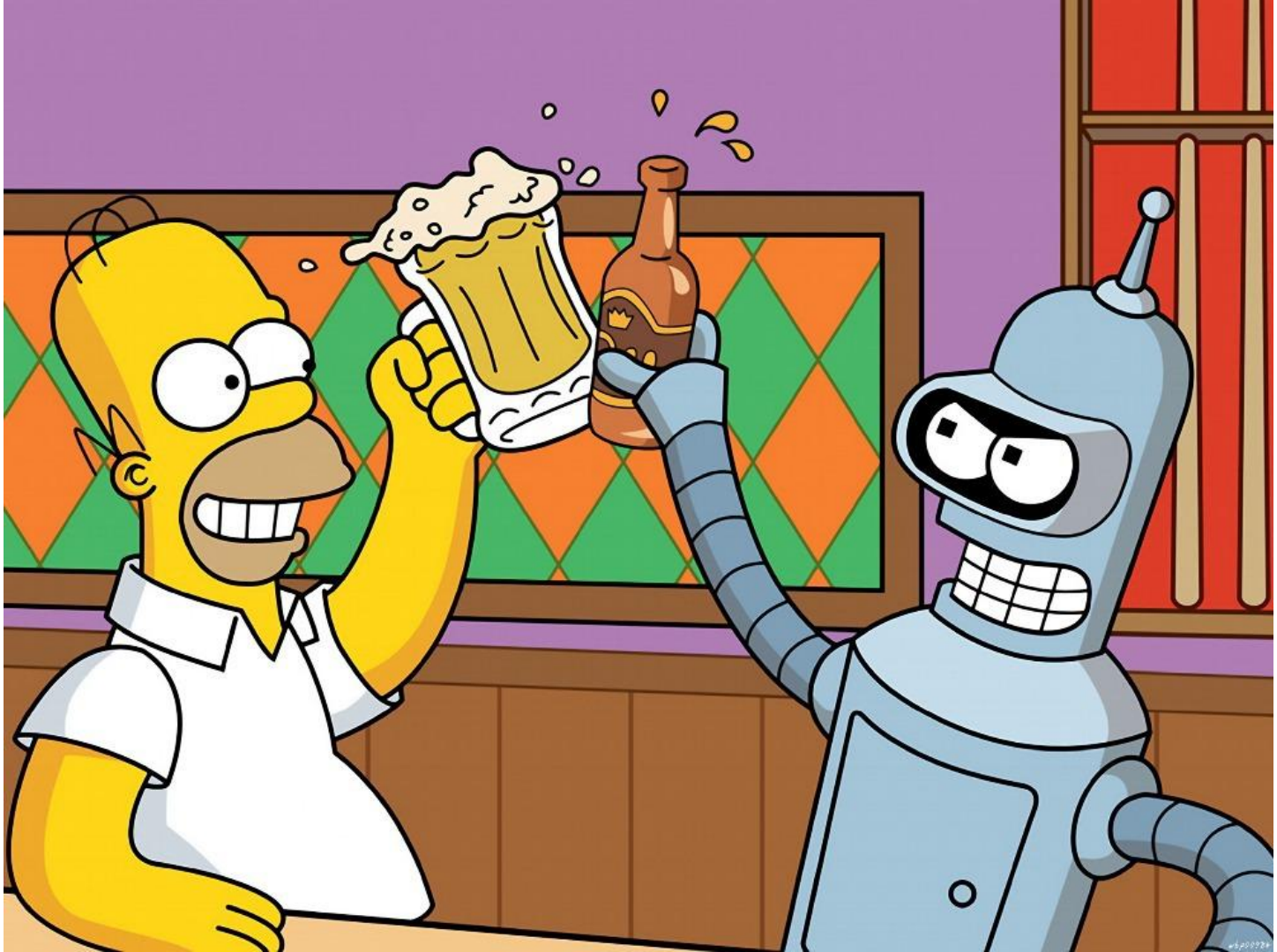
Внедрение C++ расширений в Python

Александр Боргардт
IVA Cognitive
CyberDuckNinja.com
(non-profit/community-driving)

