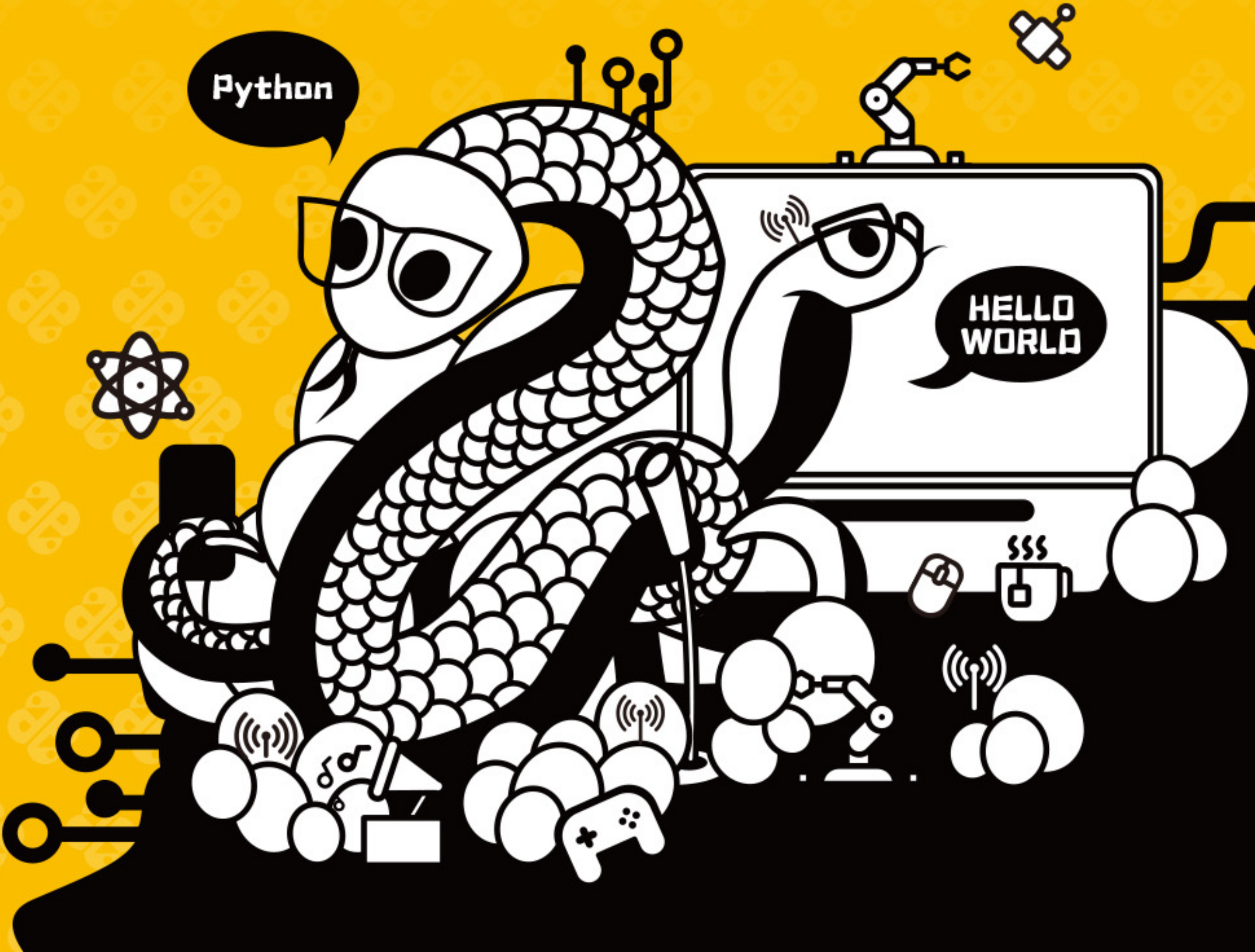
python™ × PYCHINA Python 中国社区

Python for Good

»»» PyCon China 2022

Python 启动加速探索及实践

主讲人： 严懿宸 – 阿里云



严懿宸

曾于 Oracle Labs 参与 GraalVM 开发

毕业后加入阿里云 – 编译器
目前负责 Python / Node.js 的 Runtime 优化

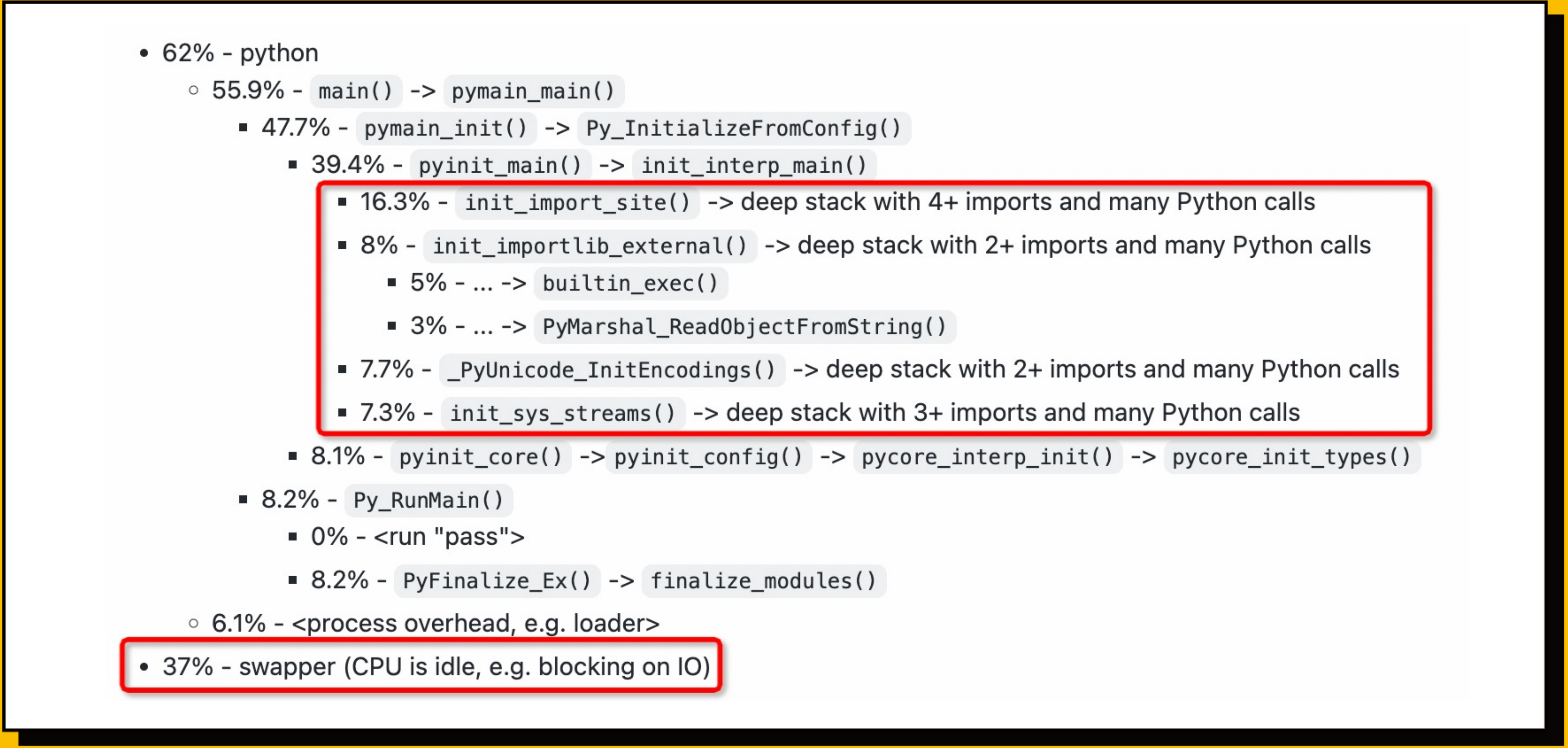
Content

- Python 启动速度简析
- PyCDS 设计与实现
- 更多讨论

Python startup time

- 62% - python
  - 55.9% - `main()` -> `pymain_main()`
    - 47.7% - `pymain_init()` -> `Py_InitializeFromConfig()`
      - 39.4% - `pyinit_main()` -> `init_interp_main()`
        - 16.3% - `init_import_site()` -> deep stack with 4+ imports and many Python calls
        - 8% - `init_importlib_external()` -> deep stack with 2+ imports and many Python calls
          - 5% - ... -> `builtin_exec()`
          - 3% - ... -> `PyMarshal_ReadObjectFromString()`
        - 7.7% - `_PyUnicode_InitEncodings()` -> deep stack with 2+ imports and many Python calls
        - 7.3% - `init_sys_streams()` -> deep stack with 3+ imports and many Python calls
      - 8.1% - `pyinit_core()` -> `pyinit_config()` -> `pycore_interp_init()` -> `pycore_init_types()`
    - 8.2% - `Py_RunMain()`
      - 0% - <run "pass">
      - 8.2% - `PyFinalize_Ex()` -> `finalize_modules()`
  - 6.1% - <process overhead, e.g. loader>
- 37% - swapper (CPU is idle, e.g. blocking on IO)

