Feasibilty of Using Twisted as a Framework to Application Server Herd

Ajan Jayant
University of California, Los Angeles

March 13, 2013

Abstract

The objective of this paper is to evaluate Twisted as a potential framework for developing a Wikimedia-style news service. It focuses on the programming language aspects of Twisted, specifically with regards to its language of implementaion, the prograaming paradigms it adheres to and the ease with which a sample prototype can be written in it. It also contrasts Twisted with othere popular frameworks, showing where it excels and where it lacks

1 Introduction

The motivation behind this assignment comes from a company who wishes to replace their current computing architecture because of problems related to the adoption of new technology(mobile phones). This creates a structural problem for the company, because Wikimedia uses the LAMP architecture, which implements a Linux (operating system), Apache HTTP Server, MySQL (database software), and PHP, Perl or Python, principal components to build a viable general purpose web server. From this definition, many of the companies problems with LAMP can be seen. Firstly, the Apache server works with HTTP protocols, so this limits the types of connections. Secondly, adding new servers that work behind the central server (master/slave) creates effecient write operations, but reading remains a bottleneck depending on the application server. Because of these reason, the application becomes to cumbersome to design, and users will find it inefficient and slow. One way to remove these drawbacks is to implement an thread-based framework, like the one provided by the Java standard library. While this has some scalability advantages, the code would be very difficult to write as there is not much documentation with regards to this application's specific purpose. As is, thread-based design would cause a host of other problems, such as deadlocks, race-conditions and synchroniztion failures in an applcation with as many constant updates as this. What is needed is another architecture, which implements better inter-server communication and allows quicker updates and faster access. The application server-herd is one such network.

2 Application Server Herd and Event-Driven Programming

The application server herd is an architecture we use to replace LAMP. They allow servers to communicate internally with each other based on certain rules. They can also communicate with clients, and then relay all the information they glean from them to each other. Because of the constant nature of these interactions, a pardign based on handling these constant, repeated interactions should be used. The event-based model is one such paradigm. In computer programming, event-driven programming (EDP) or event-based programming is a programming paradigm in which the flow of the program is determined by eventse.g., sensor outputs or user actions (mouse clicks, key presses) or messages from other programs or threads. The central idea around the event-driven model is that the user provides the code for certain evernts, and deals with them as and when they occour. This makes code design simple, and, with a well-implemented framework, very efficient.

3 Twisted: Language Features

Twisted is one such event-driven framework which is used to implement such an application. Twisted is written in Python, and so shares all its language features. As such, in order to understand how twisted works and evaluate its feasability, its underlying Python implementation must be understood. Python is a multi-paradigm language which is interpreted rather than compiled. Almost uniquely, its lexical structure is such that indentation defines different levels of control flow. Python has a strong, dynamic type system. One of the main paradigms of Python is that everything's an object; as such, functions are also objects and can be passed as-is to other functions. Python has many useful sequence types, including tuples and lists; the former is immutable, while the latter is mutable. Python's memory management uses reference counting to detect inaccessible objects, and a separate mechanism to collect reference cycles, periodically executing a cycle detection algorithm which looks for inaccessible cycles and deletes the objects involved. The gc module provides functions to force garbage collection, obtain debugging statistics, and tune the collectors parameters. All these features are implemented are important when considering how easy it is to design the application using Twisted.

4 The Prototype

4.1 Overview

The prototype implements 5 servers, which have specialiazed communication patterns and can respond to various commands as specified below. The prototype makes heavy use of inheritance, subclassing LineReceiver to create a protocol and Factory to create a server. Apart from these classes, there are statements that take system argements, which consist of the server name, host name, port number, create a new server and run the reactor.

4.2 ServerHerdProtocol

This class is used to write protocols which allow for various functionality when trying to run commands. The class has the following methods sendPeer, lineReceived, connectionMade, connectionLost, handleIAMAT, handleAT ,addPeer, handleWHATSAT, findTweets, handleUnknown and sendMessage. Each function of this protocol adds an ability to respond to a certain event; they all are event-handlers. connectionMade and connectionLost are all overriden methods, and simply log info as mentioned in the specification. lineReceiver takes input to the server and calls the other methods based on the input. The other methods are described below.