2016




Service
Recommendation*

Servise

Service
Recommendation*

http

Servise

Service
Recommendation*

http

logic

input
Binary Data.csv

Servise

Service
Recommendation*

http

Redis
Client

logic

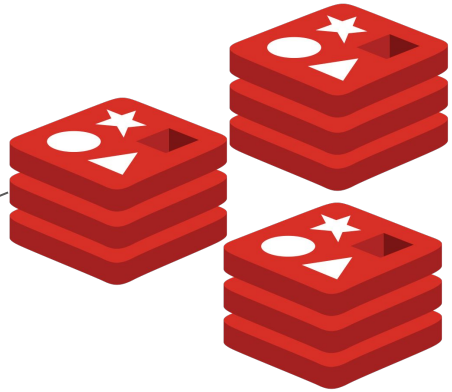
Servise

Service
Recommendation*

http

Redis
Client

logic



Servise

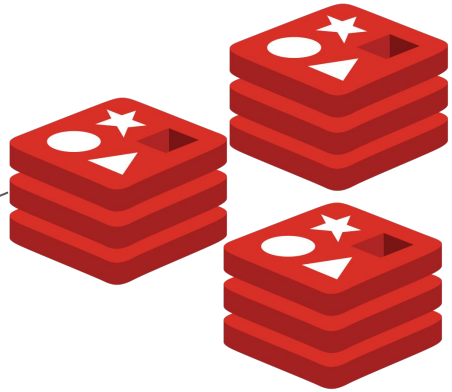
Service
Recommendation*

http

Redis
Client

logic

Cache



Output
Append online
Binary Data

Servise

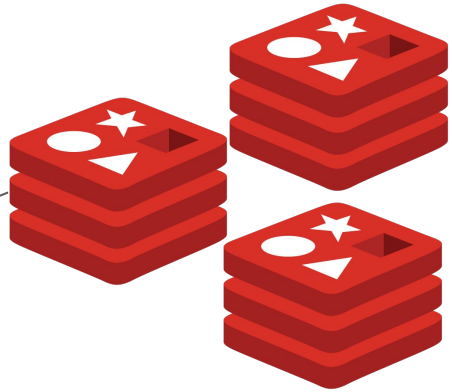
Service
Recommendation*

http

Redis
Client

logic

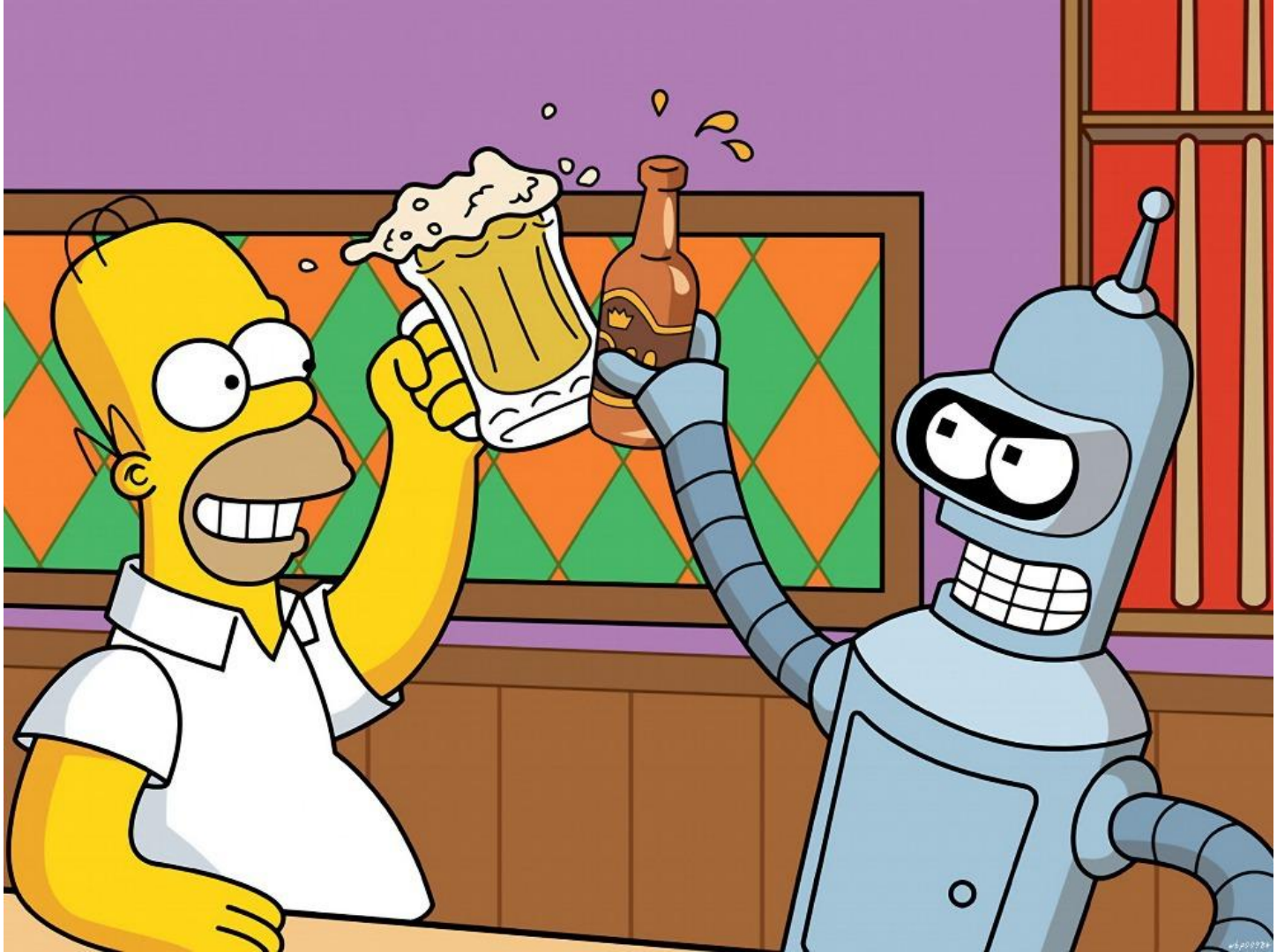
Cache



input
Binary Data.csv

Output
Append online
Binary Data

2018



Servise

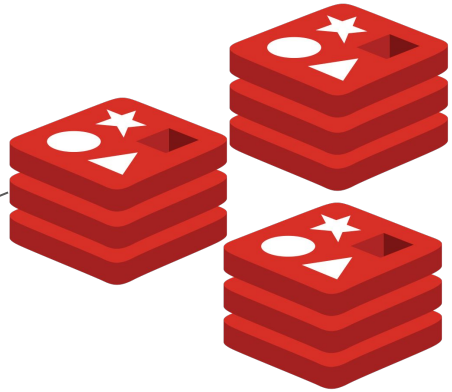
Service
Recommendation*

http

Redis
Client

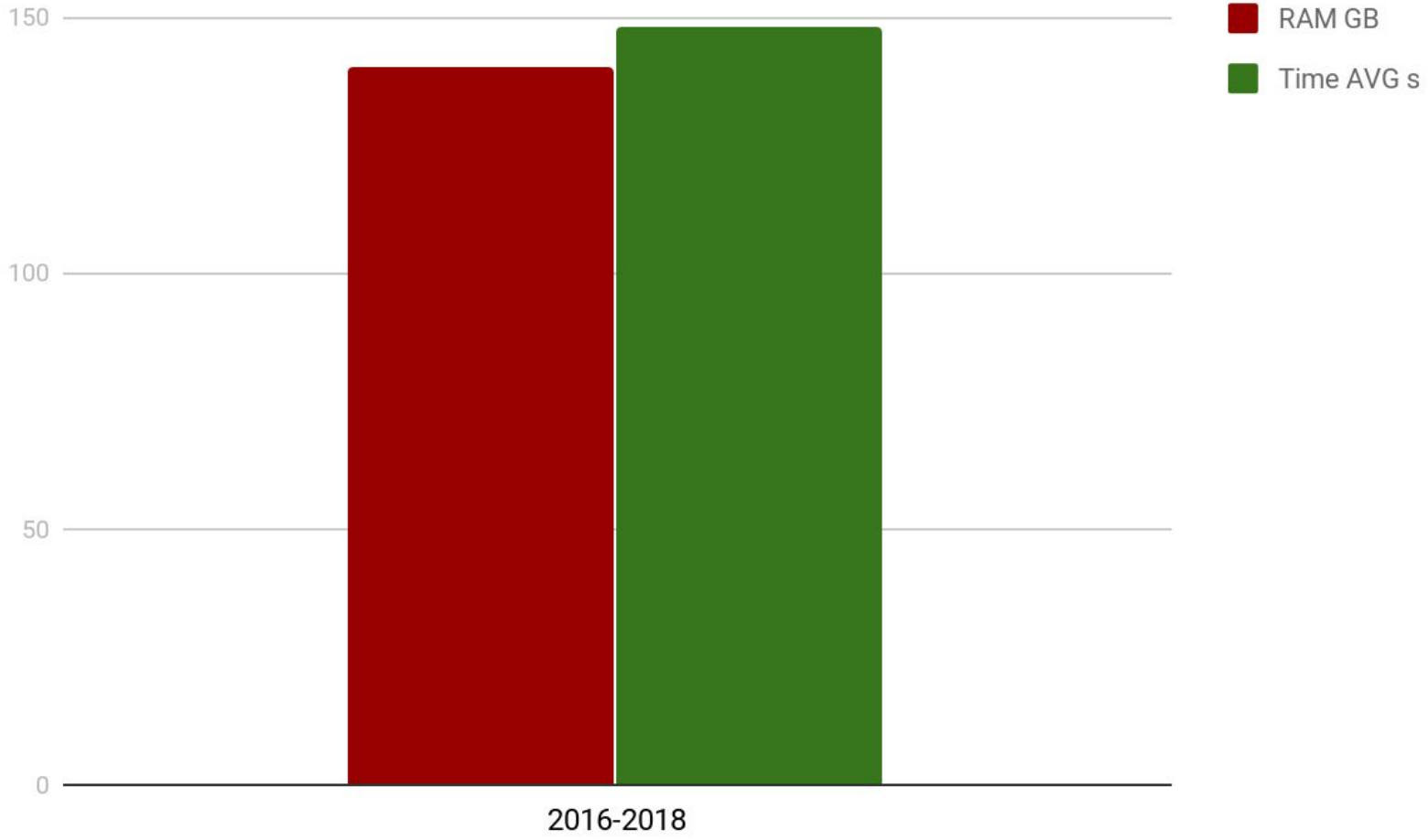
logic

Cache



input
Binary Data.csv

Output
Append online
Binary Data


150
100
50
0
2016-2018
RAM GB
Time AVG s

Servise

Service
Recommendation*

http

logic

Cache

input
Binary Data.csv

Output
Append online
Binary Data

Servise

Service
Recommendation*

http

logic

input
Binary Data.csv

Output
Append online
Binary Data

Servise

Service
Recommendation*

http

kv-storage

logic

input
Binary Data.csv

Output
Append online
Binary Data

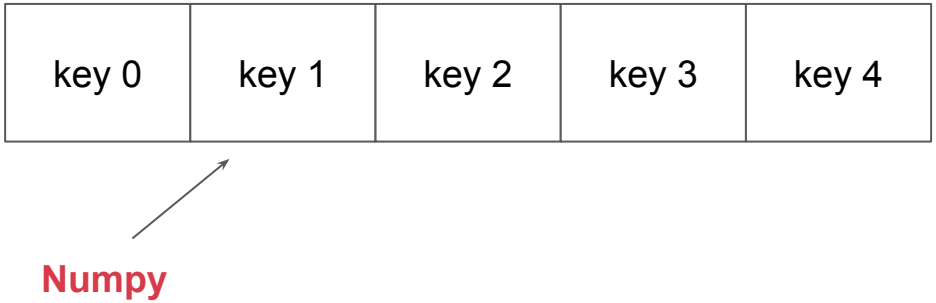
```
python3  main.py

from kv_storage import Storage
#processing
kv = Storage(“/app/data” )
#processing
with open(“binary_data.csv”) as file:
        for i in file:
                #processing
                key = Key(...)
                data= ….
                kv.put(key,data)
                #processing

#processing
result = kv.find(“hash-06-06-2018-*”)
#processing
#start http server
```


Singaltone reposit
kv_storage
storage hash
storage map
storage set
collection_N
part_N

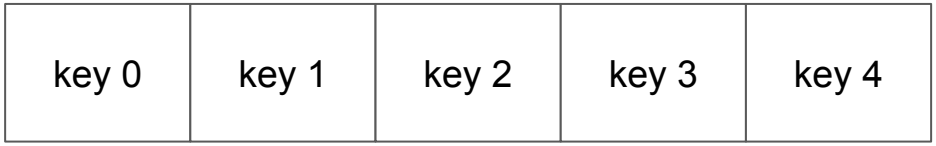
Key Index



prefix-tree + n-tree

C++

Key Index



Numpy

prefix-tree + n-tree

C++

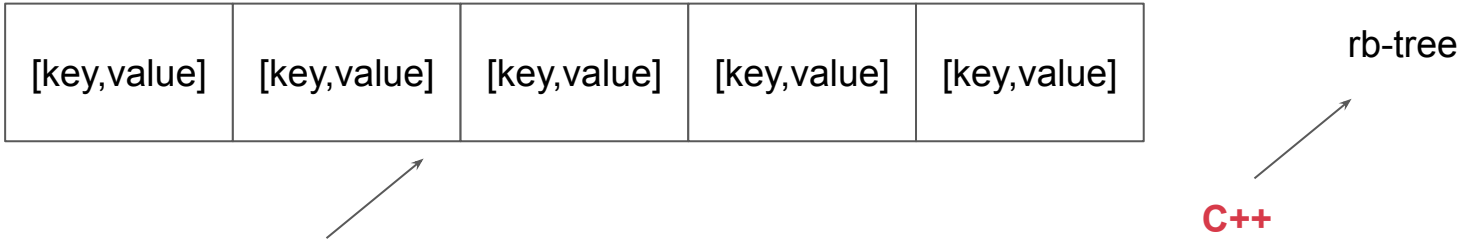
Set

Numpy + Trick and Hack

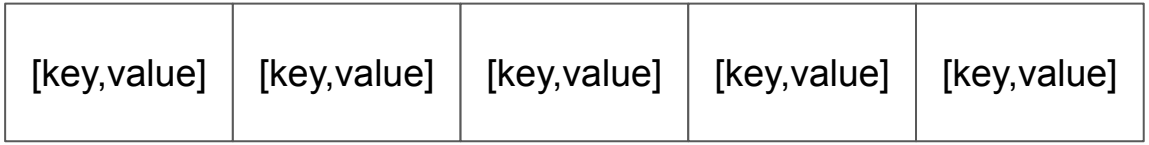
rb-tree / skip-list

C++

Map


[key,value]
[key,value]
[key,value]
[key,value]
[key,value]
rb-tree
C++
Numpy + Tricks and Hack