Python startup time

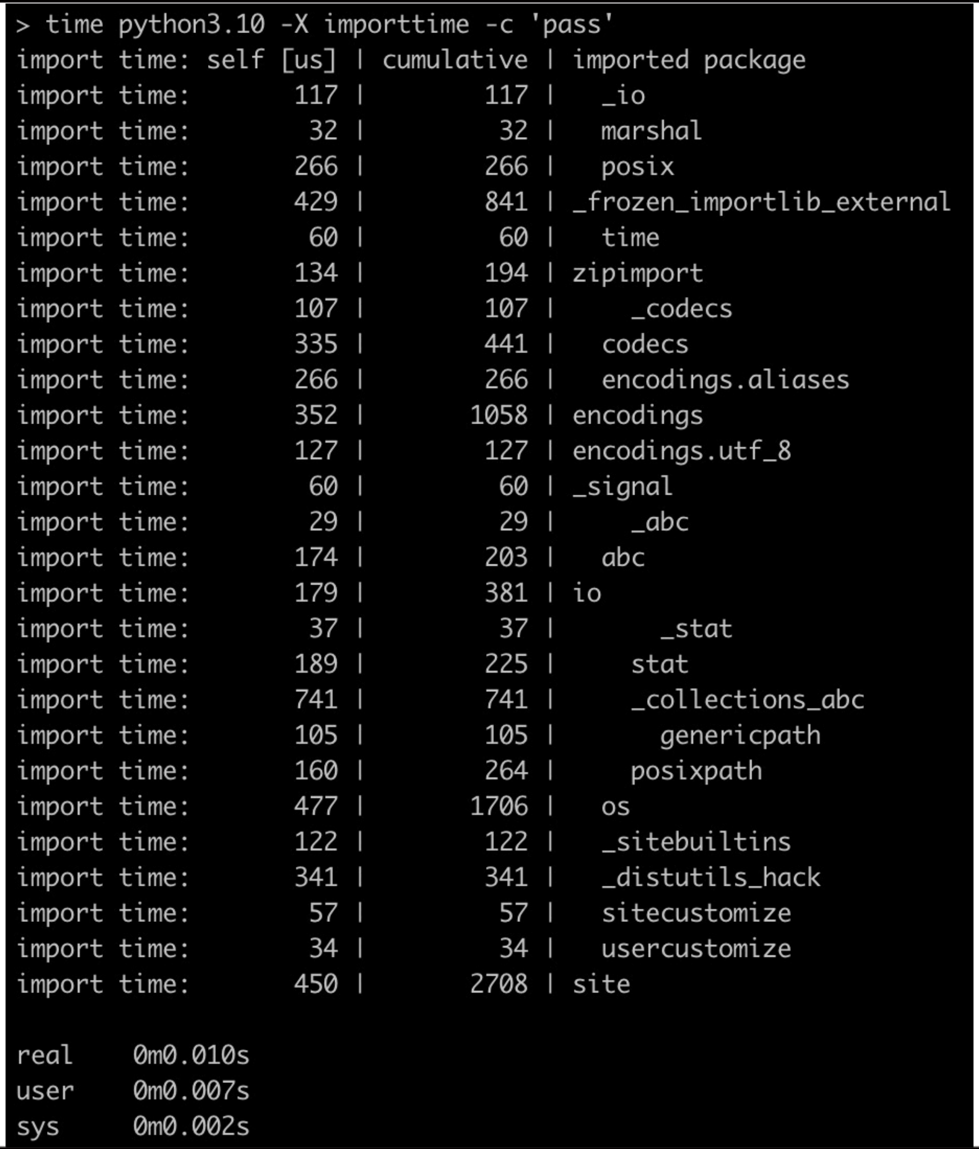


- 62% - python
  - 55.9% - `main()` -> `pymain_main()`
    - 47.7% - `pymain_init()` -> `Py_InitializeFromConfig()`
      - 39.4% - `pyinit_main()` -> `init_interp_main()`
        - 16.3% - `init_import_site()` -> deep stack with 4+ imports and many Python calls
        - 8% - `init_importlib_external()` -> deep stack with 2+ imports and many Python calls
          - 5% - ... -> `builtin_exec()`
          - 3% - ... -> `PyMarshal_ReadObjectFromString()`
        - 7.7% - `_PyUnicode_InitEncodings()` -> deep stack with 2+ imports and many Python calls
        - 7.3% - `init_sys_streams()` -> deep stack with 3+ imports and many Python calls
      - 8.1% - `pyinit_core()` -> `pyinit_config()` -> `pycore_interp_init()` -> `pycore_init_types()`
    - 8.2% - `Py_RunMain()`
      - 0% - <run "pass">
      - 8.2% - `PyFinalize_Ex()` -> `finalize_modules()`
  - 6.1% - <process overhead, e.g. loader>
- 37% - swapper (CPU is idle, e.g. blocking on IO)

Python startup time

```
Read __pycache__ -> Unmarshal -> Heap allocated code object -> Evaluate
```

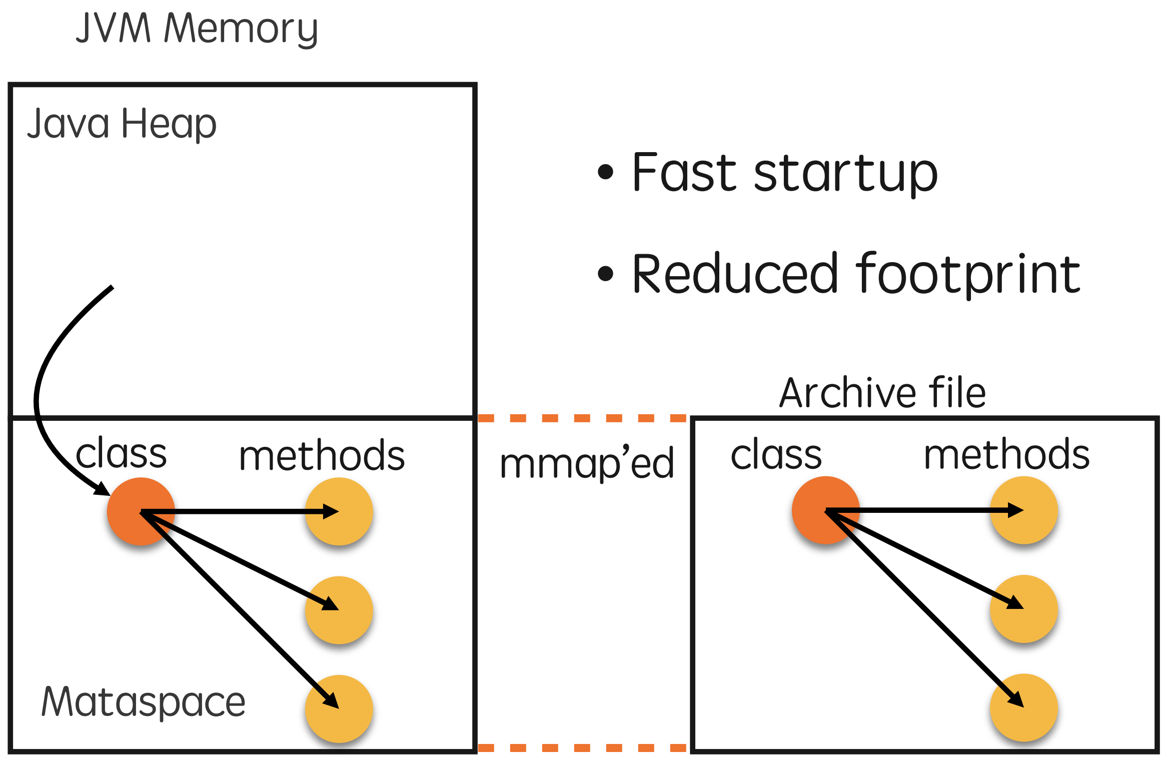
Python startup time (2)



```
> time python3.10 -X importtime -c 'pass'
import time: self [us] | cumulative | imported package
import time:       117 |        117 |   _io
import time:        32 |         32 |   marshal
import time:       266 |        266 |   posix
import time:       429 |        841 | _frozen_importlib_external
import time:        60 |         60 |   time
import time:       134 |        194 | zipimport
import time:       107 |        107 |     _codecs
import time:       335 |        441 |   codecs
import time:       266 |        266 |   encodings.aliases
import time:       352 |       1058 | encodings
import time:       127 |        127 | encodings.utf_8
import time:        60 |         60 | _signal
import time:        29 |         29 |     _abc
import time:       174 |        203 |   abc
import time:       179 |        381 | io
import time:        37 |         37 |       _stat
import time:       189 |        225 |     stat
import time:       741 |        741 |     _collections_abc
import time:       105 |        105 |       genericpath
import time:       160 |        264 |     posixpath
import time:       477 |       1706 |   os
import time:       122 |        122 |   _sitebuiltins
import time:       341 |        341 |   _distutils_hack
import time:        57 |         57 |   sitecustomize
import time:        34 |         34 |   usercustomize
import time:       450 |       2708 | site

real	0m0.010s
user	0m0.007s
sys	0m0.002s
```