4.3 ServerHerd

This class is used to write servers, and to implement the logging functionality described by the specification. Each instance will ahave a set of attributes that describe serverName, hostName, portNumber, peers, clients and logstream. There is a function buildProtocol that builds a serverHerdProtocol and logging functions that log the given information.

4.4 IAMAT command

The IAMAT command is triggered when the client sends a message of the format: IAMAT kiwi.cs.ucla.edu +27.5916+086.5640 1353118103.108893381 This input is parsed and put into a tuple,with each individual element in its own variable. The current time is calculated using standard library functions time.time(), and the difference between the current time and the time sent by the the client is calculated. The output is then sent back to the client with the AT prefix. The client data is stored, so that the server can respond correctly to the WHATSAT message when it is sent.

For each peer in the server's list, the AT message is propagated using the simple flood algorithm implemented by a for loop. An important feature of python on display here is its object-oreiented nature. addcallback, using a deffered refrence, is passed a function with no argements and a peer. This allows it to send the message to the server's peer whenever it is free, and thus all the servers manage to get updated

4.5 WHATSAT command

The WHATSAT command takes a message of the form: WHATSAT kiwi.cs.ucla.edu 100 2 and using previously received client information finds all the tweets within a certain radius of the client. It is important to note that the clients were implemented as a dictionary,displaying the breadth of Python's types, and that static type casting was used to cast string input into the desired integers. The findTweets method handled the duty of returning all the tweets.

4.6 Handling Peers

Peer handling is doen by adding a different event to the protocol. If the client sends a message 'peer' with the requisite data, namely the peer, protocol, host name and port number, an endpoint is setup, a connection is implemented and the adding server's data is added to the receiving server's attributes.

4.7 Using Twitty Twister

Finding the tweets was the most complicated part of the prototype. By going through the test directory, functions that returned tweets was discovered, and by going through the twittytwister directory and through twisted's own functions a function was able to be written using addcallback with send message passed to it.

5 Twisted: Advantages and Drawbacks

Twisted, and Python by extension, had the following advatages when designing the application.
1. Both Twisted and Python have extremely well-documented code. The API refrence of both provide enough functionality when desigining the application. In fact, a large majority of the code was taken from the two twisted tutorials on writing a server and writing a client. The extensive examples such as chatserver.py also helped greatly.
2. Many Python language features mad writing code very fast and simple. The lack of curly braces facilitated fast typeing; the use of dynamic typing also leads to the application being written quicker than before.
3. Twisted is perfectly suited for the event-driven model, as it provides lots of inheritable classes while giving us the freedom to write our own event handlers, leading to 'elegant', readable code.
Some of the drawbacks are: 1. Memory management, while easier than C, puts a lot of onus on the user, to avoid cyclic references and to recollect memory when python is out of it.
2. The twitty twister library, because of a lack of suitable examples and documentation, is very difficult to understand
3. The dynamic typing, which facilitates quicker writing of code, can also make it harder to understand for someone trying to implement something similar

6 Comparison to Node.js

Node.js has 1. A very good package management system
2. Very fast execution time
JavaScript has no built-in class features but is flexible enough to accommodate the approach.
When speaking of javascript, inheritance is prototypal. It is designed more around object factories than cascading layered-class inheritance schemes (which are an antipattern in any language, IMO). It is def-

initely more arguably a functional language. It also just started being used outside of the client-side web. So implementing servers with Node.js might be a bit challenging. Therefore Node.js definitely is a viable alternative, but I feel Twisted would provide an easier implementation

7 refrences

[1] http://twistedmatrix.com/trac/
[2] http://nodejs.org/api/
[3] http://twistedmatrix.com/documents/current/core/howto/servers.html
[4] http://effbot.org/pyfaq/how-does-python-manage-memory.htm
[5] http://docs.python.org/2/reference/datamodel.html
[6] http://twistedmatrix.com/documents/11.0.0/core/howto/defer.html
[7] http://hathawaymix.org/Weblog/20040616/
[8] http://www.nedworks.org/ mark/presentation