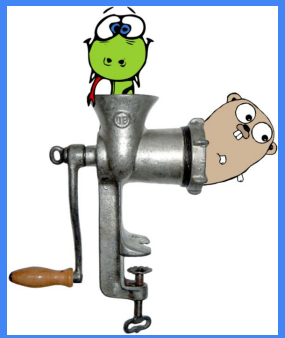
python™

CPython, Grumpy, PyPy

When, How, Why?

pypy



About Me

- Itay Weiss
- 27 Years - Alive
- 9 Years - Software Developer
- 7 Years - Army Intelligence
- 2 Years - Iguazio

CPython ≠ Python

CPython = A Language Engine

Python = Programming Language

Python and CPython

- CPython:
  - Is used by the majority of developers and users.
  - Compiles your python code into bytecode (transparently) and interprets that bytecode in an evaluation loop.
  - Performance is mainly limited by the GIL

GIL (Global Interpreter Lock)

- A binary semaphore that prevents multiple native threads from executing Python bytecodes at once.
- Must be held by the current thread before it can safely access Python objects.
- In order to emulate concurrency of execution, the interpreter regularly tries to switch threads (sys.setswitchinterval).

GIL ≠ Mutex

Simple Solution

MultiProcessing

PROS

- Code is usually straightforward
- Takes advantage of multiple CPUs & cores
- Avoids GIL limitations for CPython (basically each process has it's own GIL)
- Child processes are interruptible/killable

CONS

- Separate memory space
- Costly to run a huge number of processes
- Costly to create and destroy processes frequently
- IPC a little more complicated with more overhead
- Larger memory footprint

MultiThreading

PROS

- Shared memory
- Low memory footprint

CONS

- CPython - subject to the GIL
- Code is usually harder to understand and to get right - the potential for race conditions increases dramatically

Not So Simple Solution

Twisted/async.io (Asynchronous Programming/Event driven programing)

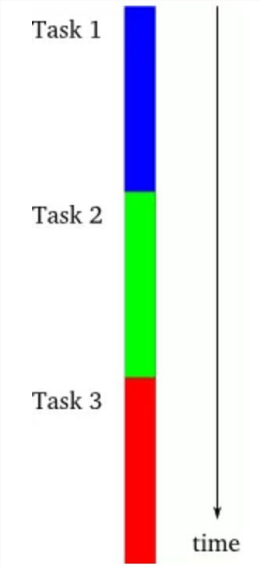


Figure 1: the synchronous model

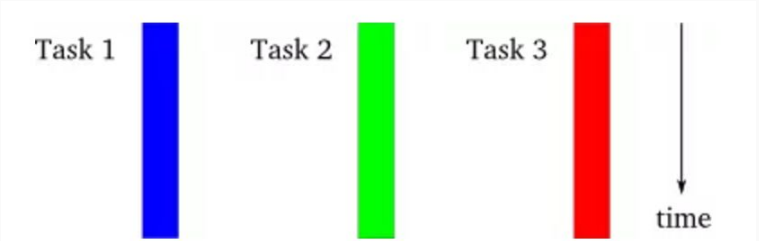


Figure 2: the threaded model

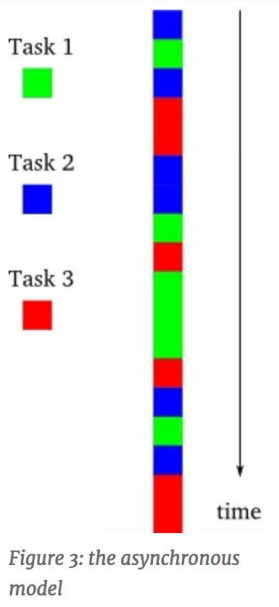


Figure 3: the asynchronous model

When Should I Use CPython

- GUI applications
- Network servers

Demo

GIL

PyPy

- Supports all of the core language (passing Python test suite)
- Supports most of the commonly used Python standard library modules (just check if it imports. If it imports, it should work):
  - __builtin__, __pypy__, _ast, _cffi_backend, _codecs, _collections, _continuation, _csv, _file, _hashlib, _io, _locale, _lsprof, _md5, _minimal_curses, _multibytecodec, _multiprocessing, _numpypy, _pickle_support, _pypyjson, _random, _rawffi, _sha, _socket, _sre, _ssl, _struct, _testing, _warnings, _weakref, array, binascii, bz2, cStringIO, cmath, cppyy, cpyext, crypt, errno, exceptions, fcntl, gc, imp, itertools, marshal, math, mmap, operator, parser, posix, pwd, pyexpat, pypyjit, select, signal, symbol, sys, termios, thread, time, token, unicodedata, zipimport, zlib
- Built using the RPython language that was co-developed with it:
  - A framework for producing implementations of dynamic languages (like python)
  - Able to automatically generate a Just-in-Time compiler for any dynamic language
- Implements a JIT compiler in Python.
- Support for Stackless and Lightweight concurrent programming (Greenlets).