AppCDS for Java



- Fast startup
- Reduced footprint

- CDS enabled: pointed to file-mapped memory
- CDS disabled: parsed from class files

History

- Commercial feature in Oracle JDK 8/9
- Open source since OpenJDK 10 (JEP 310, 2018)
- Default CDS Archives since OpenJDK 12 (JEP 341, 2019)

Adoption

- Alibaba Cloud SAE using AppCDS in production env
    - 30% startup time reduction
- OpenJDK 12+ Default CDS

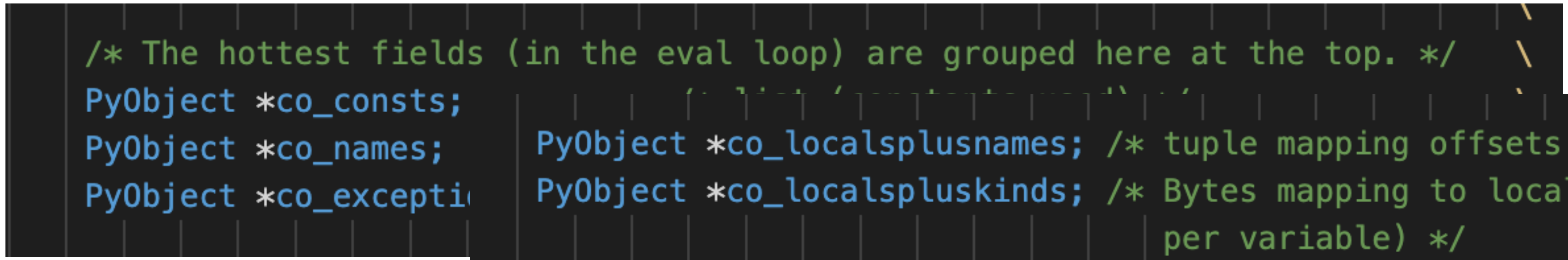
Methodology

```
def orig_import_module(name):  # *
    # find spec based on file name
    spec = finder.find_spec(name)
    # compilation from py files
    # or unmarshal pyc files
    code = spec.get_code(name)
    # exec code
    module = exec(code, {})
```

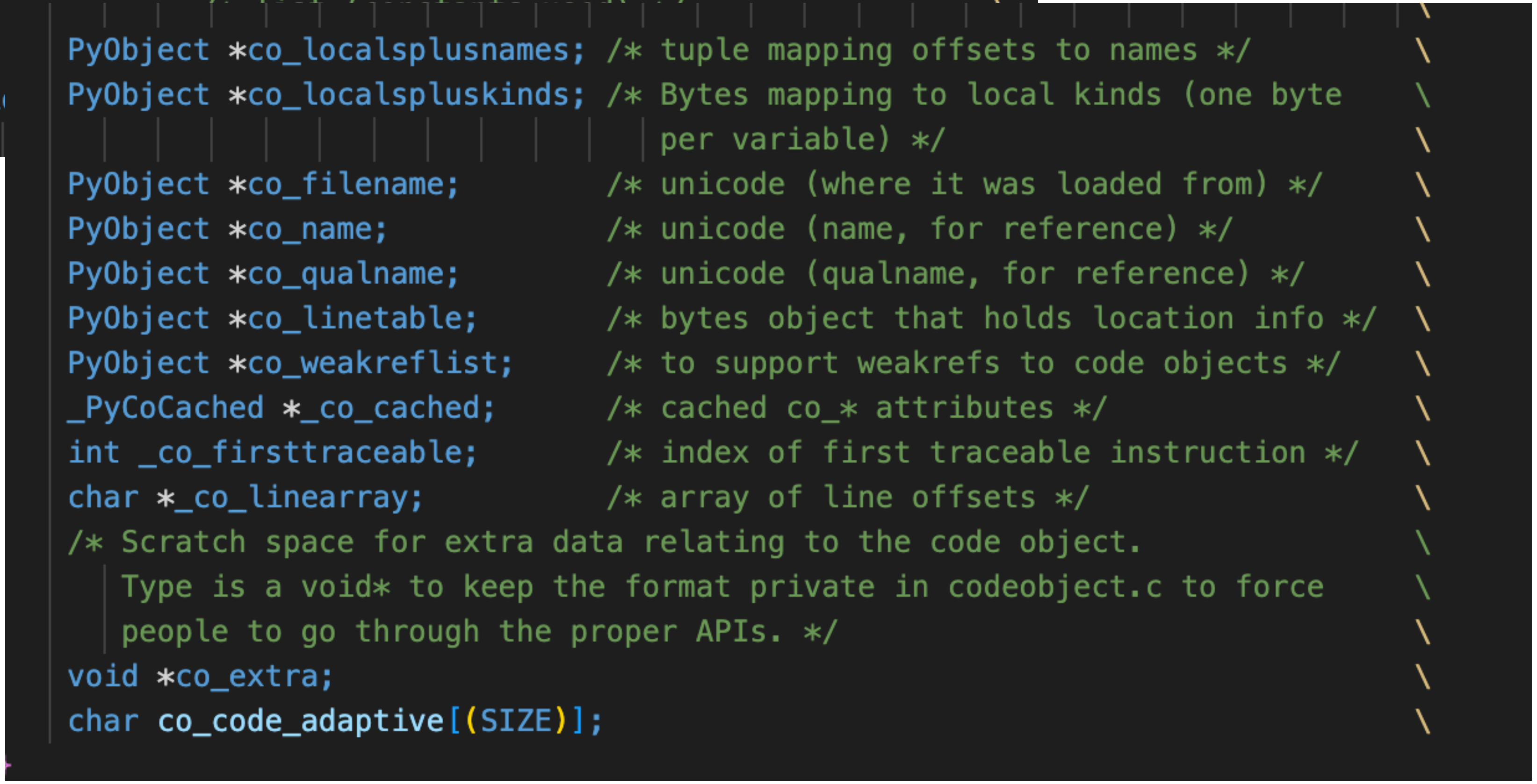
```
def cds_import_module(name):
    try:
        code = cds.code[name]
    except NotInArchive:
        # default path
        spec = finder.find_spec(name)
        code = spec.get_code(name)
    module = exec(code, {})
```

code: PyCodeObject

Snippet of PyCodeObject



```
    /* The hottest fields (in the eval loop) are grouped here at the top. */   \
    PyObject *co_consts;
    PyObject *co_names;
    PyObject *co_excepti
```

