JavaScript (Node.js) vs. Python (Twisted):
Alternatives for building application server herds.

Jason Zhang
CS131

Abstract

Having written a Twisted prototype for application server herds, our task now is to evaluate the use of JavaScript, Node.js in particular, as an alternative for the same purpose. At the top level, we already know that both can be used for writing web servers using event-driven, asynchronous I/O. But by examining the source code of both frameworks, we hope to determine which of the two is more efficient or at the very least, compile a list of their advantages and disadvantages.

1. Introduction

Node.js was created by Ryan Dahl in 2009as a server side software system primarily aiding in the development of web servers. It utilizes several built-in libraries in addition to Google's V8 JavaScript engine and libUV. With the help of V8, compilation time of this JavaScript application is significantly shorter than those of other dynamic languages, such as Python.

Regarding evented I/O, there are have been two traditional ways of dealing with concurrent connection requests. Some web servers, such as Apache, use threads to handle these requests. Node and Twisted on the other hand utilizes a looping sine non-blocking thread. Since JavaScript is a language which applications are usually viewed through the browser, a single threaded event loop is a preferred choice. Passing around closures can be easily done by calling a function to perform some type of I/O and passing it a callback function.

Keep in the mind that the usage of Node.js extends beyond just web development. It may be used to construct IRC, http, and chat servers as well. For the purpose of this topic, we will be attempting to construct the latter so that we'll be able to compare the server setup process with that of Python's Twisted.

2. Implementation

In this section, we will go through the process of setting up a chat server with JavaScript's Node.js and compare each function call with a similar one found in Twisted.

The first step in creating a server is by declaring a variable to "require('net)". The require function acts as a retrieval method for the net module which contains methods for creating servers and clients.

To create a server, Node.js provides the command “net.createServer([options], [connectionListener])” which creates a new TCP server. “options” determines whether or not the socket will automatically send a FIN packet (a packet sent to indicate that the sender will no longer use the session to send or receive data) with true disabling the feature.An alternative would be to use the “end()” method. By default, “connectionListener” is set as a listener for the ‘connection event’.

The last step is to specify which port to listen in on and can be done using “server.listen(port, [hostname], [backlog], [callback])”. The latter two arguments being the maximum length of the queue of pending connections (defaulted at 511) and a listener for the ‘listening’ event respectively. Putting it all together:

```
varhttp = require(‘http);
var server = net.createServer(function(c) {
… do stuff…
})
server.listen(8124, function())
```

As shown, the listening functionality can be implemented with, at the very least, three lines of code using JavaScript while Twistedhas the added burden of creating a Factory class.

```
… createChatFactory class…
from twisted.internet import reactor
reactor.listenTCP(8124, ChatFactory())
reactor.run()
```

With the basics down, we can proceed to work on accepting clients. For this, we can initialize an empty array, “sockets”. Within the server creation function, we use “sockets.push(socket)” to add a client to the array. This is done similarly in Twisted in which “factory.clients” is initialized to an empty array and with each request, a client is added using “self.factory.clients.append(self)”. For disconnecting, Node.js calls on a combination of “socket.on”, the ‘end’ event, and “sockets.splice” (the full code can be seen in the prototype provided with this document). In this implementation, it takes three lines of code to remove a client. In Twisted, this is easily done with “self.factory.clients.remove(self)”.

The last feature needed for a chat server is receiving and distributing the messages. This is handled using another socket function which calls upon the ‘data’ event which is triggered specifically by the reception of data. A for-loop is used to traverse the array of clients and each one is sent the received message. Twisted operates in a similar way by using “for c in self.factory.clients: c.message(line)”. The part before the colon is essentially the equivalent of Node.js’ for-loop and the command after is what sends the message.

3. Results and Discussion

Implementing a chat server using either Node.js or Twisted is extremely easy due to their predefined functions which practically takes care of all the port listening and data parsing. When comparing the completed codes, I think it's important to note that while Node.js requires fewer lines to get the server up, Twisted is actually superior in implementing after-connection tasks such as message handling. What took Node.js a dozen lines of code took Twisted a single command.

With better documentation and fewer functions to familiarize with, using Node.js to set up a server was a whole lot faster. Twisted requires the users to be knowledgeable of its numerous classes such as (but not limited to) "LineReceiver" and "ServerFactory". But JavaScript's ease of use does come with a tradeoff as Python's included functions are a lot more versatile and robust.

In the interest of time, I've only designed a bare-bones chat server with both of these frameworks. The problem with that is that the simplicity of the server will not suffice in exploring every fault with both languages. My goal here has been to simply compare the ease and efficiency of setting up a server. Because we were able to create a server herd from building upon Twisted's chat server, I have no doubt that we can also do the same with Node.js, making it a viable candidate for building an application server herd.

4. Conclusion

It comes as no surprise that Twisted has more functionality due to the fact that it's been around for a lot longer. That in itself is a huge advantage of Node.js. However, it may be important to note that Node.js is faster and simpler to use, mainly because there aren't so many protocols at the user's disposal. Ironically, Node.js also has better documentation for beginning users, listing lines of important functions while describing each and every one. Twisted, on the other hand, relies on the user to peruse through its source code to figure out what tools are available. Ultimately, choosing which framework to use boils down to whether or not you have the time to browse through the documentation and whether or not you'll need a diverse array of functionality. If the answer is no to both of these questions, then Node.js may just be the right choice to build a rivaling application server herd.

5. References

[1] http://net.tutsplus.com/tutorials/javascript-ajax/this-time-youll-learn-node-js/
[2] http://en.wikipedia.org/wiki/Nodejs
[3] http://nodejs.org/api/net.html#net_net_createserver_options_connectionlistener
[4] http://stackoverflow.com/questions/3461549/what-are-the-use-cases-of-node-js-vs-twisted
[5] http://twistedmatrix.com/documents/11.1.0/core/examples/chatserver.py