Map

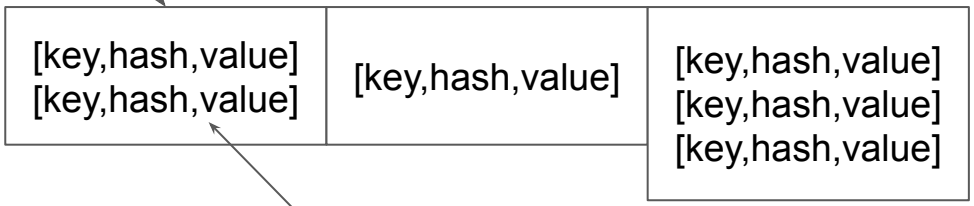


rb-tree

C++

Numpy + Tricks and Hack

list

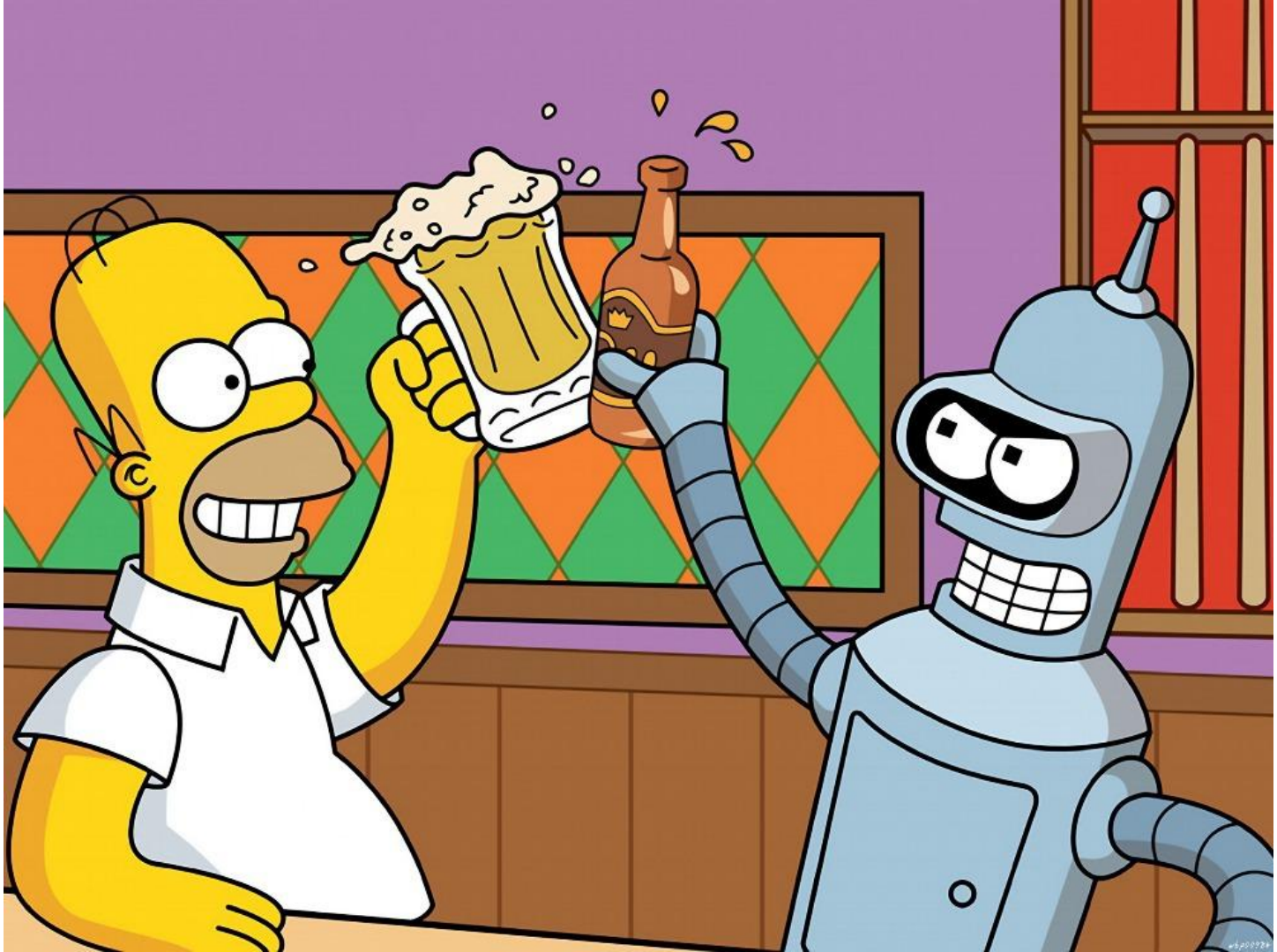


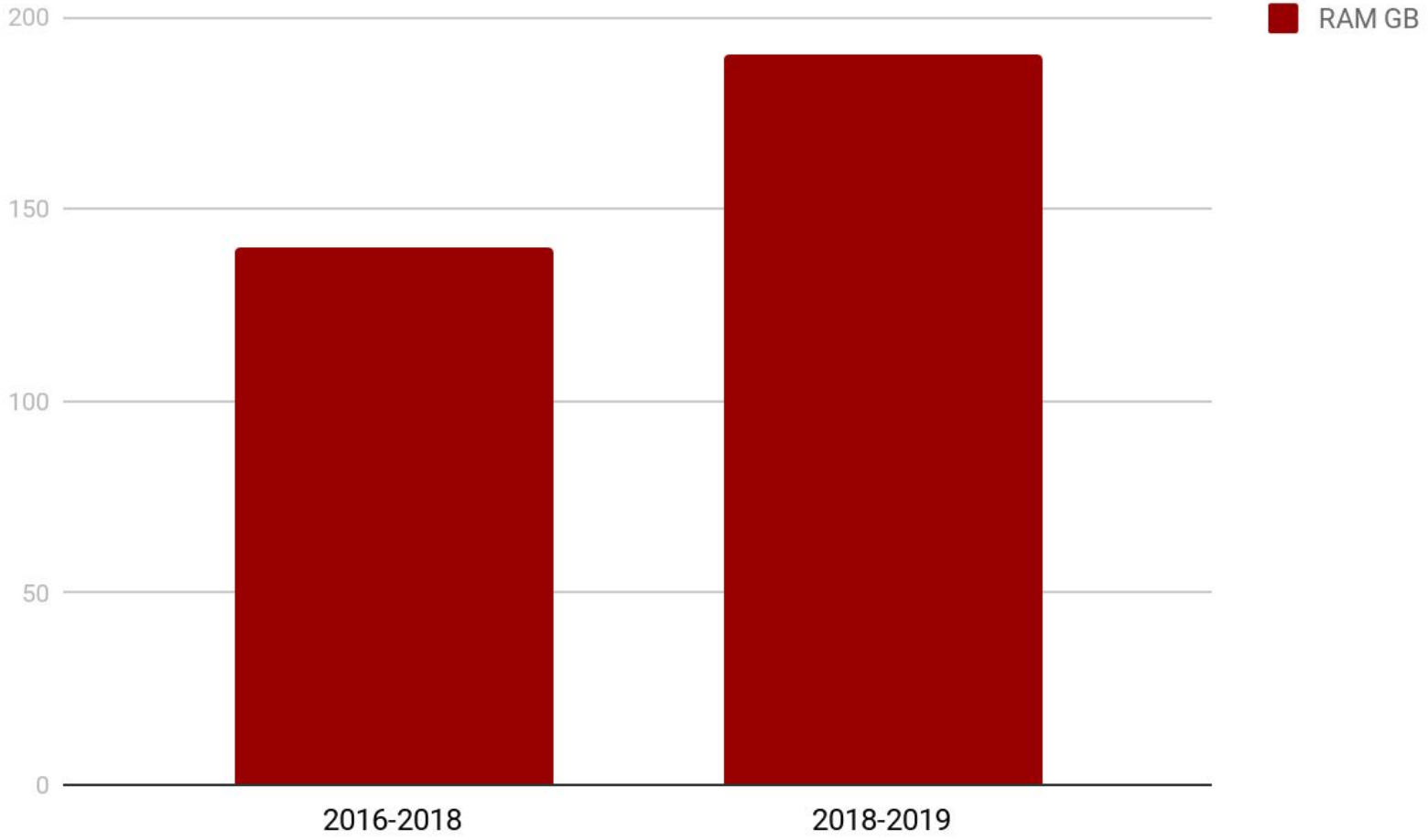
dict -> index : compressed list

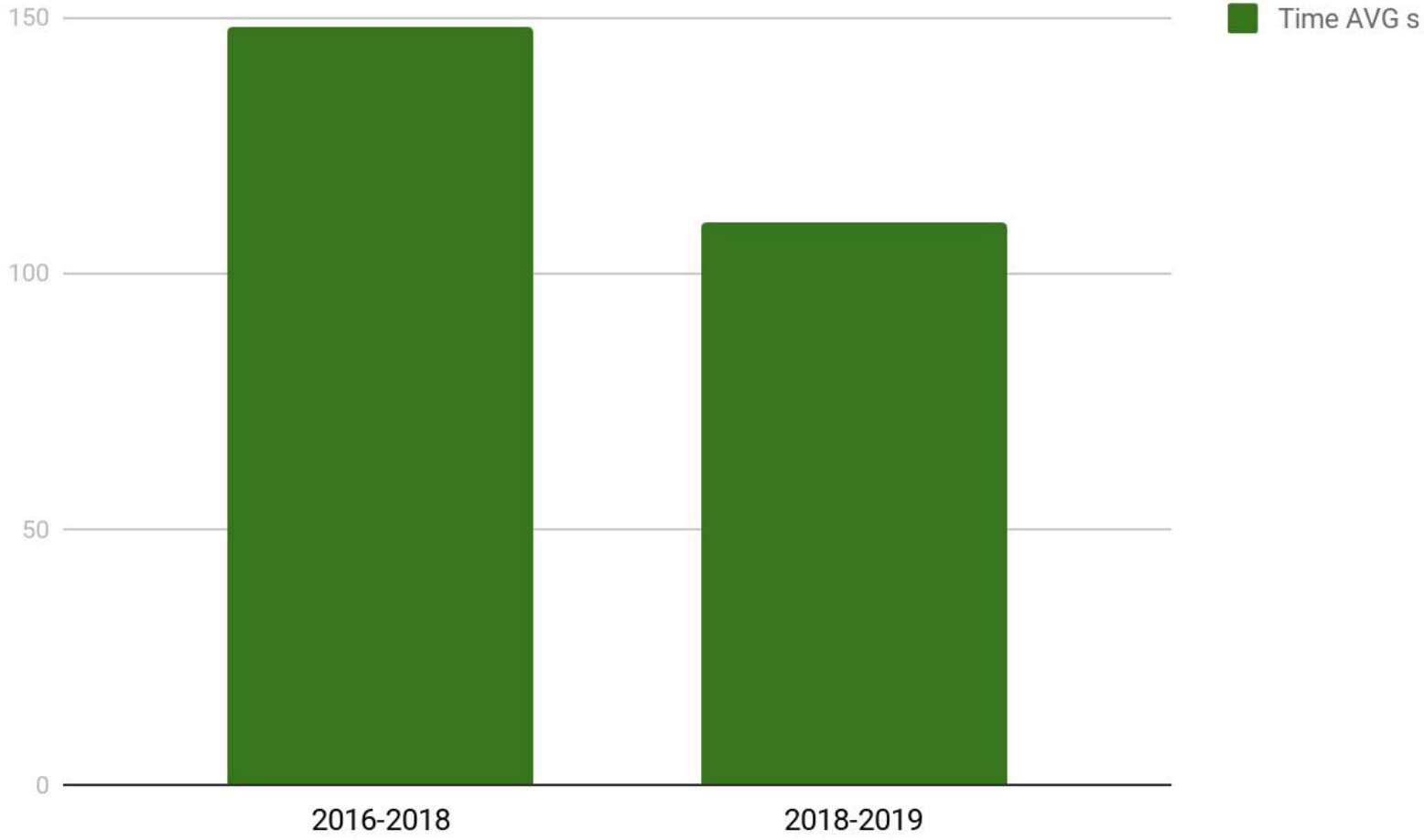
C++

Numpy + Tricks and Hack

2019-20xx




200
150
100
50
0
RAM GB
2016-2018
2018-2019


150
100
50
0
Time AVG s
2016-2018
2018-2019

Servise

Service
Recommendation*

http

kv-storage

logic

input
Binary Data.csv

Output
Append online
Binary Data

Servise

Service
Recommendation*

http

kv-storage

logic

input
Binary Data.csv

Output
Append online
Binary Data

Python Memory Tips and Tricks

Python Memory Tips and Tricks

- Gc Magic slow down

Python Memory Tips and Tricks

- Gc Magic slow down
  - Gc off

Python Memory Tips and Tricks

- Gc Magic slow down
  - Gc off
  - Gc manual

Python Memory Tips and Tricks

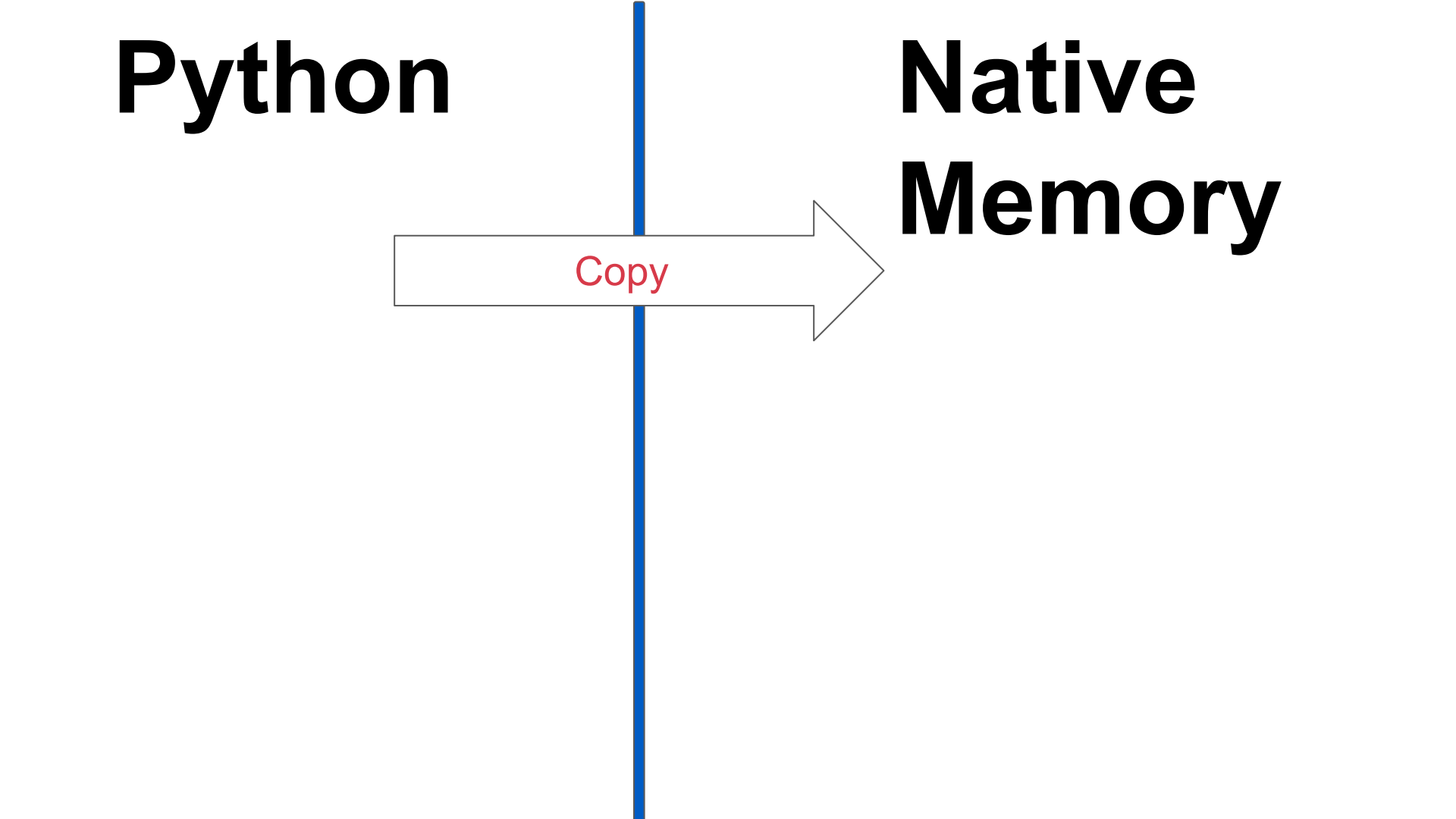
- Gc Magic slow down
  - Gc off
  - Gc manual
- __slots__