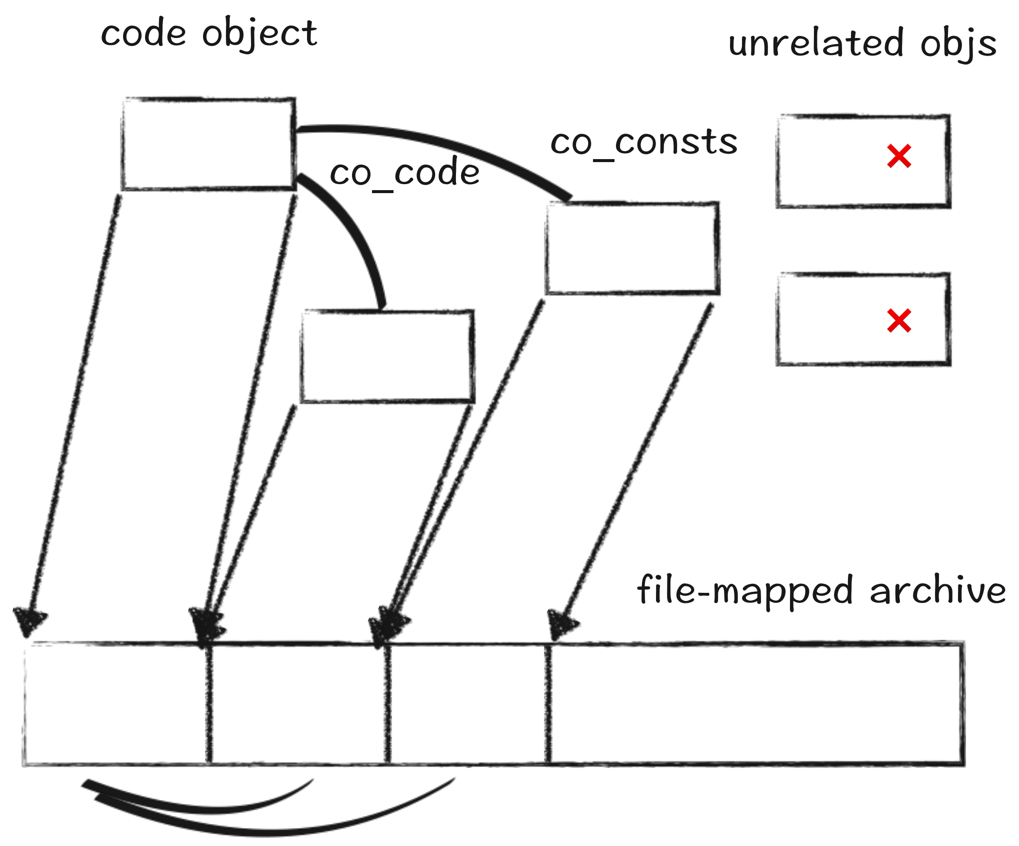


```
    PyObject *co_localsplusnames; /* tuple mapping offsets to names */        \
    PyObject *co_localspluskinds; /* Bytes mapping to local kinds (one byte   \
                                     per variable) */                         \
    PyObject *co_filename;        /* unicode (where it was loaded from) */    \
    PyObject *co_name;            /* unicode (name, for reference) */         \
    PyObject *co_qualname;        /* unicode (qualname, for reference) */     \
    PyObject *co_linetable;       /* bytes object that holds location info */ \
    PyObject *co_weakreflist;     /* to support weakrefs to code objects */   \
    _PyCoCached *_co_cached;      /* cached co_* attributes */                \
    int _co_firsttraceable;       /* index of first traceable instruction */  \
    char *_co_linearray;          /* array of line offsets */                 \
    /* Scratch space for extra data relating to the code object.              \
       Type is a void* to keep the format private in codeobject.c to force    \
       people to go through the proper APIs. */                               \
    void *co_extra;                                                           \
    char co_code_adaptive[(SIZE)];                                            \
```

