ThriftPy Documentation

Release 0.1.15

Lx Yu

December 12, 2014

Contents

ThriftPy is a pure python implementation of Apache Thrift in a pythonic way.

The official thrift python lib is not pythonic at all, it needs a complicated process of installation, and the generated sdk is very ugly. Everytime the thrift file changed you have to re-generate the sdk which causes more pain in development.

ThriftPy helps that, it's compatible with Apache Thrift so you no longer need to install 'thrift' package, it can import thrift file on the fly so you no longer need to re-generate the sdk again and again and again.

Github: https://github.com/eleme/thriftpy

CHAPTER 1

Code Demo

ThriftPy make it super easy to write server/client code with thrift. Let's checkout this simple pingpong service demo.

We need a 'pingpong.thrift' file:

```
service PingPong {
    string ping(),
}
```

Then we can make a server:

```
import thriftpy
pingpong_thrift = thriftpy.load("pingpong.thrift", module_name="pingpong_thrift")

from thriftpy.rpc import make_server

class Dispatcher(object):
    def ping(self):
        return "pong"

server = make_server(pingpong_thrift.PingPong, Dispatcher(), '127.0.0.1', 6000)
server.serve()
```

And a client:

```
import thriftpy
pingpong_thrift = thriftpy.load("pingpong.thrift", module_name="pingpong_thrift")

from thriftpy.rpc import make_client

client = make_client(pingpong_thrift.PingPong, '127.0.0.1', 6000)
client.ping()
```

See, it's that easy!

You can refer to 'examples' and 'tests' directory in source code for more usage examples.

CHAPTER 2

Features

Currently ThriftPy have these features (also advantages over the upstream python lib):

- Supports python2.6+, python3.3+, pypy and pypy3.
- Compatible with Apache Thirft. You can use ThriftPy together with the official implementation servers and clients, such as a upstream server with a thriftpy client or the opposite.

  Currently implemented protocols and transports:

  - binary protocol (python and cython)
  - buffered transport (python & cython)
  - tornado server and client (with tornado 4.0)
  - framed transport
  - json protocol
- Can directly load thrift file as module, the sdk code will be generated on the fly.

  For example, `pingpong_thrift = thriftpy.load("pingpong.thrift", module_name="pingpong_thrift")` will load 'pingpong.thrift' as 'pingpong_thrift' module.

  Or, when import hook enabled by `thriftpy.install_import_hook()`, you can directly use `import pingpong_thrift` to import the 'pingpong.thrift' file as module, you may also use `from pingpong_thrift import PingService` to import specific object from the thrift module.
- Pure python implementation. No longer need to compile & install the 'thrift' package. All you need is thriftpy and thrift file.
- Easy RPC server/client setup.

CHAPTER 3

Installation

Install with pip

```
$ pip install thriftpy
```

You may also install cython first to build cython extension locally.

```
$ pip install cython thriftpy
```

CHAPTER 4

Usage Notice

4.1 Use Cython Binary Protocol

The TCyBinaryProtocol can be used to accelerate serialize and deserialize.

Note: The TCyBinaryProtocol and TCyBufferedTransport must be used together.

```
from thriftpy.protocol import TCyBinaryProtocolFactory
from thriftpy.transport import TCyBufferedTransportFactory
from thriftpy.rpc import client_context

server = make_server(
    pingpong_thrift.PingPong, Dispatcher(), '127.0.0.1', 6000,
    proto_factory=TCyBinaryProtocolFactory(),
    trans_factory=TCyBufferedTransport())
print("serving...")
server.serve()
```

The same goes for client.

```
from thriftpy.protocol import TCyBinaryProtocolFactory
from thriftpy.transport import TCyBufferedTransportFactory
from thriftpy.rpc import make_client

client = make_client(
    pingpong_thrift.PingPong, '127.0.0.1', 6000,
    proto_factory=TCyBinaryProtocolFactory(),
    trans_factory=TCyBufferedTransportFactory())
client.ping()
```

Or client context:

```
from thriftpy.protocol import TCyBinaryProtocolFactory
from thriftpy.transport import TCyBufferedTransportFactory
from thriftpy.rpc import client_context

with client_context(
        pingpong_thrift.PingPong, '127.0.0.1', 6000,
        proto_factory=TCyBinaryProtocolFactory(),
        trans_factory=TCyBufferedTransportFactory()) as c:
    c.ping()
```

4.2 Better Module

To load thrift file as better module, provide a *module_name* in *load*.

The direct loaded TObjects can't be pickled.

```
>>> ab = thriftpy.load("addressbook.thrift")
>>> pickle.dumps(ab.AddressBook())
PicklingError: Can't pickle <class 'addressbook.AddressBook'>
```

TObjects can be pickled when load with *module_name* provided.

```
>>> ab = thriftpy.load("addressbook.thrift", "addressbook_thrift")
>>> pickle.dumps(ab.AddressBook())
b'\x80\x03caddressbook_thrift\nAddressBook\nq\x00)\x81q\x01}q\x02X\x06\x00\x00\x00peopleq\x03Nsb.'
```

You can also use *from ... import ...* style after a standard module load.

```
>>> ab = thriftpy.load("addressbook.thrift", "addressbook_thrift")
>>> from addressbook_thrift import *
```

CHAPTER 5

Benchmarks

Some benchmark results:

```
# apache thrift py binary
binary protocol struct benchmark for 100000 times:
encode  -> 3.74061203003
decode  -> 5.02829790115

# apache thrift c binary
accelerated protocol struct benchmark for 100000 times:
encode  -> 0.398949146271
decode  -> 0.536000013351

# thriftpy & pypy2.3
binary protocol struct benchmark for 100000 times:
encode  -> 0.413738965988
decode  -> 0.605606079102

# thriftpy & py3.4
binary protocol struct benchmark for 100000 times:
encode  -> 3.291545867919922
decode  -> 4.337666034698486

# thriftpy & py3.4 + cython
cybin protocol struct benchmark for 100000 times:
encode  -> 0.5828649997711182
decode  -> 0.8259570598602295
```

Checkout the *benchmark/benchmark.rst* for detailed benchmark scripts and scores.

CHAPTER 6

Changelogs

https://github.com/eleme/thriftpy/blob/master/CHANGES

CHAPTER 7

Contribute

1. Fork the repo and make changes.
2. Write a test which shows a bug was fixed or the feature works as expected.
3. Make sure travis-ci test succeed.
4. Send pull request.

CHAPTER 8

Contributors

https://github.com/eleme/thriftpy/graphs/contributors