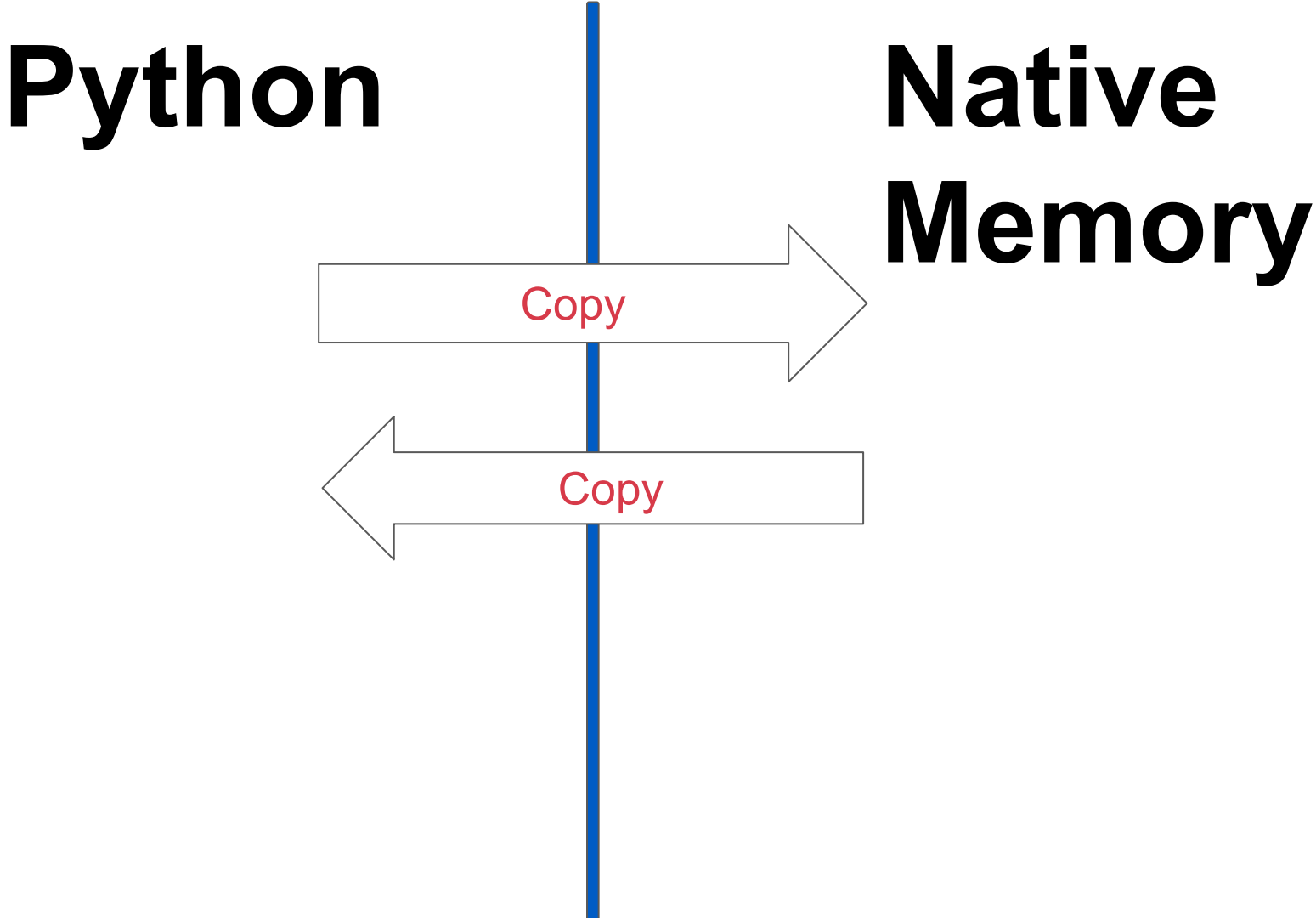
Python Memory Tips and Tricks

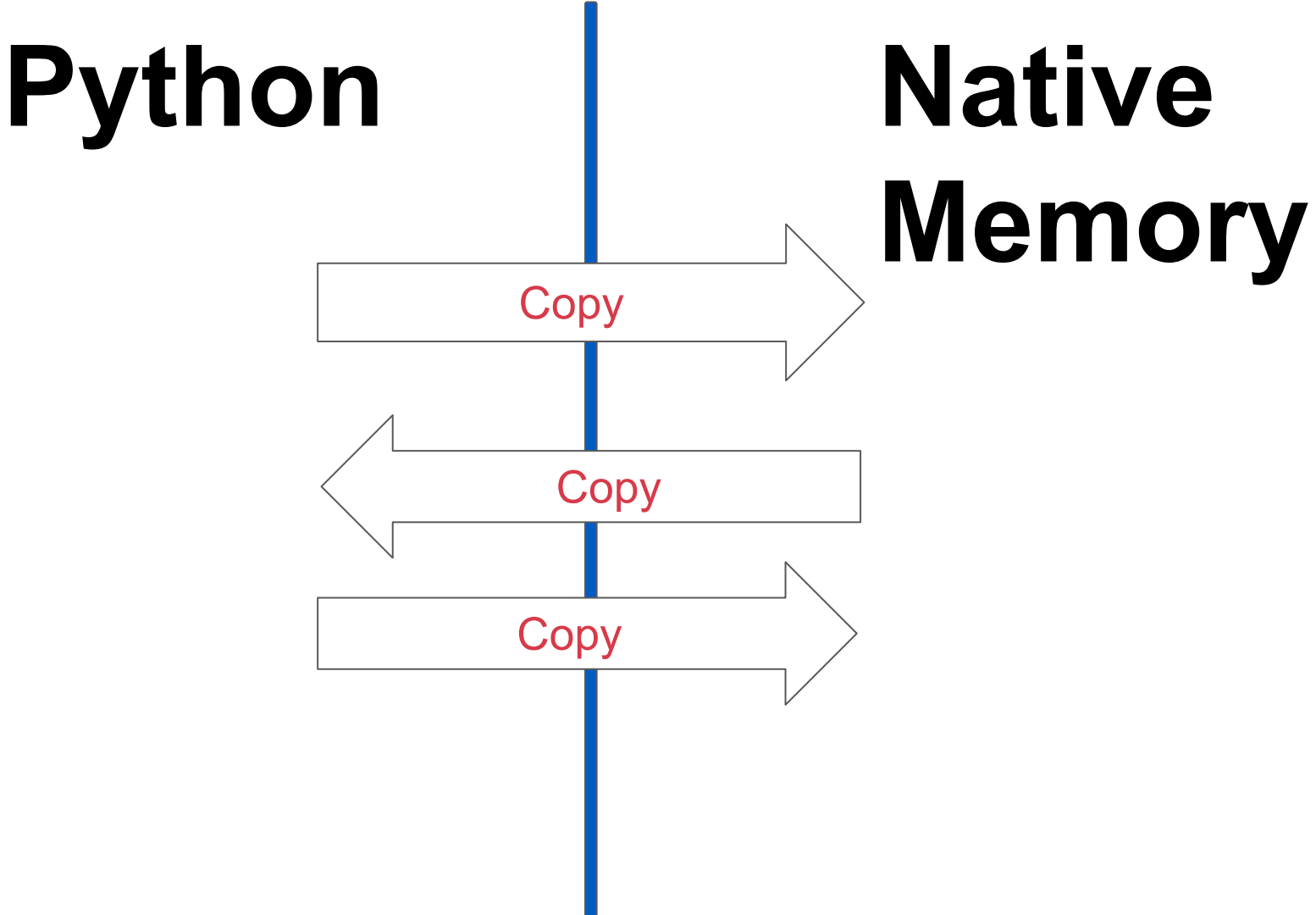
- Gc Magic slow down
  - Gc off
  - Gc manual
- __slots__
- Jemalloc