AppCDS for Python Preview

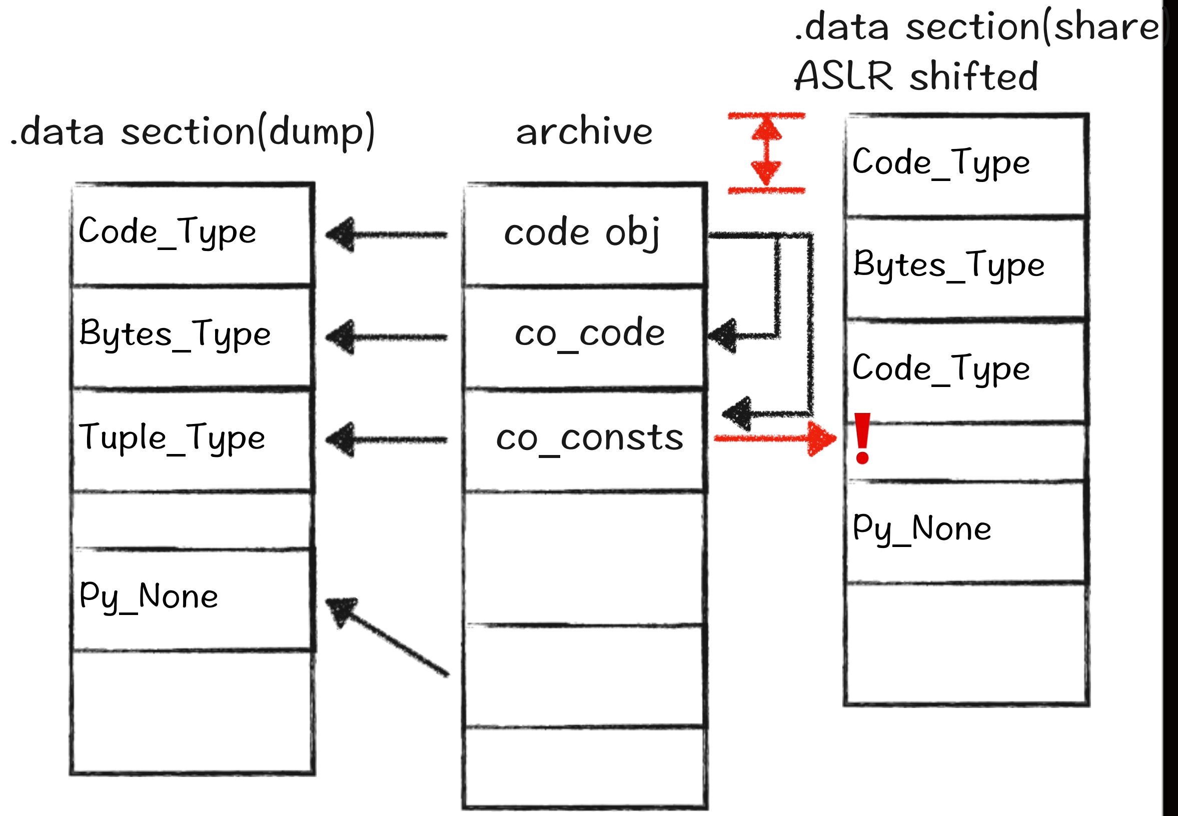
Python for Good
PyCon China 2022

Create Archive File



Make a compact memory image by recursive copy.

Fix-up pointers in runtime



MAP_FIXED ensures that the pointer inside the archive is correct, but ASLR causes the pointer to the data section invalid. Iterating over the object graph to fix the pointers.

Load code object from (shared) mmap-ed file.

Code object (and its fields)
may contain following types:

- bool/None/ellipsis
- float/complex
- long/bytes/str
- tuple
- frozenset

```
def cds_import_module(name):
    try:
        code = cds.code[name]
    except NotInArchive:
        # default path
        spec = finder.find_spec(name)
        code = spec.get_code(name)
    module = exec(code, {})
```

Questions?

Online environment to test:
https://lab.openanolis.cn/#/apply/chapters?courseId=117

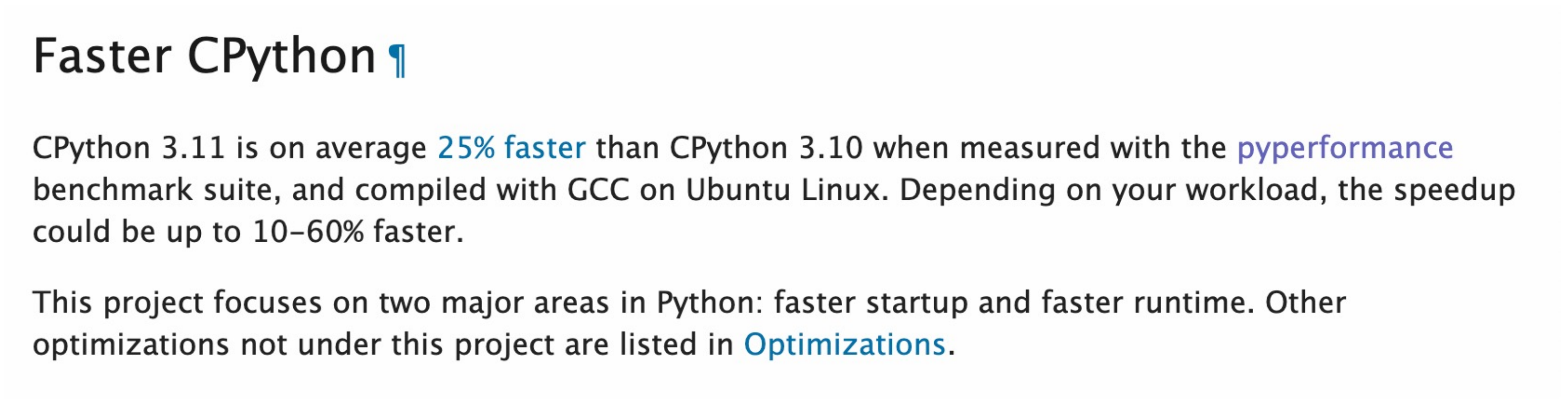
Github project:
https://github.com/alibaba/code-data-share-for-python

Pros:
- ~15% faster startup

Cons
- Generate archive for specific packages
- Do not work for frequently-changed codes

Community

- faster-cpython
- Cinder (Facebook)
    - Lazy Import
- nogil (FAIR)

