

Tornado Web and MongoDB

An Introduction to AsyncMongo

MongoPhilly

2011-04-26

John C. Zablocki

Development Lead, MagazineRadar

Engineering Lead, I'mOK

Agenda

- Evented I/O Web Servers
- Introducing Tornado Web
- Asynchronous Tornado
- Tornado and PyMongo
- Introducing AsyncMongo
- Tornado and AsyncMongo
- Fetching Data with AsyncMongo
- Inserts and Updates with AsyncMongo
- Questions



EVENTED I/O WEB SERVERS

The C10k Problem

- How does a web server handle 10k concurrent requests?
 - Traditional approach is multi-process (Apache) or multi-threaded (IIS, Apache)
 - Threads and processes are expensive to create, in terms of memory and CPU
 - Scaling web servers such as IIS and Apache to thousands of connections therefore becomes expensive

Evented I/O Web Servers

- A Single-threaded server with an event loop that handles all requests
- By using non-blocking libraries for database, file or network access, that single thread is able to handle thousands of concurrent connections
- Typically architected around epoll or select() (for non-Linux systems)
- Callbacks used to notify thread that non-blocking I/O has completed
- Any blocking calls on that thread, will block other requests – most platforms are lacking non-blocking APIs
- Explained with bunnies at - <http://www.slideshare.net/simon/evented-io-based-web-servers-explained-using-bunnies>

Motivation for Evented I/O

The cost of I/O

L1 cache: 3 cycles

L2 cache: 14 cycles

RAM: 250 cycles

Disk: 41,000,000 cycles

Network: 240,000,000 cycles

Borrowed from a presentation on Node .js by Francisco Treacy (Bing it)

Some Evented I/O Web Servers

- **nginx**
 - Load balancing and reverse proxying
 - Static and index file serving
 - SSL and TLS support
 - Online reconfiguration
- **node.js**
 - JavaScript based
 - Built on epoll, kqueue or select
 - Server and app dev framework based on callbacks
- **Manos de Mono**
 - C# (Mono) based
 - Built on libvent (epoll)
 - Server and app dev framework based on callbacks
- **Twisted Web**
 - Python based
 - Server and app dev framework
 - Can serve CGI scripts and WSGI applications



INTRODUCING TORNADO WEB

Tornado Web Overview

- Open source version of the framework that powers FriendFeed
- Consists of non-blocking web server and web application framework similar to Google's AppEngine
- Includes modules for templating, third-party authorization, localization and a non-blocking http client, among others
- Install via `easy_install tornado`
- Runs on Linux, Mac OS X, Windows (support is experimental)

Tornado Web Hello, World!

```
import tornado.web
import tornado.ioloop

class MainHandler(tornado.web.RequestHandler):

    def get(self):
        self.write("Hello, Tornado world!")

application = tornado.web.Application([
    (r"/", MainHandler)
])

if __name__ == "__main__":

    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

Asynchronous Tornado Web

```
import tornado.web
import tornado.ioloop

class AsyncHandler(tornado.web.RequestHandler):

    @tornado.web.asynchronous
    def get(self):
        http = tornado.httpclient.AsyncHTTPClient()
        http.fetch("http://z.com", callback=self.on_resp)

    def on_resp(self):
        self.write("Hello, Tornado world!")
        self.finish()
```

PyMongo and Tornado Web

- PyMongo does not provide support for working with asynchronous socket libraries, such as Tornado or Twisted
- PyMongo does provide connection pooling, so similar performance benefits may be achieved via multi-threaded code sharing a connection
- A driver based on twisted is actively being developed -
<https://github.com/fiorix/mongo-async-python-driver>

PyMongo and Tornado Example

```
import tornado.web  
import tornado.ioloop
```

```
import pymongo
```

```
class RestaurantListHandler(tornado.web.RequestHandler):
```

```
    def get(self):
```

```
        db = pymongo.Connection("localhost", 27017)  
        venues = db.venues.find()  
        self.render("List.html", venues=venues)
```

Introducing AsyncMongo

- AsyncMongo is an asynchronous library for accessing MongoDB
- Built on top of Tornado's ioloop
- Released to the community by bit.ly
- Current version is 0.1.3 and is stable, but not feature complete (when compared to pymongo)
- Relies on PyMongo's BSON library

AsyncMongo and Tornado Example

```
import tornado.web
import tornado.ioloop

import asyncmongo
class RestaurantListHandler(tornado.web.RequestHandler):

    @tornado.web.asynchronous
    def get(self):

        db = asyncmongo.Client(host="192...", port=27017, ...)
        db.venues.find(limit=10, callback=self.cb)

    def cb(self, response, error):
        self.render("List.html", venues=response)
```

Fetching Data with AsyncMongo

@tornado.web.asynchronous

```
def get(self):
```

```
    id = self.get_argument("id", None)
```

```
    if id != None:
```

```
        spec = { "_id" : pymongo.objectid.ObjectId(id) }
```

```
    db = DBFactory.create()
```

```
    db.venues.find_one(spec, \
```

```
        callback=lambda r, e: self.render("Edit.html", venue=r)
```

Inserting Data with AsyncMongo

```
@tornado.web.asynchronous
```

```
def post(self):
```

```
    doc = {}
```

```
    doc["name"] = self.get_argument("name")
```

```
    doc["city"] = self.get_argument("city")
```

```
    doc["state"] = self.get_argument("state")
```

```
    db = DBFactory.create()
```

```
    db.venues.insert(doc, callback=self._post_callback)
```

```
def _post_callback(self, response, error):
```

```
    self.write("Created")
```

```
    self.redirect(r"/")
```

Updating Data with AsyncMongo

@tornado.web.asynchronous

```
def post(self):
```

```
    id = self.get_argument("_id", None)
```

```
    doc = {}
```

```
    doc["_id"] = pymongo.objectid.ObjectId(id)
```

```
    doc["name"] = self.get_argument("name")
```

```
    doc["city"] = self.get_argument("city")
```

```
    doc["state"] = self.get_argument("state")
```

```
    db = DBFactory.create()
```

```
    db.venues.update({ "_id" : pymongo.objectid.ObjectId(id) }, \
                    doc, callback=lambda r, e: self.redirect(r"/"))
```

Deleting Data with AsyncMongo

```
@tornado.web.asynchronous
```

```
def post(self):
```

```
    id = self.get_argument("_id", None)
```

```
    db = DBFactory.create()
```

```
    db.venues.remove({ "_id" : pymongo.objectid.ObjectId(id) }, \
                      callback=lambda r, e: self.redirect(r"/"))
```

Links

- <http://www.kegel.com/c10k.html>
- <http://www.tornadoweb.org/>
- <https://github.com/facebook/tornado>
- <https://github.com/bitly/asyncmongo>
- <http://www.codevoyeur.com>
- <http://dllhell.net>
- <http://twitter.com/codevoyeur>
- <http://about.me/johnzablocki>



Questions?