Twisted Places Proxy Herd

Stanley Ku
University of California, Los Angeles
May 31, 2014

Abstract—Twisted provides an event-driven networking framework on top of Python and serves as a potential candidate for implementing an application server herd. This paper examines pros and cons of the Twisted framework with regards to the ease of implementation, maintainability, and reliability.

Index Terms—Twisted, Node.js, application server herd

1 Introduction

The LAMP architecture, a stack composed of Linux, Apache, MySQL, and PHP, provides a general-purpose web server. Typical deployments make use of multiple redundant servers behind a load-balancing virtual router. The redundancy improves reliability and performance. However, adoption of new technology and the shifting Internet architecture lead to new considerations: more frequent updates, access via various protocols, and mobile clients, that impact the performance of the LAMP architecture. This gives motivation to explore the viability of an application server herd, implemented using event-driven programming and Twisted.

2 Twisted

2.1 Overview

Twisted is an event-driven networking engine that supports numerous protocols. At the core is the reactor design pattern and deferred objects. The reactor is the main event loop that handles service requests/events from one or more clients concurrently and dispatches them to the relevant event handlers. Twisted is based on a single thread implementation because multi-threading fails to resolve efficiency and programming simplicity. Threading may lead to poor performance due to context switching, and synchronization and increased complexity due to concurrency. Because it is single threaded, the reactor is required not to block. Blocking calls are pushed off to deferred objects, which is a placeholder for the future. Callbacks are added to the deferred objects to perform some computation when the deferred objects have returned with a value.

2.2 Syntax

Since Twisted is written in Python, the syntax is very concise. Twisted does a very good job with abstracting the technicalities and providing simple API calls. A simple server can be implemented in under fifty lines of code (LOC). The application server herd prototype discussed later in the paper is implemented with less than 200 LOC.

3 Implementation

3.1 Overview

Most of the code is in the ProxyHerdProtocol class, which is subclassed from twisted.internet.protocol.Protocol. The ProxyHerdServer creates a factory that instantiates the protocol on a per-connection basis. The reactor listens on a specified port and invokes the factory.

3.2 Protocol

The protocol uses the LineReceiver protocol that dispatches the LineReceived method for each line. In the method, it checks for the command, first token in the line and dispatches the corresponding event handler.

3.3 IAMAT Handler

The IAMAT handler processes a client message. It first checks that it is a valid command. If its not, it returns the invalid command to the client. It parses the messages into tokens and formats the server response to client. The original message and client time stamp is stored in a local cache. The formatted response is sent back to the client and a separate method is called to forward the client location to the neighbors of current server and propagates through the entire herd. During the information forwarding, the current server acts as a client, establishes connections to its neighbors, and sends the message. The neighbors are kept in a global dictionary with the server name being the keys and list of neighbors as corresponding value.

3.4 AT Handler

The AT message is sent by servers as a result of IAMAT client location update. The handler checks if an entry for corresponding client exists in cache and if the time stamp matches. If so, then the message is a duplicate and the method returns. This stops the server from forwarding the duplicate update and prevents an infinite loop of circulation. Otherwise, the server will update the cache with the new client information and continue forwarding the message to its neighbors.

3.5 WHATSAT Handler

The WHATSAT handler processes the client request to perform a Google Places API call. The handler finds the client information in the cache and uses it, along with the client request, to generate the API call. The getPage method is used to invoke the API call. This is a blocking call that generates a deferred object. A callback is added that will process the returned response. Because the API doesnt have a parameter that limits the search results, the Python json module is used to load the JSON response and filter out the extraneous response. The filtered results are then sent back to the client.

3.6 Logging

Logging is done with the Python logging module. During initialization of the factory, logging is configured with a filename that lists the server name, date, and time. All notable events and errors are logged: number of client connections, events, etc.

3.7 Execution

A normal Python call is used to instantiate a server. The call takes in a single argument: name of the server.

```
python proxyherd.py <server name>
```

3.8 Testing

I wrote a Shell script that initalizes the five servers. It sends an IAMAT client command to Alford server to test that the message gets forwarded to all the server. Then it kills the Powell server and sends a different IAMAT client command to Alford server to make sure that the Bolden server still receives the message. Then it kills the Parker server and sends a third IAMAT client command to Bolden server to test that there is no longer a viable route to foward the message to the other remaing servers: Alford and Hamilton. Lastly, it sends a WHATSAT command to Hamilton server to perform a Google Places API call. Once all that is done, it kills all the alive servers.

4 Benefits and Drawbacks

For those familar with Python, the code development is straight-forward. One can easily adapt the given examples and search through the API documentation to implement the application server herd. Being that it is in Python, prototyping is quick and painless. The callback idea might pose some confusion who those without prior asynchronous programming experience.

5 Node.js Comparison

Node.js provides a platform for server-side and networking applications. It is largely implemented in JavaScript and follows the same event-driven networking paradigm. Node.js is relatively newer and therefore less mature than Twisted. However, in a short few years, it has gained tremendous popularity and growth. Many large corporations are using Node.js and the package repository has grown to include tens of thousands of packages. Like Twisted, Node.js syntax is very concise. Its arguable which is easier to learn and develop code in. Some say that Node.js is easier because the core functionality of Node.js is very small and thus easier to understand. Functionality can be incrementally added through libraries. I think that it also depends on prior exposure to the two languages and personal preference. A positive for Node.js is that it has very good performance, largely due to the performance of the V8 JavaScript Engine that it runs on top of. In the age of scalability and performance, this is a huge plus. Another appeal is the prevalence of JavaScript and JSON on the web. This really allows Node.js to slot into the overall web architecture. JavaScript Node.js should not be used in applications that require heavy server-side computation. Being that it is single threaded, this annuals all the throughput benefits offered by its non-blocking I/O. In this scenario, a threaded platform is as better approach. In summary, Node.js is a great alternative to Twisted and excels in certain areas.

6 Conclusion

Twisted provides a simple yet powerful framework for implementing the application server herd. The combination of rapid code development in Python and Twisteds APIs allowed for a quick and painless prototype implementation. However, I also think that Node.js provides a great alternative framework. Its rapid growth and adoption by many large companies speak for itself. Whether it be Twisted or Node.js, the application server herd architecture is a viable option for the traditional web server architecture.

References

[1] Schmidt, D., ”Reactor: An Object Behavioral Pattern for Concurrent Event Demultiplexing and Dispatching”, 1995.
[2] Noller, J., ”Twisted - Hello, Asynchronous Programming”, http://jessenoller.com/blog/2009/02/11/twisted-hello-asynchronous-programming, (Accessed: 31 May 2014).
[3] Wen, B. ”6 things you should know about Node.js”, (Java Word), http://www.javaworld.com/article/2079190/scripting-jvm-languages/6-things-you-should-know-about-node-js.html, (Accessed: 31 May 2014)