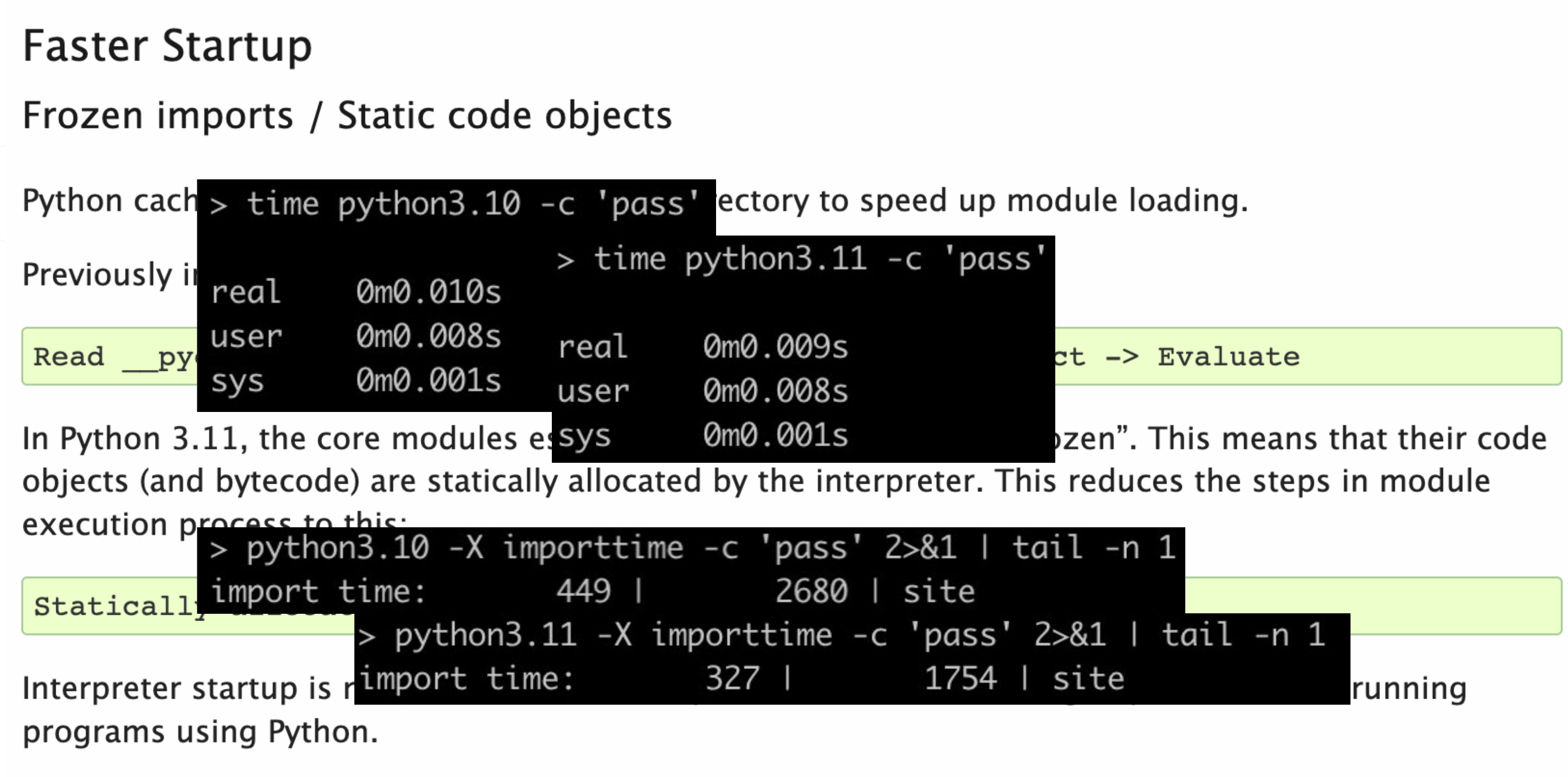


Faster CPython ¶

CPython 3.11 is on average 25% faster than CPython 3.10 when measured with the pyperformance benchmark suite, and compiled with GCC on Ubuntu Linux. Depending on your workload, the speedup could be up to 10–60% faster.

This project focuses on two major areas in Python: faster startup and faster runtime. Other optimizations not under this project are listed in Optimizations.

- MS & Guido
- Startup performance & runtime performance
    - Deep-freeze
    - Specializing Adaptive Interpreter
    - Inline cache



Faster Startup

Frozen imports / Static code objects

Python cach ectory to speed up module loading.

Previously i

```
Read __py                                                   ct -> Evaluate
```

In Python 3.11, the core modules es   zen”. This means that their code objects (and bytecode) are statically allocated by the interpreter. This reduces the steps in module execution process to this:

```
Staticall
```

Interpreter startup is r   running programs using Python.

```
> time python3.10 -c 'pass'

real    0m0.010s
user    0m0.008s
sys     0m0.001s
```

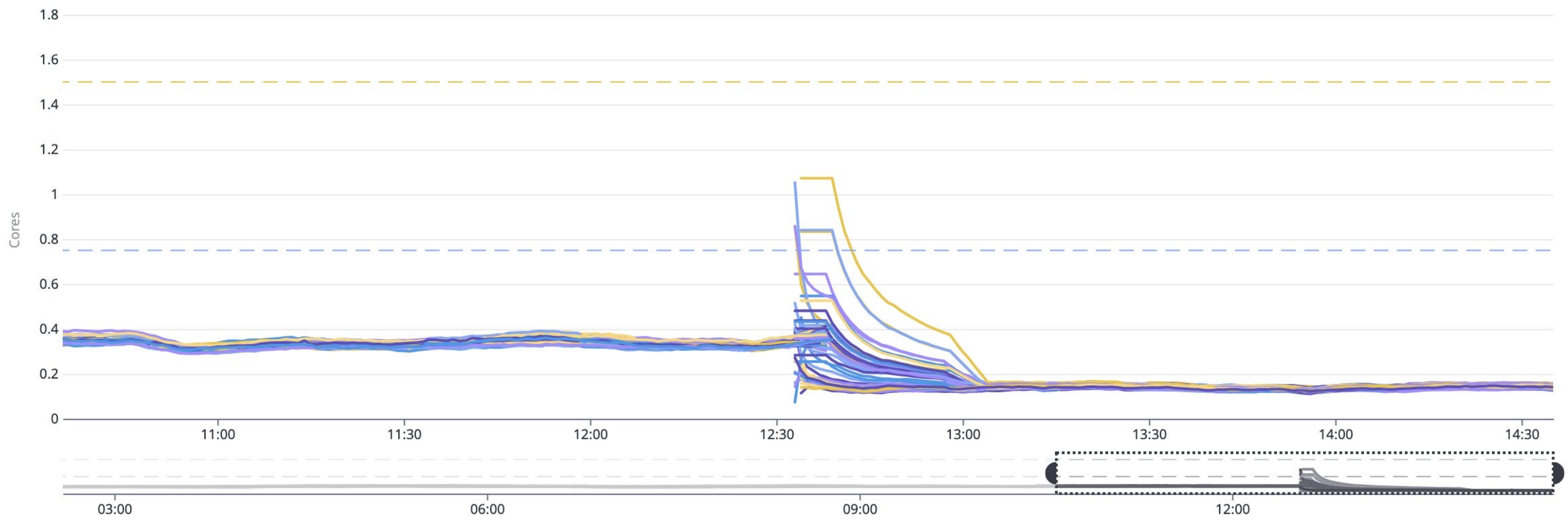
```
> time python3.11 -c 'pass'

real    0m0.009s
user    0m0.008s
sys     0m0.001s
```