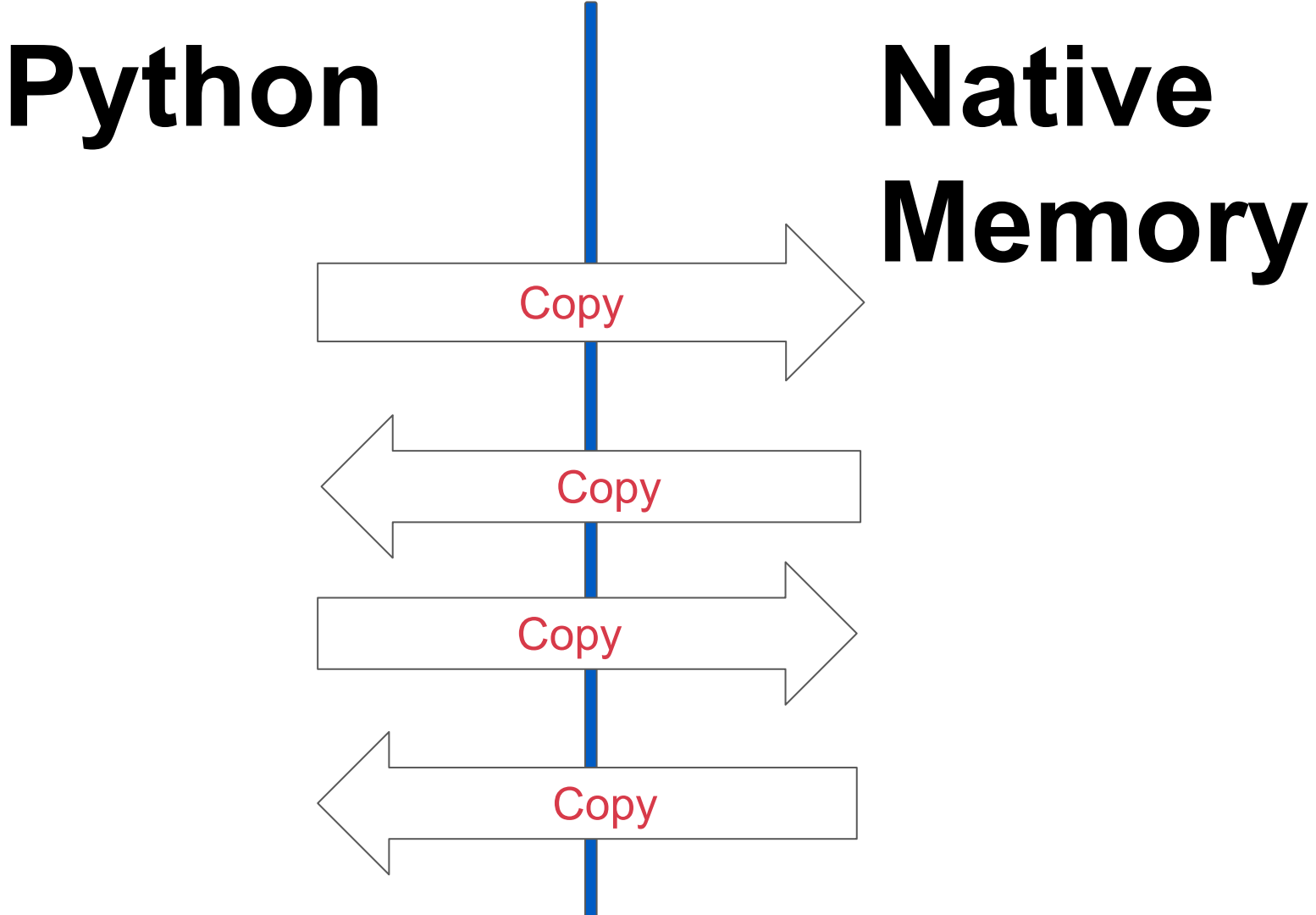
Python

Native Memory


Python
Native Memory
Copy


Python
Native Memory
Copy
Copy

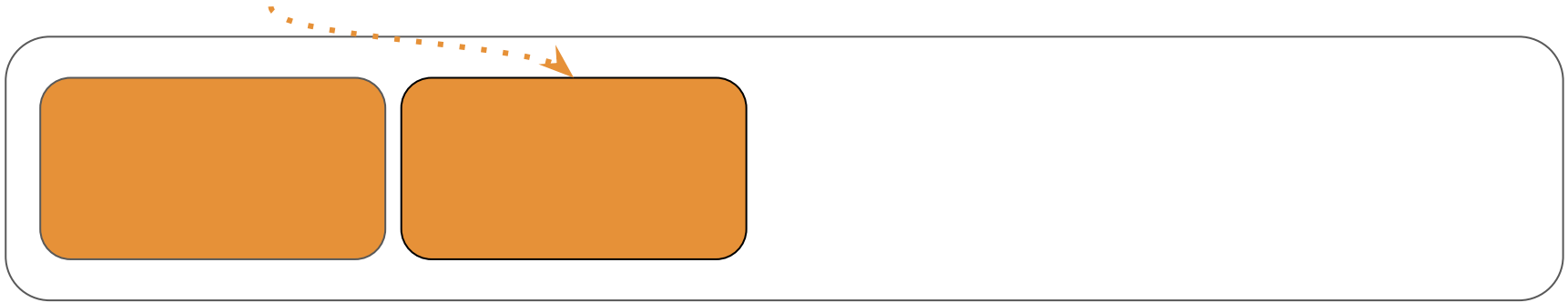

Python
Native Memory
Copy
Copy
Copy


Python
Native Memory
Copy
Copy
Copy
Copy

off-heap*

Python Process

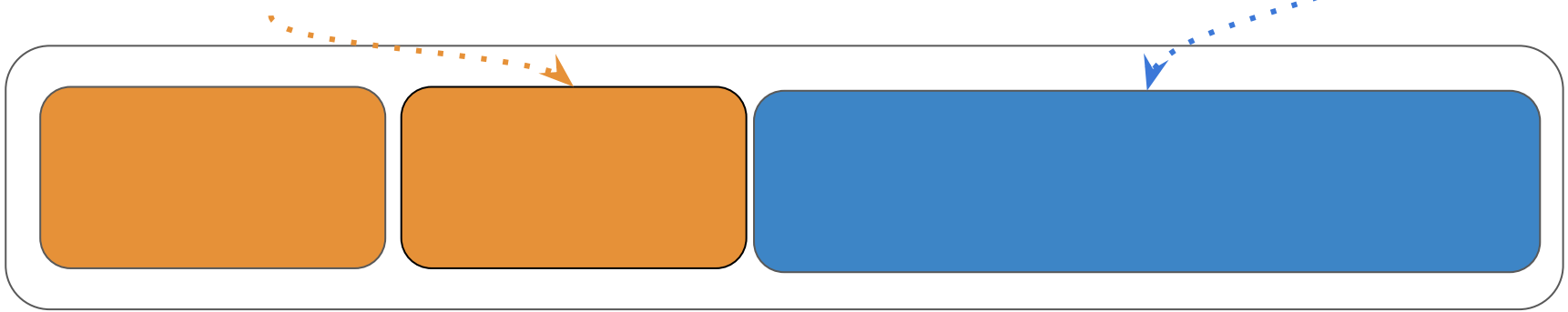
On-Heap
Managed by Gc



Python Process

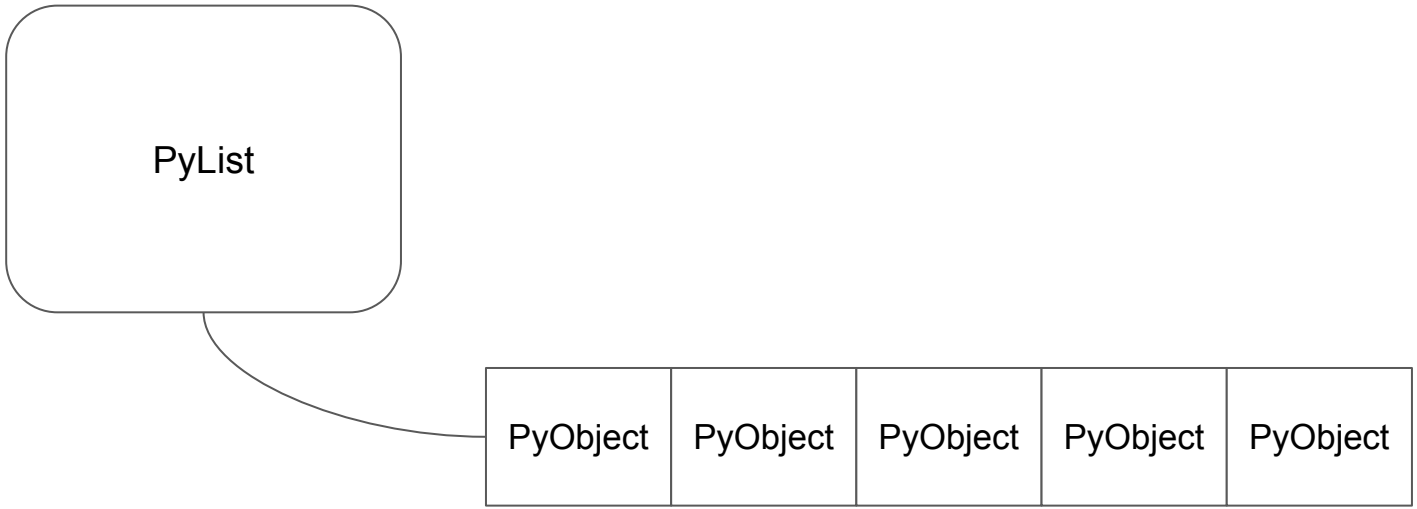
Off-Heap
Not managed by GC

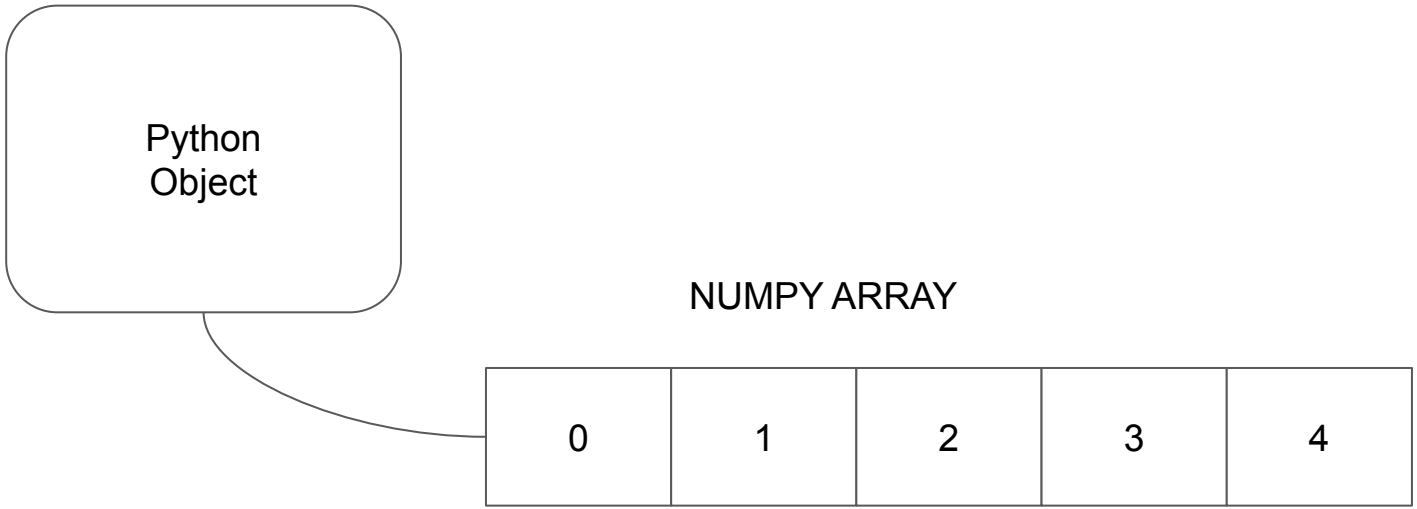
On-Heap
Managed by Gc



Python Process

NUMPY


PyList
PyObject
PyObject
PyObject
PyObject
PyObject


Python
Object
NUMPY ARRAY
0
1
2
3
4

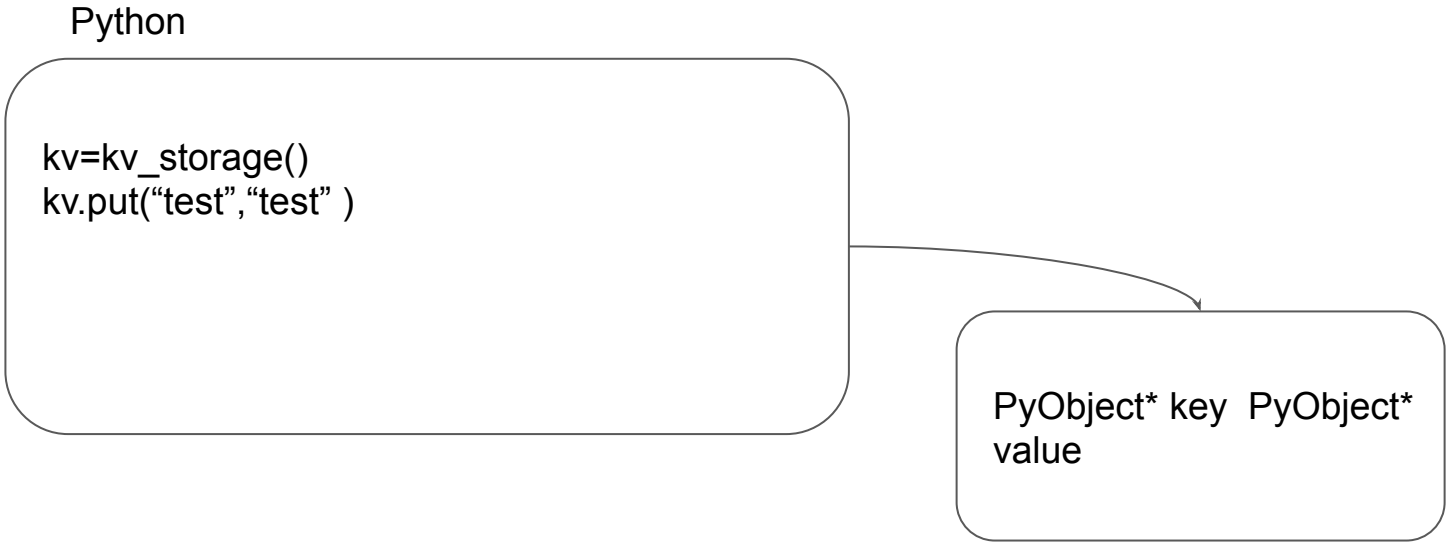
PEP 3118 -- Revising the buffer protocol

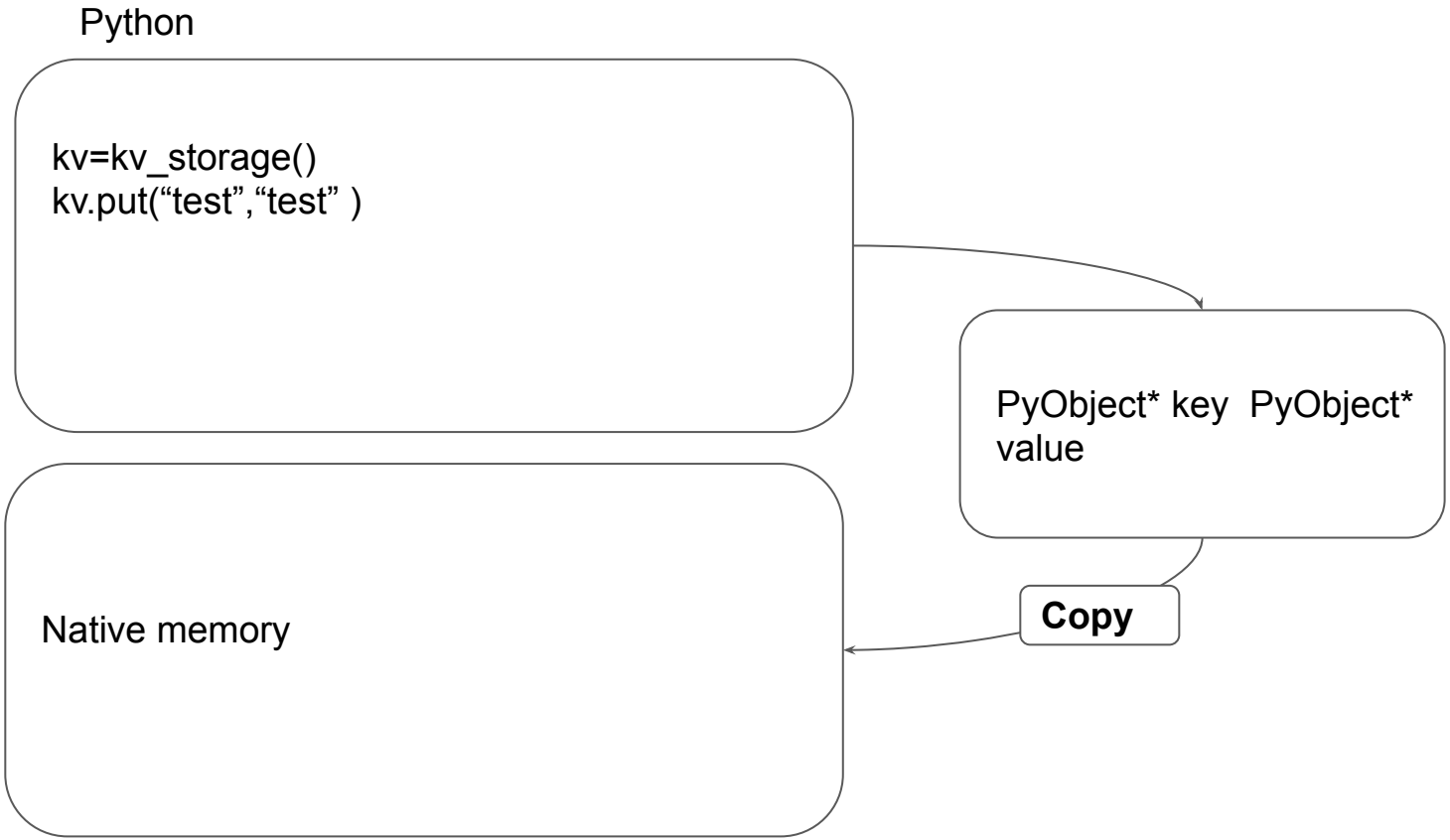
PEP:	3118
Title:	Revising the buffer protocol
Author:	Travis Oliphant <oliphant at ee.byu.edu>, Carl Banks <pythondev at aerojockey.com>
Status:	Final
Type:	Standards Track
Created:	28-Aug-2006
Python-Version:	3000
Post-History:	

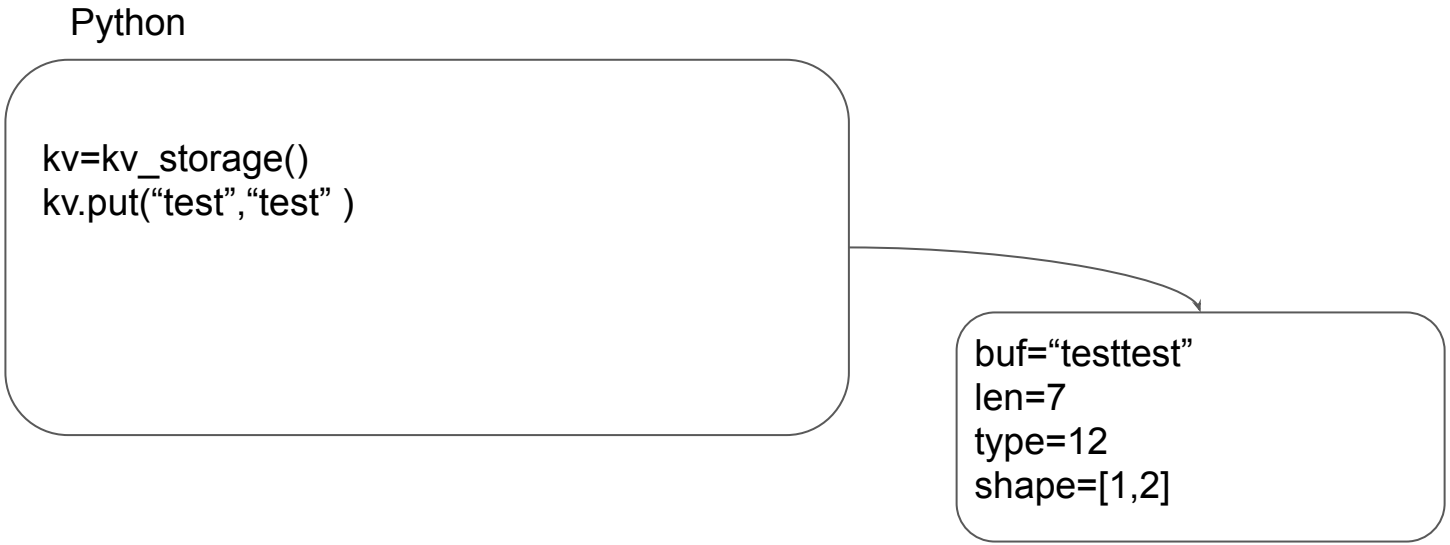
```
struct bufferinfo {
    void *buf;
    Py_ssize_t len;
    int readonly;
    const char *format;
    int ndim;
    Py_ssize_t *shape;
    Py_ssize_t *strides;
    Py_ssize_t *suboffsets;
    Py_ssize_t itemsize;
    void *internal;
} Py_buffer;
```

