Introduction to Grumpy

M157q

Golang Taipei Gathering #25

20170718

Who is M157q?

- Shun-Yi Jheng
- https://github.com/M157q
- https://blog.m157q.tw
- Just a Pythonista (since 2012) who began to learn Golang about 5 months ago (GTG #21).
- DevOps & Architect @ Tagtoo

Outline

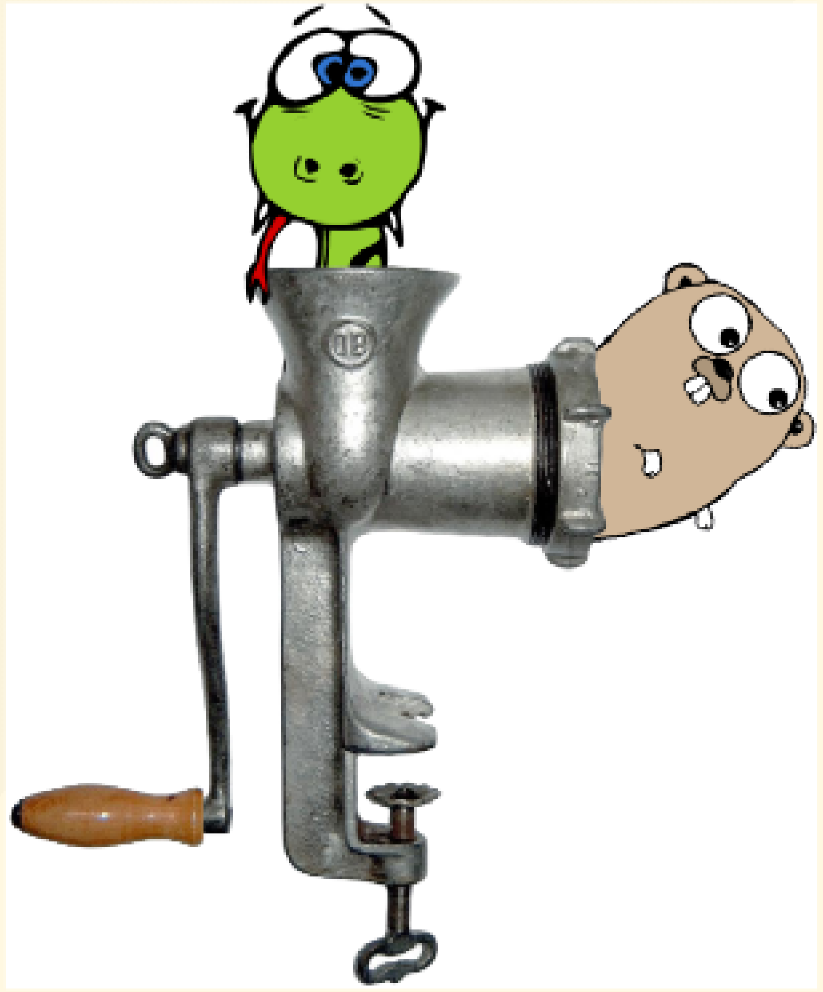
What is Grumpy?

Grumpy == Go running Python

What is Grumpy?

- A Python to Go source code transcompiler developed by Google.
- A near drop-in replacement for CPython 2.7.
- Compiles Python code to native Go code, rather than to bytecode. (No VM).
- No GIL (Global Interpreter Lock).
- Leverages Go's garbage collection for object lifetime management instead of counting references.
- The compiled Go source code is a series of calls to the Grumpy runtime.

Why Grumpy?



Why Grumpy? (from Google)

> “ It's very difficult to make concurrent workloads perform well on CPython. ”

> “ For other Python runtimes, each had trade-offs and none solved the concurrency problem without introducing other issues. ”

> “ Implement an alternative runtime optimized for real-time serving? ”

> “ We wanted first class language interoperability and Go’s powerful runtime type reflection system made this straightforward. Python in Go felt very natural. ”

Why Grumpy? (from me)

- Python -> **Easy to write**, but slow.
- Golang -> Not that easy to write, but **fast**.
- Grumpy -> Try to achieve **easy to write and fast**.
- Make Python run faster by using Golang.

Who might need Grumpy?

Who might need Grumpy?

- Pythonistas who eager performance.
- People who interested in both Golang and Python.
- People who don't want rewrite Python code to Golang code.

Limitations of Grumpy

Limitations of Grumpy

- Never be supported:
  - `exec`, `eval` and `compile`
    - These dynamic features of CPython are not supported by Grumpy because Grumpy modules consist of statically-compiled Go code.
  - C extension modules
    - Grumpy has a different API and object layout than CPython and so supporting C extensions would be difficult.

Limitations of Grumpy

- Will support but doesn't yet
  - Language features
  - Builtin functions and types
  - Standard library
  - C locale support
    - Go doesn't support locales in the same way that C does.
    - Locale-dependent may not currently work the same as in CPython.
  - Python 3

Source Code Overview

```
Python Source Code => Grumpc => Golang Source Code => Binary
```

Source Code Overview: Components

- Grumpy Compiler (`grumpc`)
  - Python, `compiler`
- Grumpy Runtime
  - Golang, `runtime`
- Grumpy Standard Library
  - Python, `third_party` or Golang, `lib`

Source Code Overview: Directories

- `compiler`: Python package implementating Python -> Go transcompilation logic.
- `lib`: Grumpy-specific Python standard library implementation.
- `runtime`: Go source code for the Grumpy runtime library.
- `third_party/ouroboros`: Pure Python standard libraries copied from the Ouroboros project.
- `third_party/pypy`: Pure Python standard libraries copied from PyPy.
- `third_party/stdlib`: Pure Python standard libraries copied from CPython.
- `tools`: Transcompilation and utility binaries.