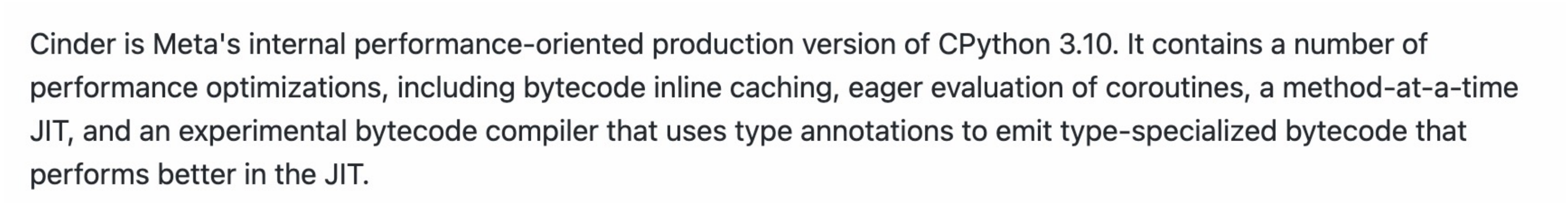
```
> python3.10 -X importtime -c 'pass' 2>&1 | tail -n 1
import time:       449 |       2680 | site
```

```
> python3.11 -X importtime -c 'pass' 2>&1 | tail -n 1
import time:       327 |       1754 | site
```

Web (Gunicorn) Service CPU Usage



Cinder

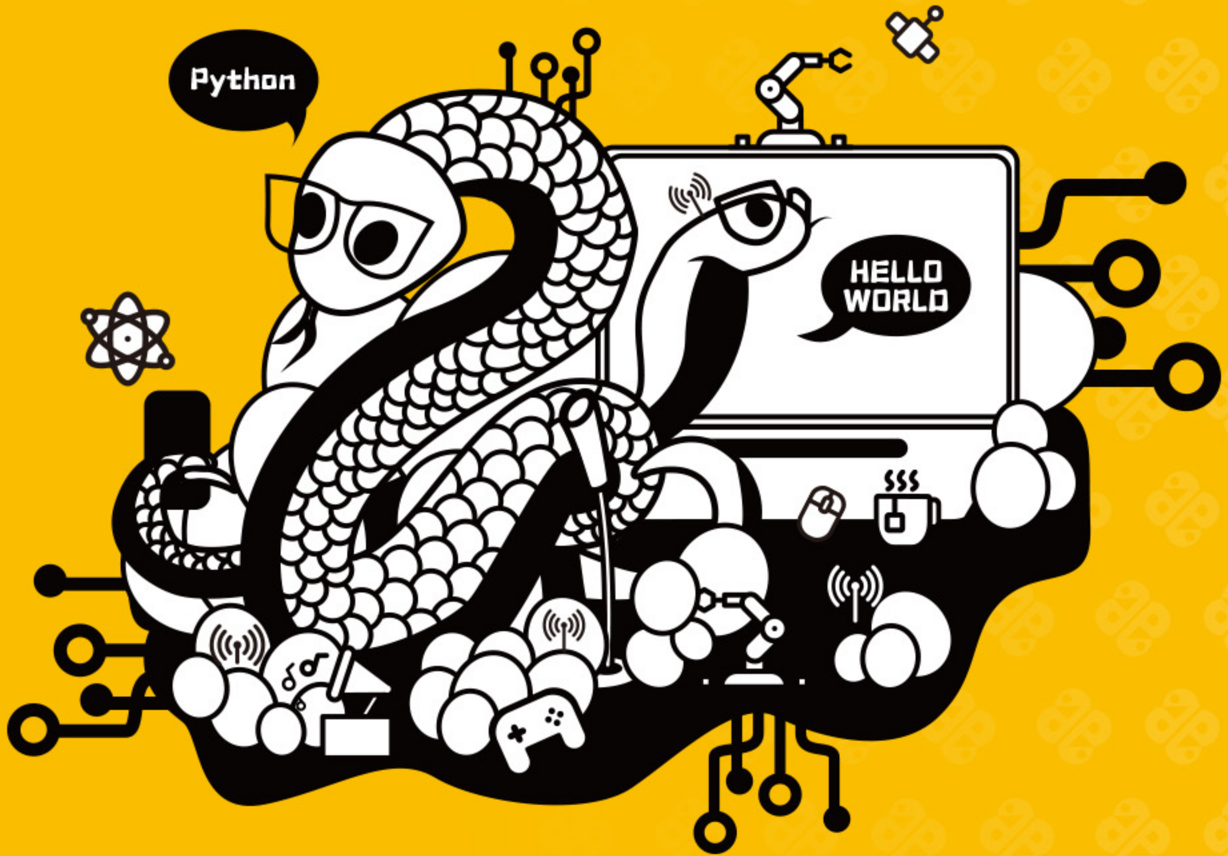


Cinder is Meta's internal performance-oriented production version of CPython 3.10. It contains a number of performance optimizations, including bytecode inline caching, eager evaluation of coroutines, a method-at-a-time JIT, and an experimental bytecode compiler that uses type annotations to emit type-specialized bytecode that performs better in the JIT.

- Cinder JIT
- Lazy import

nogil

- FAIR



Thanks!

感谢观看