The Python Programming Language and the Twisted Framework

Geomar Manzano
University of California, Los Angeles

Abstract

Twisted is an event-driven network programming framework written in Python, making it a potential candidate for implementing a Wikipedia-style application server herd. A prototype using Twisted will be examined to determine its viability. There are also other options in order to implement a Wikipedia-style server such as Node.js.

1 Introduction

LAMP is a model of web service solution stacks, which is essentially a software bundle, originally consisting of four open-source software: Linux, Apache, MySQL, and PHP. Linux is used as the operating system, Apache as the web server, MySQL as the relational database management system, and PHP as the scripting language. Some sites such as Wikipedia use the LAMP model. However, in some situations the LAMP model is unfit so an alternative must be found. For this reason, this paper will determine the viability of an application server herd through the use of the Twisted framework. We first examine the Python programming language which Twisted is based on and then examine Twisted itself using a prototype for an application server herd.

2 The Python Programming Language

The Python programming language is a relatively new and widely used general-purpose language that is used in many of today's applications. The syntax of Python is very simple and clean making it easy for both readability and writability. In fact, simplicity is such a core idea that if you type in `import this` on the Python interpreter, you get a series of sentences which advocate for simplicity. Other than being a simple language, there are many features in Python that allow for faster development times and an increase in productivity.

2.1 Multiparadigm

The Python language is very flexible when it comes to writability. One may choose between the imperative, functional, and object-oriented styles or even mix and match the three together. As an example, suppose that we wish to find the maximum from a given set of integers and append the result of the maximum multiplied by some intenger *n* to the end of the list. The following is a demonstration on how to accomplish this using each of the three paradigms:

```
# Imperative Style
def imperativeMax(lst, n):
  max = lst[0]
  for number in lst:
    if number > max: max = number
  return lst + [max * n]

# Functional Style
def functionalMax(lst, n):
  return lst + [n * max(lst)]

# Object-Oriented Style
class IntegerList:
  def __init__(self, lst):
    self.lst = lst
  def objectOrientedMax(self, n):
    self.lst.append(n * max(self.lst))
```

Since Python is multiparadigm, programmers from different backgrounds can easily pick up the language and start programming in the style that they are most familiar or comfortable with. For this reason, learning Python should be incredibly quick if one has been exposed to other languages before.

2.2 Type System

Python is a dynamically-typed runtime language. Dynamic typing is when type checks are left until runtime. In general, dynamically typed languages allow code which is more expressive than type systems that employ static type checking [5]. There is a cost though to being dynamically typed, and that is a sacrifice in performance at runtime. For statically typed languages, type checks are usually done at compile time so checks do not need to be done when a program is in execution. A dynamically typed language would perform type checks at runtime, consequently making it slower than statically typed languages. However if speed is not an issue, whih may be the case in some applications, than the trade-off may be worth it.

In addition, Python is also a strongly typed language as the interpreter keeps track of all the variables types. In a strongly typed language, you are simply not allowed to do anything that is incompatible with the type of data you are working with [2]. A variable may be used only in ways that respect its type - for example, it is impossible to perform double operations on an int value or to use a boolean value as an integer [6]. For this reason, Python is a relatively safe language compared to weakly typed languages such as C and C++.

Duck typing is another feature of the Python language. This is a feature that is only available to object-oriented languages. In duck typing, an object of a particular class is compatible with a method if it is able to supply all the methods asked of it by the method or function at runtime. As an example, if an object walks like a duck, swims like a duck, and quacks like a duck, then that object may be called a duck. The advantage of duck typing is increased flexibility as well as being able to handle changing interfaces easily. The disadvantage is that bugs may be harder to catch. For example, if some object is able to act like a duck, but is not actually a duck then a problem will arise.

2.3 Scope and Bindings

Python is a statically scoped language. Static scoping is when the scope of a variable can be statically determined - that is, prior to execution. This permits a human reader and a compiler to determine the type or value of every variable in the program simply by examining its source code [4]. In contrast, a dynamically scoped language such as the original LISP would need the human reader and a compiler to trace through the execution of a program to determine the type or value of a variable.

Python uses dynamic type binding for its variables. With dynamic type binding, the type of a variable is not specified by a declaration statement, nor can it be determined by the spelling of its name. Instead, the variable is bound to a type when it is assigned a value in an assignment statement [4]. This feature of Python greatly increases its flexibility. Simplicity and flexibility are the core principles of Python which it definitely achieves.

2.4 Memory Management

In Python, all variables are simply references to objects on the heap. This is incredibly important to take note of when programming in Python. Each time a new value is generated by running an expression, Python creates a new object, which is allocated on the heap, to represent that object. Each object also has two standard header fields: a type designator used to mark the type of the object, and a reference count used to determined when it is allowed to reclaim the object through the use of the garbage collector [1]. As an example, consider the following example:

```
listA = [[1, 2], [3, 4]]
listB = listA
listB[1][1] = 22
```

By doing the above, we arrive at the situation given by Figure 1.