How Are PyPy Getting Rid Of The GIL

```
def f(list1, list2):
    x = list1.pop()
    list2.append(x)
```

```
def f(list1, list2):
    global_lock.acquire()
    x = list1.pop()
    list2.append(x)
    global_lock.release()
```

Software Transactional Memory

```
def f(list1, list2):
    while True:
        t = transaction()
        x = list1.pop(t)
        list2.append(t, x)
        if t.commit():
            break
```

JIT(Just In Time)

- Done during execution (runtime).
- Continuously analyses the code being executed and identifies parts of the code where the speedup gained from compilation would outweigh the overhead of compiling that code
- In theory JIT compilation can yield faster execution than static compilation.

Demo

Performance + How to use PyPy

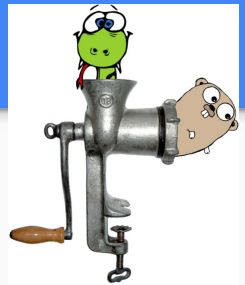
When Should I Use PyPy

- Long running processes
- When running mostly Python code.

Grumpy

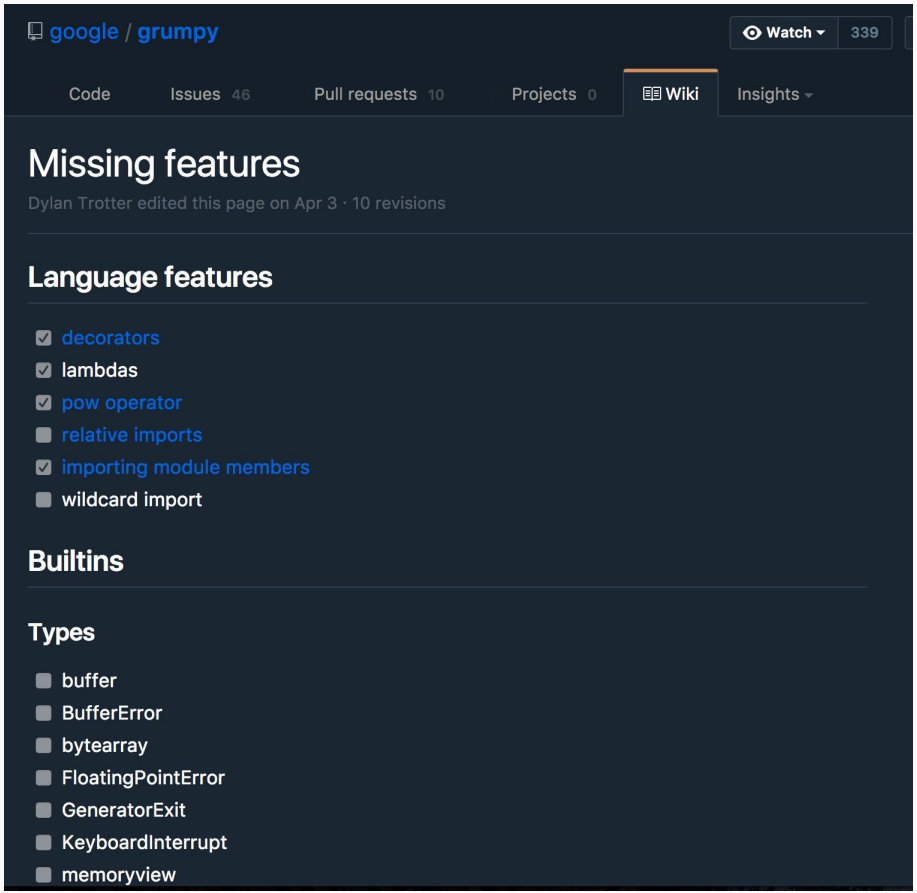
- Google has YouTube which is:
  - Written In CPython (2.7)
  - Serves millions of requests per second
- Google decides to Implement an alternative runtime optimized for real-time serving.
- Google decides to use Go.
- Grumpy was born!

Grumpy



- Python to Go source code transcompiler and runtime.
- Grumpy has no VM.
- Has no GIL.
- Uses Go’s garbage collection instead of counting references.
- Uses Go’s goroutines
- No support for C extension modules.