Grumpy Compiler (`grumpc`, `compiler`)

- Parsing Python code and generating Golang code.
- Written in Python and uses the pythonparser module to accomplish parsing.
- The grumpc script itself lives at `tools/grumpc`.
- It is supported by a number of Python modules in the `compiler` subdir.

Grumpy Runtime

- The Go code generated by `grumpc` performs operations on data structures that represent Python objects in running Grumpy programs.
- These data structures and operations are defined in the grumpy Go library (source is in the `runtime` subdir of the source distribution).
- This runtime is analogous to the Python C API and many of the structures and operations defined by grumpy have counterparts in CPython.

Grumpy Standard Library

- Much of the Python standard library is written in Python and thus "just works" in Grumpy.
- These parts of the standard library are copied from CPython 2.7 (possibly with light modifications).
- For licensing reasons, these files are kept in the `third_party` subdir.
- The parts of the standard library that cannot be written in pure Python, e.g. file and directory operations, are kept in the `lib` subdir.
- In CPython these kinds of modules are written as C extensions. In Grumpy they are written in Python but they use native Go extensions to access facilities not otherwise available in Python.

How to use Grumpy?

```
git clone https://github.com/google/grumpy.git
```

How to use Grumpy?

- `grumprun`

```
$ echo "print 'hello, world'" | make run
```

- `grumpc`

```
$ echo 'print "hello, world"' > hello.py
$ make
$ export GOPATH=$PWD/build
$ export PYTHONPATH=$PWD/build/lib/python2.7/site-packages
$ build/bin/grumpc hello.py > hello.go
$ go build -o hello hello.go
$ ./hello
```

How to use Grumpy?

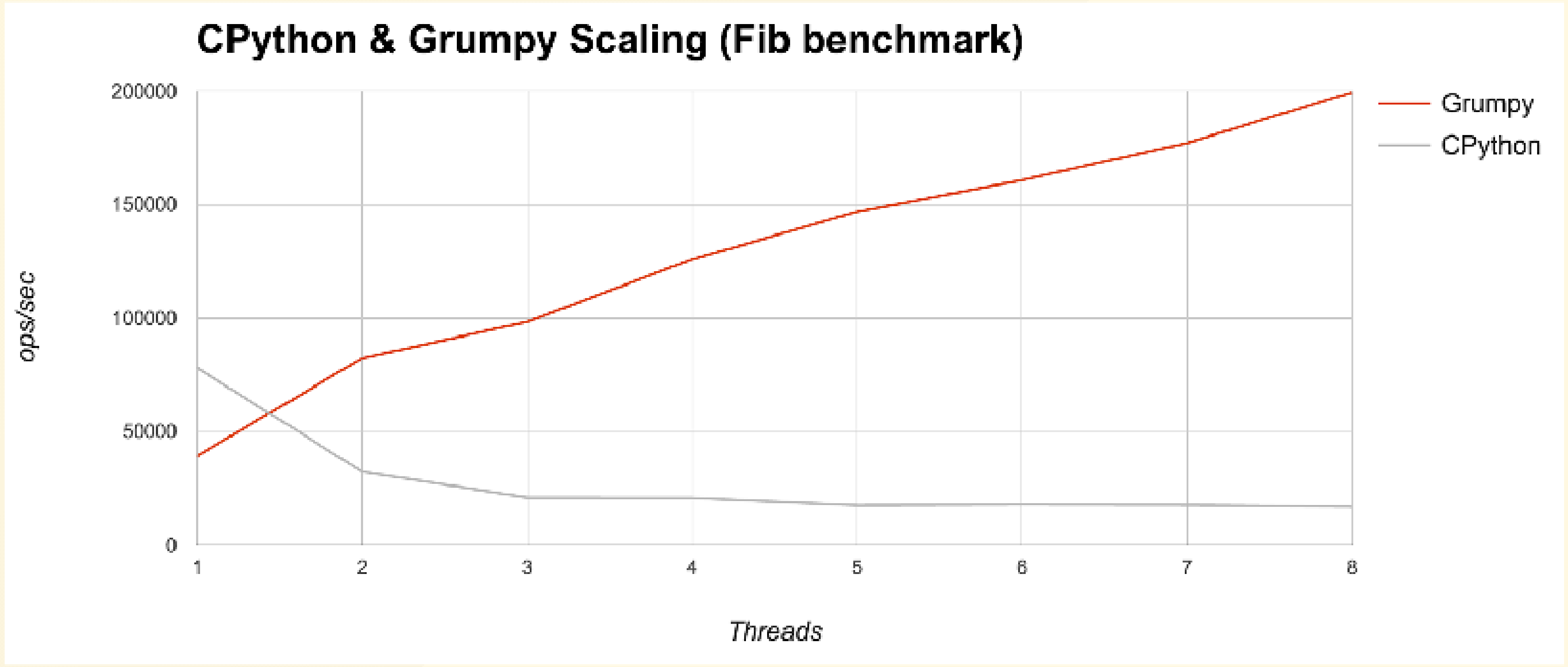
- Advance Tips
  - Native Imports
  - Using C libraries in Grumpy
    - Wrapping the C library using `cgo`
    - Using the Go package from Python via Grumpy native imports.

Performance:

Before and After using Grumpy

Performance

“



”

Conclusion

- An experimental, but progressive project.
- A good example for transpiler.
- High possibility that someday Google will use it for prodcuts written in Python for improving performance, especially ones related to concurrency.
- Currently, not suitable for using in production.
- If you happens to be a person interested in both Python and Golang, this is a good project to study and contribute to.

References

- google/grumpy
- Grumpy: Go running Python!
- Grumpy: Go running Python | Hacker News

Q&A