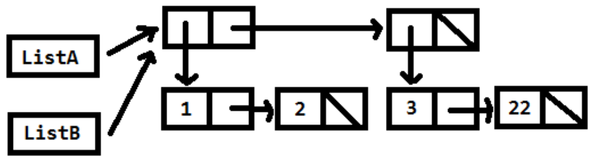

Figure 1: List Example

Notice that the object referenced by listA is modified. This is due to the assignment statement in the code above which allowed listB to be able to reference the object referenced by listA. As a result, listB is able to modify the object's values. Now recall that each object in the heap has an associated reference count. If the reference count for an object were to drop to zero, which will happen if both listA and listB no longer reference the object in the example, then Python's garbage collector will automatically deallocate the memory space occupied by the object with no explicit action needed from the programmer. Interestingly, Python has two levels of garbage collection: the first level being the reference count method and the second being the mark and sweep method.

2.5 Concurrency

Python supports concurrent programming although it is not as great as compared to Java or Scala in which both have extensive support for concurrency. In addition, Python has difficulty with multithreaded work due to the global interpreter lock which prevents multiple threads of Python code from running simultaneously [3]. An interpreter that uses a global interpreter lock always allows exactly one thread to execute at a time, even if run on a multi-core processor. For this reason, it is probably best to choose another language when planning to do concurrent programming. However there are some implementations of Python that do try to make improvements on concurrency.

3 The Twisted Networking Framework

Twisted is an event-driven networking framework written in Python. It supports several protocols such as HTTP, XMPP, NNTP, IMAP, SSH, IRC, and FTP. Since Twisted is based on the event-driven programming paradigm, users of Twisted write short callbacks which are called by the framework [7].

4 Implementation of the Server Herd

The prototype, written in Python using Twisted, instantiates a server per connection. In the prototype code, the reactor's role is to listen in on the given port number given to listenTCP as well as call the protocol of the server.

4.1 IAMAT Implementation

The IAMAT command is used to tell the server the location of a client. There are four arguments to parse: the command IAMAT, the client's ID, the location of the client, and the time of the client during which it sent the IAMAT message. We first check if the number of arguments passed to IAMAT is indeed four to continue. Afterwards we setup the necessary response string to respond to the client. Lastly, the response string is sent to the neighbors of the server which is found in the global variable SERVERS.

4.2 WHATSAT Implementation

The WHATSAT command is used as a query for clients to find information about places near other client's locations on the server. There are four arguments to parse: the command WHATSAT, the client's ID, a radius from the client, and the upper bound of information to receive from Places data within the given radius of the client. For this method, an API call to Google places is necessary. This method also requires the use of the API key. Afterwards, the amount of information is returned in a JSON format.

4.3 AT Implementation

The AT command is used by servers acting as clients to update information about a client. Firstly, a check is done to determine if a client already exists in the server. The time stamp of the client is checked against to determine if a message is a duplicate, which will return if this is indeed the case. If the message is not a duplicate, the server will update the information concerning the client and forward the update to the neighbors of the server.

5 Comparison with Node.js

Node.js is an event driven I/O server-side JavaScript environment. Similar to the Twisted framework, it is also asynchronous and is designed for web applications. Since Twisted was released seven years earlier than Node.js, Twisted is much more mature. Both also rely on callbacks since they are both event-driven.

6 Conclusion

The features that Twisted provides allows us to easily implement an application server herd. Since Twisted is built on top of Python, it is quite easy to develop network applications rather quickly due to the simplicity and flexibility of the Python language itself. In addition, there is a considerable amount of documentation on how to use Twisted due to its mature status. In sum, the application server herd model is definitely a viable alternative to the LAMP model.

References

[1] LUTZ, M. *Learning Python*, fifth ed. O'Reilly, June 2013.

[2] PYTHON, W. Why is python a dynamic language and also a strongly typed language. `https://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language`, Feb 2012.

[3] PYTHON, W. Global interpreter lock. `https://wiki.python.org/moin/GlobalInterpreterLock`, Jan 2015.

[4] SEBESTA, R. *Concepts of Programming Languages*, tenth ed. Pearson, Jan 2012.

[5] TRATT, L. Dynamically typed languages. *Advances in Computers 77* (July 2009), 149–184.

[6] UNIVERSITY, C. Safety and strong typing. `http://www.cs.cornell.edu/courses/cs1130/2012sp/1130selfpaced/module1/module1part4/strongtyping.html`, 2005.

[7] WIKIPEDIA. Twisted. `https://en.wikipedia.org/wiki/Twisted_%28software%29`, Oct 2015.