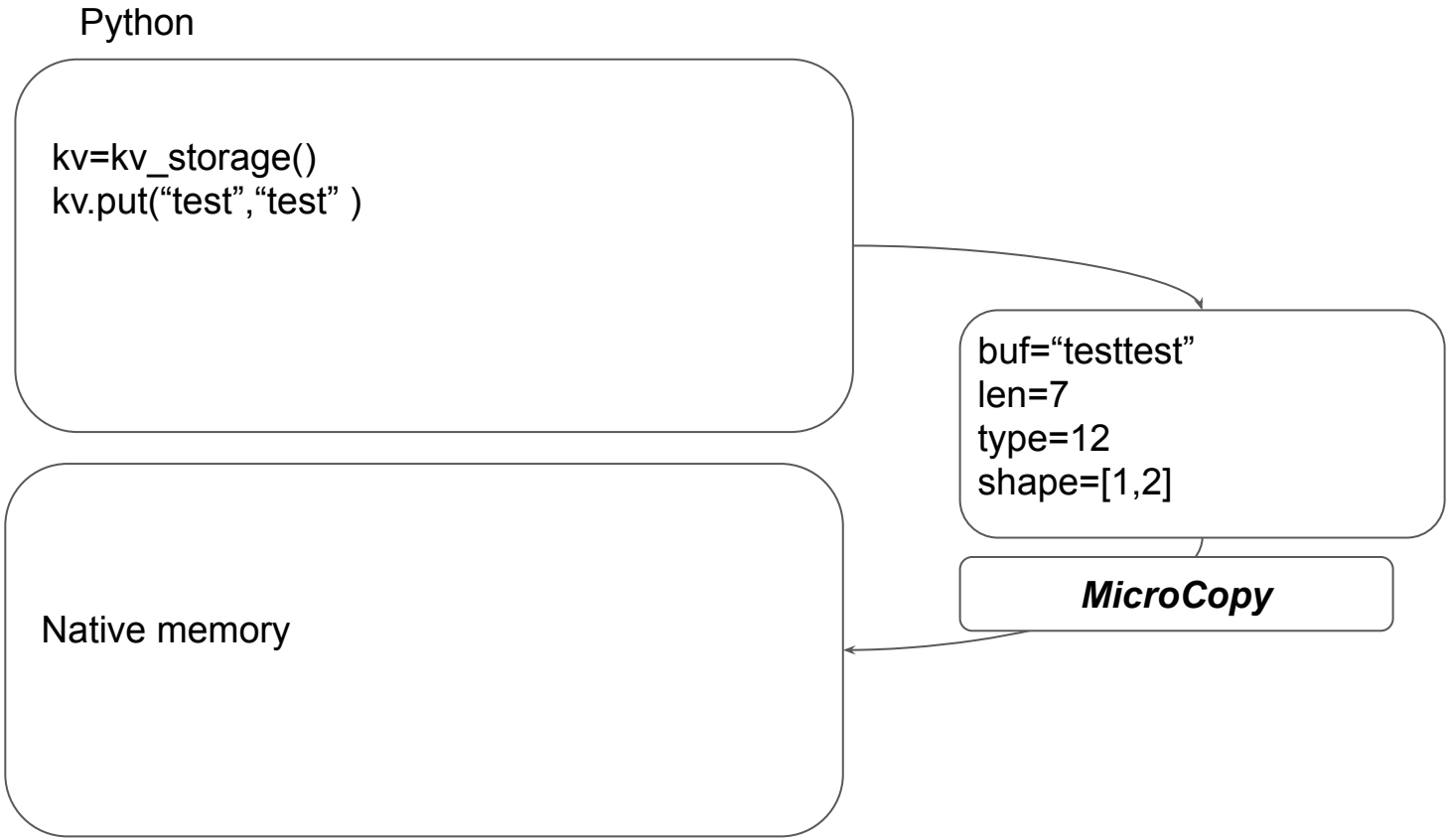
Python

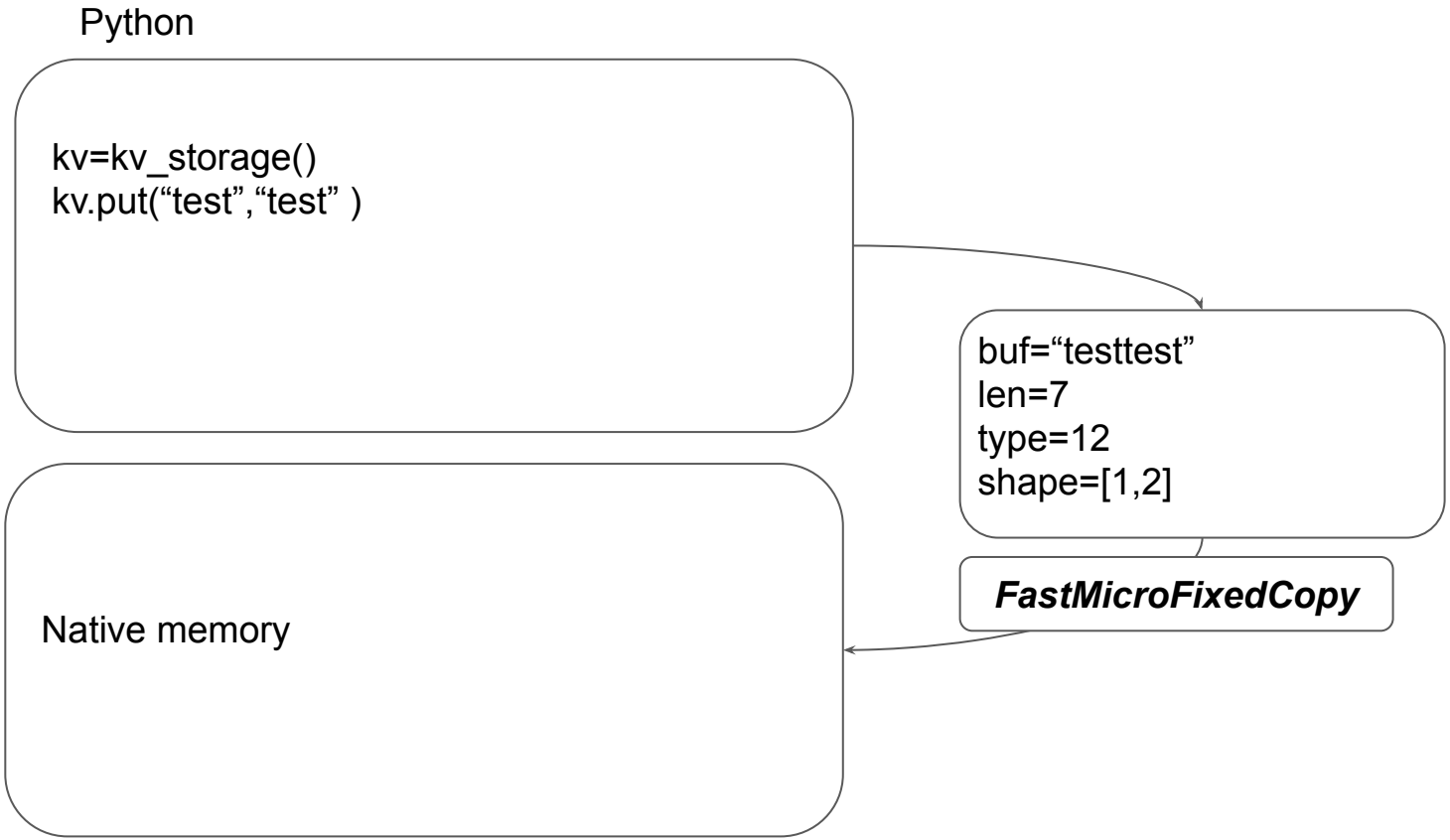
```
kv=kv_storage()
kv.put(“test”,“test” )
```

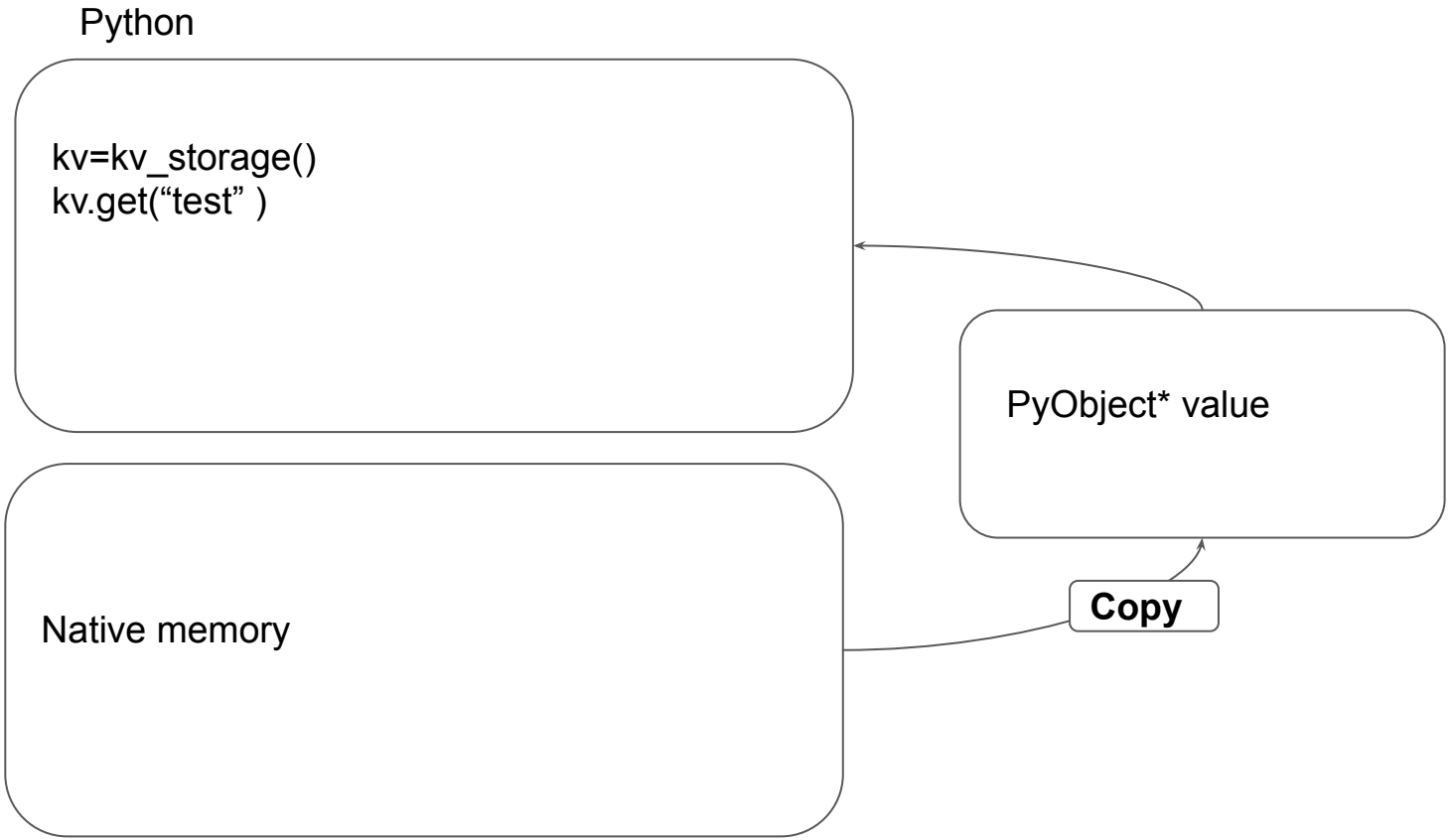

Python
kv=kv_storage()
kv.put(“test”,“test”)
PyObject* key PyObject* value

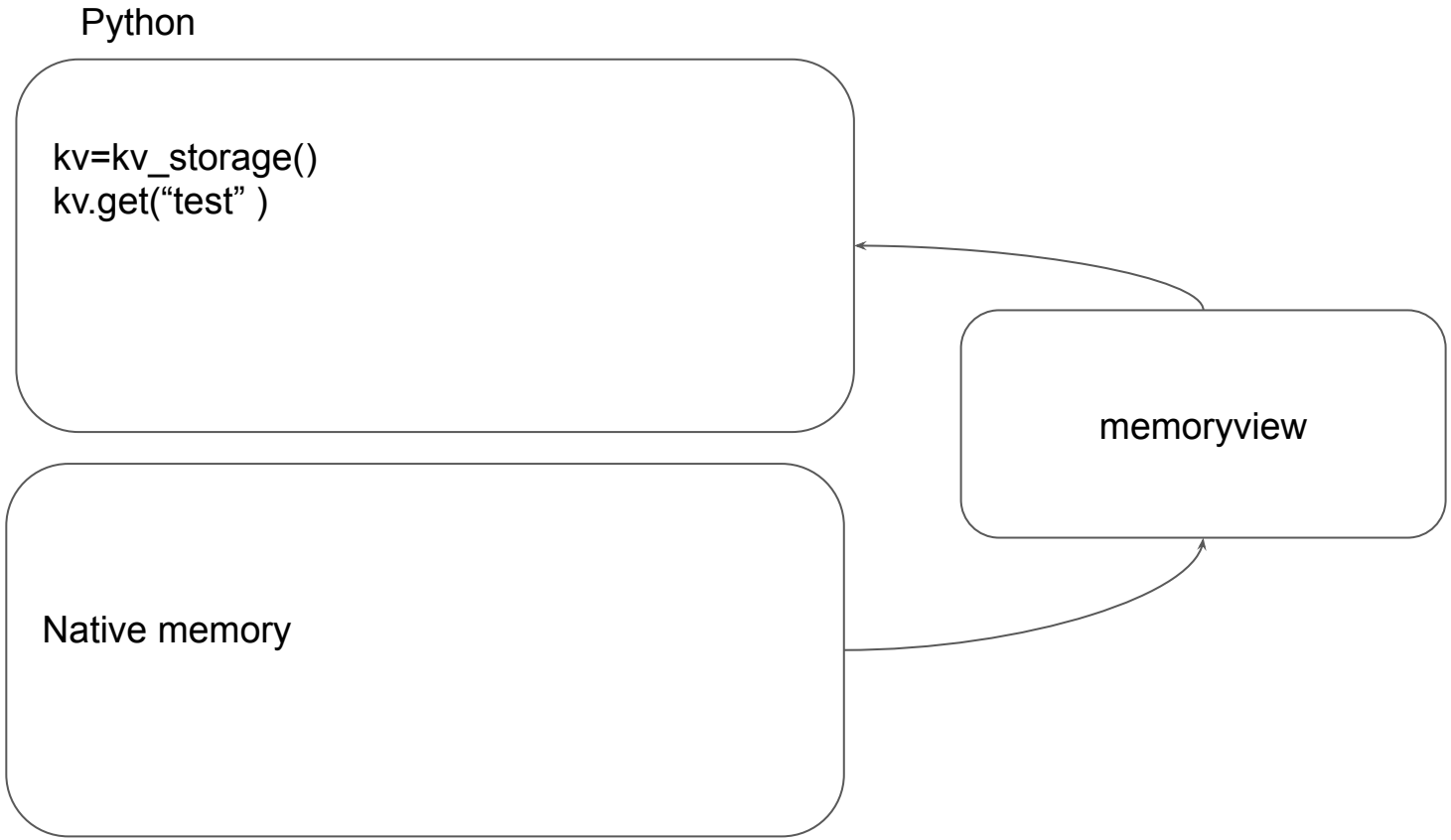

Python
kv=kv_storage()
kv.put(“test”,“test”)
PyObject* key PyObject* value
Copy
Native memory