Grumpy Features



google / grumpy

Watch 339

Code | Issues 46 | Pull requests 10 | Projects 0 | Wiki | Insights

Missing features

Dylan Trotter edited this page on Apr 3 · 10 revisions

Language features

- [x] decorators
- [x] lambdas
- [x] pow operator
- [ ] relative imports
- [x] importing module members
- [ ] wildcard import

Builtins

Types

- [ ] buffer
- [ ] BufferError
- [ ] bytearray
- [ ] FloatingPointError
- [ ] GeneratorExit
- [ ] KeyboardInterrupt
- [ ] memoryview
- [ ] UnicodeTranslateError

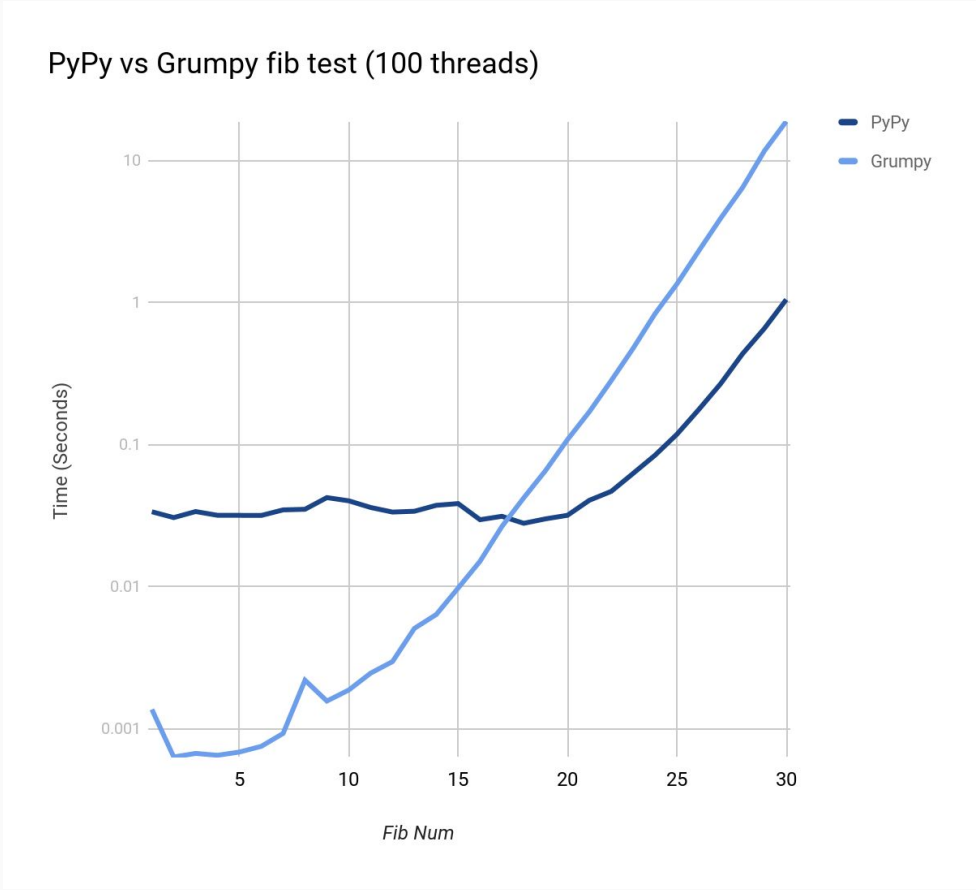
Functions

- [ ] apply()
- [ ] coerce()
- [ ] filter()
- [ ] format()
- [x] globals()
- [ ] input()
- [ ] intern()
- [ ] locals()
- [x] map()
- [ ] pow()
- [ ] reduce()
- [ ] reload()
- [ ] reversed()
- [x] round()
- [x] sorted()
- [ ] vars()

Demo

Performance + How to use Grumpy

It All Comes Down To This



When Should I Use Grumpy

- When you’re using a lot of threads, each executing a very small task.
- When you need to use any of the go libraries.

Basic Comparison

CPython

- The reference implementation of Python
- Written in C
- compiles Python code to intermediate bytecode which is then interpreted by a virtual machine

PyPy

- Supports all of the core language (passing Python test suite)
- supports most of the commonly used Python standard library modules
- Implements a JIT compiler in Python

Grumpy

- Python to Go source code transcompiler.
- Grumpy has no VM.
- Uses Go's garbage collection instead of counting references.

Performance Comparison

GIL

- A binary semaphore that prevents multiple native threads from executing Python bytecodes at once.
- Must be held by the current thread before it can safely access Python objects

JIT

- Done during execution (runtime).
- In theory JIT compilation can yield faster execution than static compilation.

Go

- Has no GIL.
- Statically compiled.
- Uses Go's garbage collection instead of counting references.
- Uses Go's goroutines

Thanks

- David Beazley (http://www.dabeaz.com/) - https://www.youtube.com/watch?v=ph374fJqFPE
- Dave Peticolas - http://krondo.com/an-introduction-to-asynchronous-programming-and-twisted/
- Emmanuel Klinger - http://emmanuel-klinger.net/python-100-times-faster-than-grumpy.html
- PyPy blog: https://morepypy.blogspot.co.il/2011/06/global-interpreter-lock-or-how-to-kill.html
- Stackoverflow
  - https://stackoverflow.com/questions/18946662/why-shouldnt-i-use-pypy-over-cpython-if-pypy-is-6-3-times-faster
  - https://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python
  - https://stackoverflow.com/questions/62814/difference-between-binary-semaphore-and-mutex

More Questions

me@itayweiss.tech