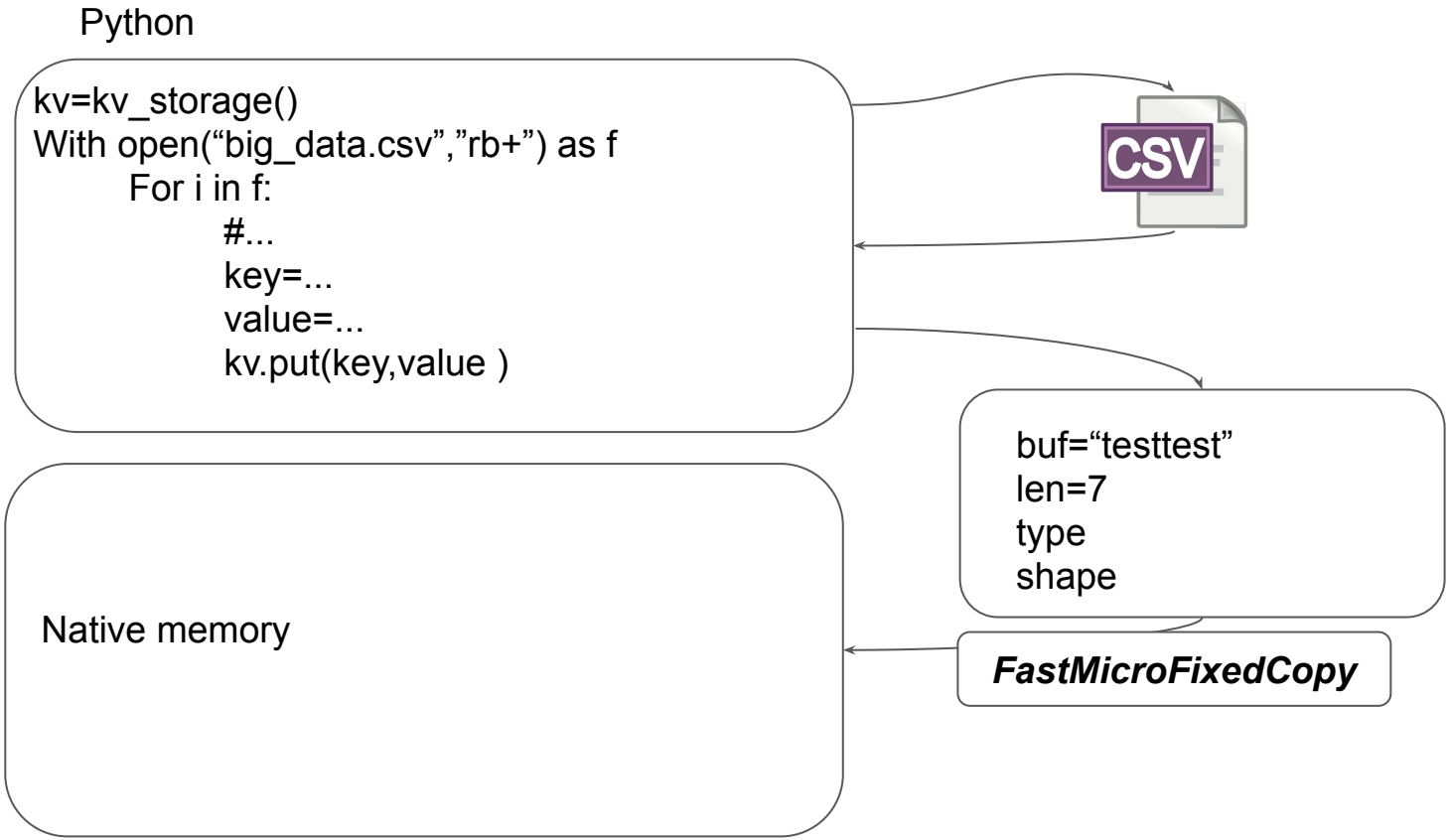

Python
kv=kv_storage()
kv.put(“test”,“test”)
buf=“testtest”
len=7
type=12
shape=[1,2]

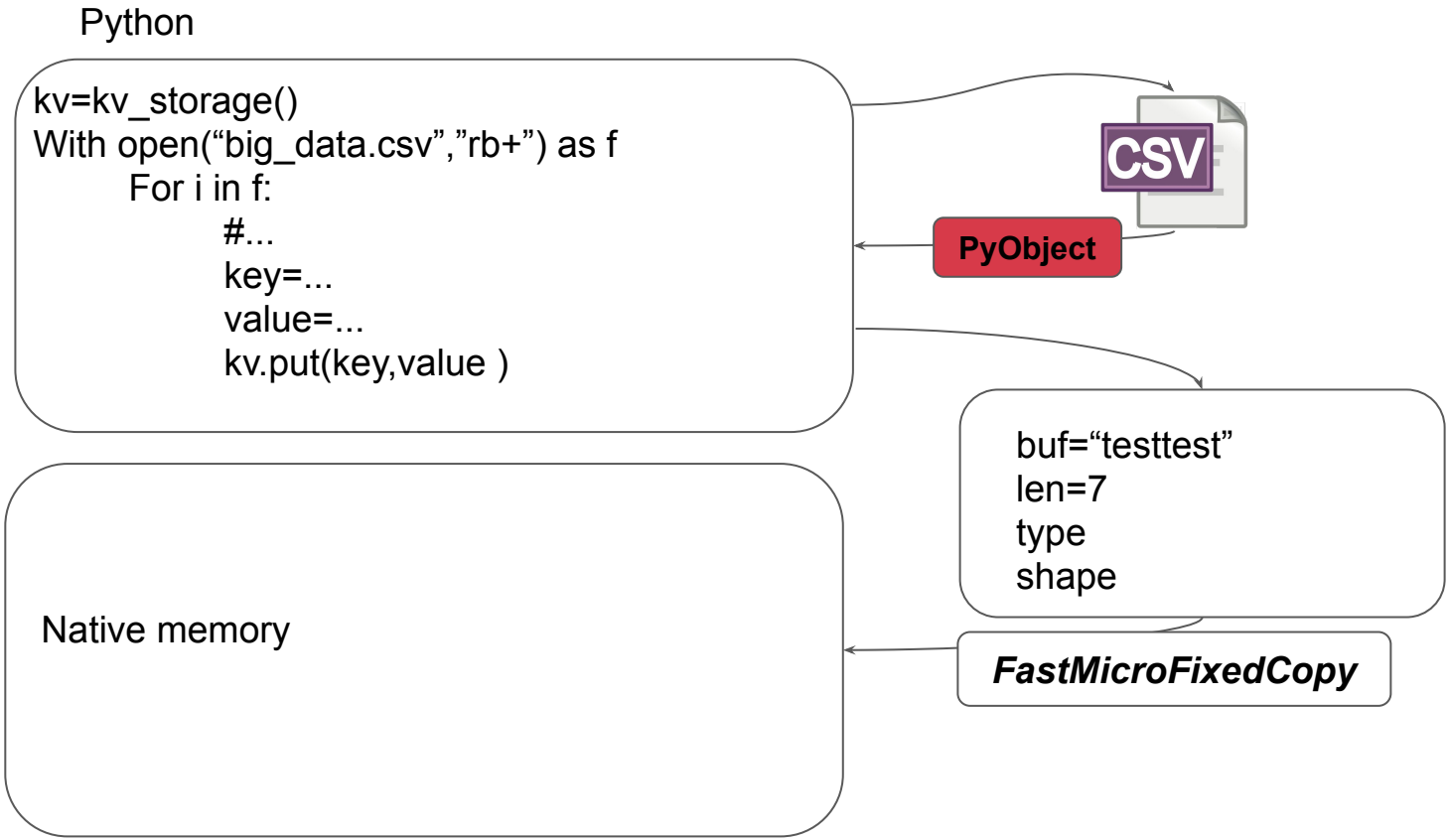

Python
kv=kv_storage()
kv.put(“test”,“test”)
buf=“testtest”
len=7
type=12
shape=[1,2]
MicroCopy
Native memory

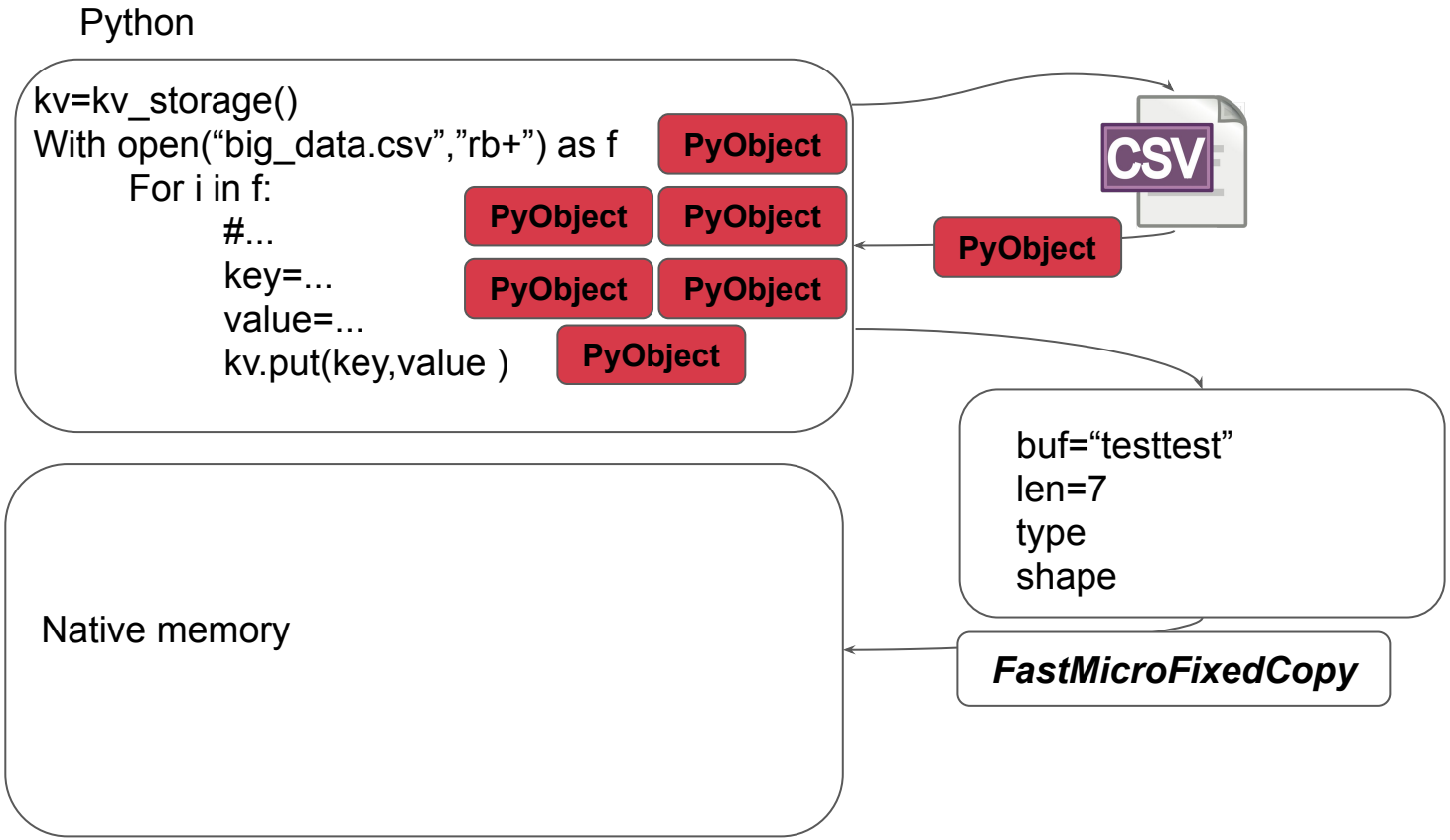

Python
kv=kv_storage()
kv.put(“test”,“test”)
buf=“testtest”
len=7
type=12
shape=[1,2]
FastMicroFixedCopy
Native memory


Python
kv=kv_storage()
kv.get(“test”)
PyObject* value
Native memory
Copy


Python
kv=kv_storage()
kv.get(“test”)
memoryview
Native memory


Python
kv=kv_storage()
With open(“big_data.csv”,”rb+”) as f
For i in f:
#...
key=...
value=...
kv.put(key,value)
CSV
buf=“testtest”
len=7
type
shape
FastMicroFixedCopy
Native memory


Python
kv=kv_storage()
With open(“big_data.csv”,”rb+”) as f
For i in f:
#...
key=...
value=...
kv.put(key,value)
CSV
PyObject
buf=“testtest”
len=7
type
shape
FastMicroFixedCopy
Native memory


Python
kv=kv_storage()
With open(“big_data.csv”,”rb+”) as f
For i in f:
#...
key=...
value=...
kv.put(key,value)
PyObject
PyObject
PyObject
PyObject
PyObject
PyObject
CSV
buf=“testtest”
len=7
type
shape
FastMicroFixedCopy
Native memory

Module

- pyThread
- Sub-interpretation*
- pep 554*
- dask*,airflow* re-load

Module non thread-safe

Module non thread-safe

- pyThread*

Module non thread-safe

- pyThread*
- Sub-interpretation*

Module non thread-safe

- pyThread*
- Sub-interpretation*
- pep 554*

Module non thread-safe

- pyThread*
- Sub-interpretation*
- pep 554*
- dask*,airflow*

Module non thread-safe

- pyThread*
- Sub-interpretation*
- pep 554*
- dask*,airflow* re-load

A Embedded python will save us.

runner

Embedded python

from kv_storage import Storage

```
//....
py::scoped_interpreter python_;
py::module kv_storage;
```

```
//....
py::scoped_interpreter python_;
py::module kv_storage;
//....
char init_script[] =
       "import sys, os
       from importlib import import_module
       sys.modules['kv_storage'] = kv_storage";
//....
py::exec(init_script, py::globals(), py::dict("kv_storage"_a = kv_storage));
//....
```

```
//....
py::scoped_interpreter python_;
py::module kv_storage;
//....
char init_script[] =
       "import sys, os
       from importlib import import_module
       sys.modules['kv_storage'] = kv_storage";
//....
py::exec(init_script, py::globals(), py::dict("kv_storage"_a = kv_storage));
//....
char load_script[] =
       "import sys, os
       from importlib import import_module
       sys.path.insert(0, os.path.dirname(path))
       module_name, _ = os.path.splitext(path)
       import_module(os.path.basename(module_name))";
//....
py::exec(load_script, py::globals(), py::dict("path"_a = script_path_.string()));
//....
```

runner

Embedded python

```
from kv_storage import Storage, Cache
...
```

```
//....
py::scoped_interpreter python_;
py::module kv_storage;

char init_script[] =
       "import sys, os
       from importlib import import_module
       sys.modules['kv_storage'] = kv_storage";
//....
py::exec(init_script, py::globals(), py::dict("kv_storage"_a = kv_storage));
//....
char load_script[] =
       "import sys, os
       from importlib import import_module
       sys.path.insert(0, os.path.dirname(path))
       module_name, _ = os.path.splitext(path)
       import_module(os.path.basename(module_name))";
//....
py::exec(load_script, py::globals(), py::dict("path"_a = script_path_.string()));
//....
```

```
//....
py::scoped_interpreter python_;
py::module kv_storage;

char init_script[] =
       "import sys, os
       from importlib import import_module
       sys.modules['kv_storage'] = kv_storage";
//....
py::exec(init_script, py::globals(), py::dict("kv_storage"_a = kv_storage));
//....
char load_script[] =
       "import sys, os
       from importlib import import_module
       sys.path.insert(0, os.path.dirname(path))
       module_name, _ = os.path.splitext(path)
       import_module(os.path.basename(module_name))";
std::vector<std::string> args <- copy int argc, char* argv[] / pep...
py::list tmp;
for (auto it = args.begin();it!=args.end();++it) {
       tmp.append(*it);
}
py::module::import("sys").add_object("argv", tmp);
py::exec(load_script, py::globals(), py::dict("path"_a = script_path_.string()));
//....
```

runner

Embedded python

```
from kv_storage import Storage, Cache
...
```

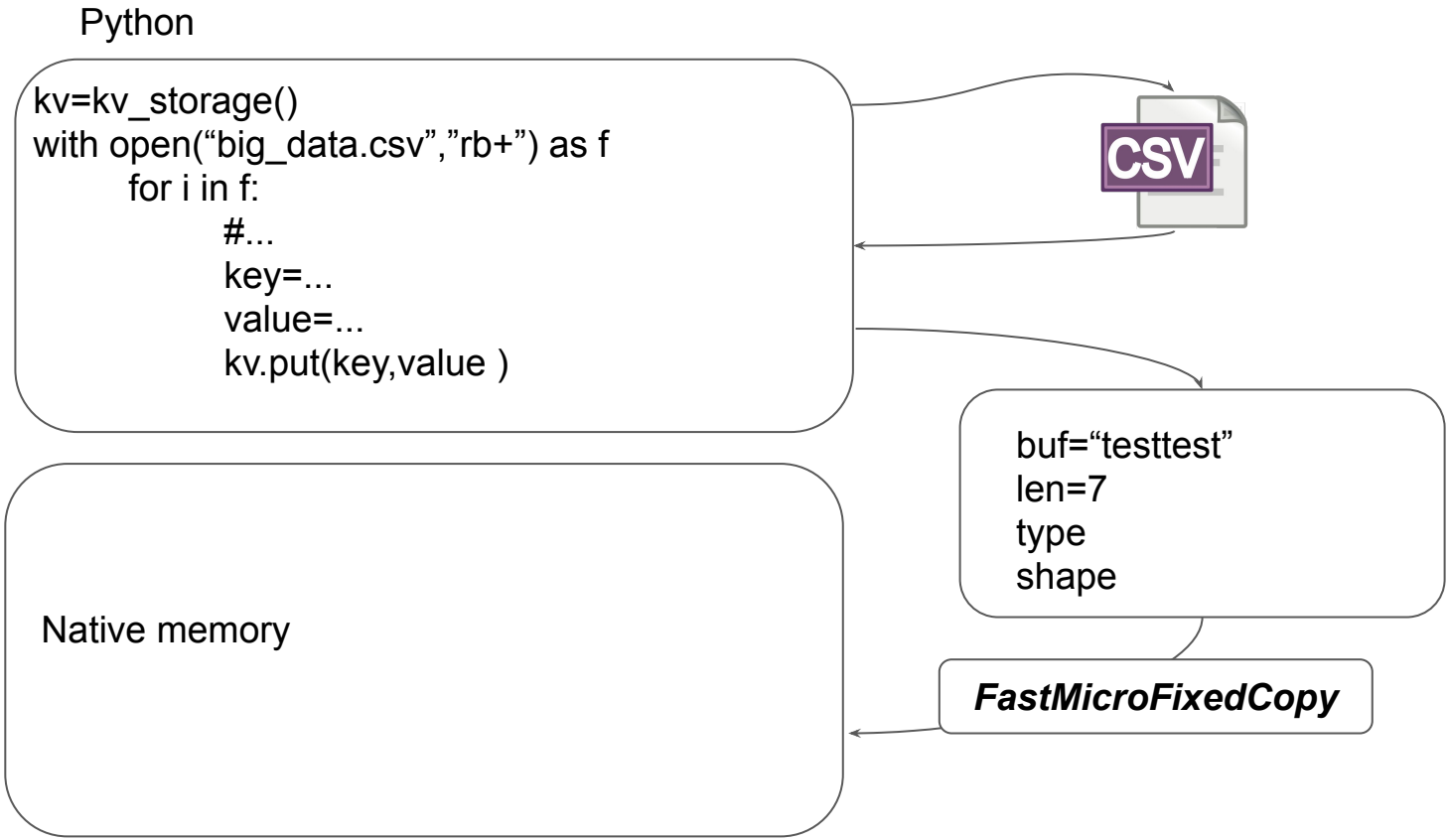
runner --native_configuration=cfg.ini

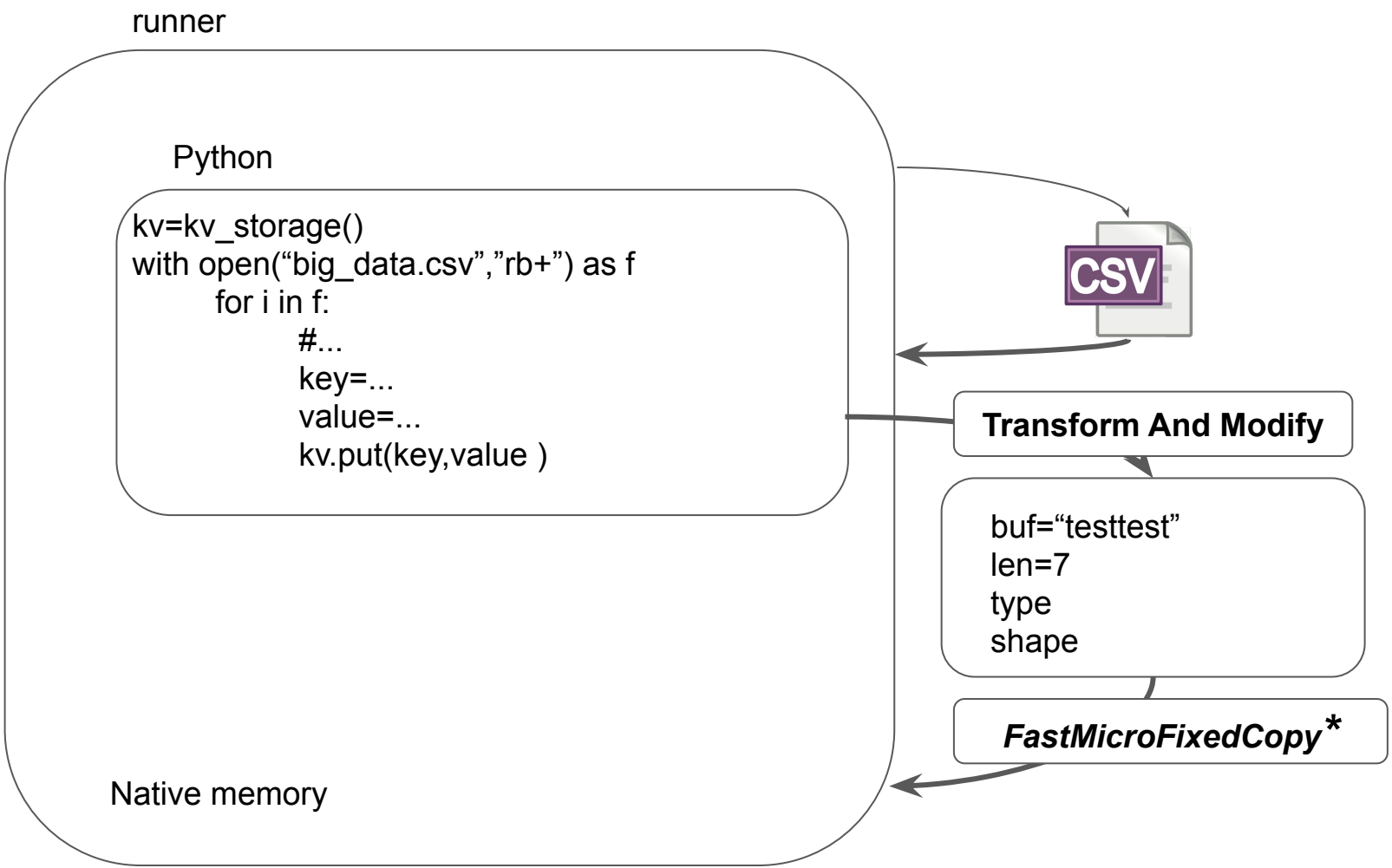
cfg.ini

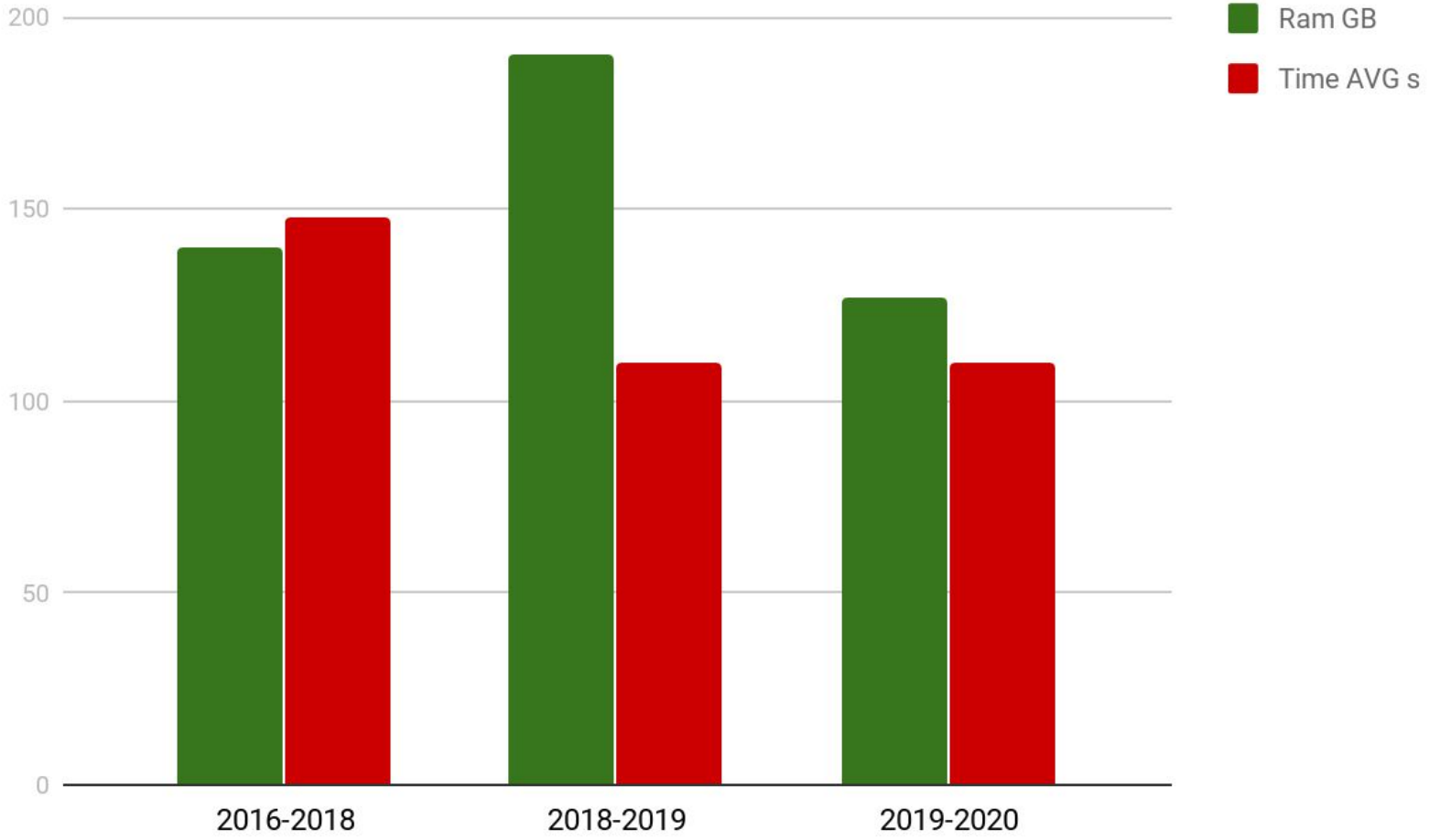
```
[kv_storage]
cache=””
csv=””
```

Embedded python

```
from kv_storage import Storage, Cache
...
```


Python
kv=kv_storage()
with open(“big_data.csv”,”rb+”) as f
for i in f:
#...
key=...
value=...
kv.put(key,value)
CSV
buf=“testtest”
len=7
type
shape
FastMicroFixedCopy
Native memory


runner
Python
kv=kv_storage()
with open(“big_data.csv”,”rb+”) as f
for i in f:
#...
key=...
value=...
kv.put(key,value)
CSV
Transform And Modify
buf=“testtest”
len=7
type
shape
FastMicroFixedCopy*
Native memory


200
150
100
50
0
2016-2018
2018-2019
2019-2020
Ram GB
Time AVG s

Useful right now !

Useful right now !

Python is very flexible

Useful right now !

Python is very flexible

Python as the Best glue

Modules are a good implementation tool …

Useful right now !

Python is very flexible

Python as the Best glue

Modules are a good implementation tool …

- Cython
- pybind11

Useful right now !

Python is very flexible

Python as the Best glue

Modules are a good implementation tool …

- Cython
- Pybind11

Embedded python

CYBER DUCK NINJA

cyberduckninja.com

github.com/cyberduckninja

Borgardt Alexander

aa.borgardt@yandex.ru

kotbegemot

@k0tb9g9m0t