CS 131 Homework 6. Comparing scripting languages

Braden Anderson
University of California, Los Angeles

Abstract

In this paper I'll assess the suitability of using Twisted, the web framework written in Python, for implementing a Wikimedia-style service. I'll contrast Python with Java, and the Twisted framework wth Node.js. Overall, Twisted's ease of protoyping and potential scalability make it a very good candidate for this project.

1 Introduction

Twisted is an asynchronous, event-driven network programming framework. It has been in continuous development since 2002, and it is one of the most robust event-driven frameworks available for Python. I've developed a prototype in Twisted that approximates a server herd, simulating what it would be like to be running Twisted in a multi-server environment. I found that prototyping with Twisted was extremely easy because of Python's ease of use and Twisted built-in modules.

My prototype servers accept connections over TCP, and messages sent to servers are propagated throughout the entire herd via a flooding algorithm. I also included functionality for interfacing with Twitter's API, but because of recent changes in their API I've chosen to mock out that functionality instead of implementing it fully.

2 Notes on implementation

My implementation emphasizes scalability and failure tolerance. Messages are flooded through the herd according to the neighbor graph supplied to each HerdServerFactory. A new node can be easily added by creating a new factory instance, allocating a unique port number, and passing that information to the TCPServer constructor.

I designed connections between neighbors to maximize failure tolerance. Rather than maintaining an always-on connection between neighbors in the herd, I create a Client dynamically whenever a message needs to be sent from server to server. The Client expires after the connection is made and the message is sent. This maximizes reliability over an unreliable network. However, if a Client fails to connect to a Server, the message is not sent and that neighbor is not updated (unless it is connected to later in the flooding algorithm).

Logging is done with a single log file, `herd.log`. It is implemented using Python's `logging` module running in DEBUG mode. Each line in the log file includes the name of the server that's doing the logging. Logging includes new connections, lost connections, received messages, sent messages, and forwarded AT commands.

The responses from Twitter are mocked out currently, because Twitty Twister no longer works with their API and the newest versions of the Twitter API require OAuth authentication. The results-per-page argument to `WHATSAT` can be any positive integer, and the radius argument must be positive and at most 100.

My implementation can be viewed at `github.com/bradencanderson/CS131/`.

3 Suitability of Python

There are numerous points of contrast between Python and an enterprise-friendly language such as Java. The main theme here is that Java enforces the creation of good, safe code, but sometimes it can be very heavy-handed and actually limit productivity.

There are numerous areas of concern:

Duck typing

This is both a pro and a con for Python. A lack of

static type checking increases the frequency of runtime errors. Duck typing also requires the programmer to have a deeper understanding of the code base – a Java developer can see that members of class `foo` are not the same as members of class `bar`, but a Python developer would need to investigate the code base to obtain an intuition for how types are used. Another alternative for the Python developer is to use a debugger to learn more about the program at runtime, or to use code analysis tools such as pylint or pychecker.

On the other hand, duck typing makes it far easier to deal with changing interfaces. Java generally requires lots of boilerplate code to initialize objects and pass them around, while Python doesn't. This is especially apparent with names arguments, `*args`, and `**kwargs` – there is a lot more flexibility that comes with Python's dynamic typing, but that flexibility comes at a cost. In general static type checking is meant to protect the programmer from making buggy code, but that same buggy code could be prevented by designing high-quality APIs.

Lazy evaluation

This is another tradeoff. Lazy evaluation has a lot of potential for faster, more flexible programs. However, lazy evaluation means that some code paths will be evaluated very rarely, leading to some bugs that are hard to track down and fix. Early detection of bugs ("fail fast") is a very big win for eager evaluation.

Interpreted language

This is a win for Python because it allows developers to iterate much faster. Developing from within an interpreter rather than an IDE is far superior because it is an opportunity for "REPL-driven development" – programmers can throw a breakpoint into their program and execute Python expressions from within the interpreter, working to solve the problem at hand. This style of development is far faster than what is possible in Java, since the Java debuggers are far less robust and generally don't support REPL.

Memory management

Memory shouldn't be an issue here, because it's so inexpensive. Additionally, since Wikimedia isn't a real-time system, real-time garbage collection (or manual memory management) isn't necessary, so Python and Java's memory management should both be fine. It also looks like Python and Java programs have similar memory footprints[1].

Multithreading

Python does support multithreading, however we won't be using it. CPython only uses a single core for most tasks, and Python's memory management does not behave well with threads – introducing the need for the "global interpreter lock", which makes it impossible to execute Python bytecodes truly concurrently[2]. Multithreading is generally reserved for asynchronous I/O, and multiprocessing is generally required to see the benefit of a multicore CPU. Java has far greater support for multithreading than Python.

Testing

Python has robust support for all kinds of tests, including unit tests and integration tests. As with Java applications, testing helps make sure that code isn't broken. In Python, testing is especially important because of the dynamic typing and lazy evaluation – the program often changes dynamically at runtime. Because of this, Python development requires very disciplined testing practices.

4 Suitability of Twisted

The Twisted framework is ideal for implementing an application server herd. Here I'll go over specific design elements that make Twisted suitable.

Deferreds

Twisted offers robust support for event-driven programming. This lends itself to the web, where some operations can be very slow (sending a message to a foreign IP), and their behavior is unpredictable. Twisted allows thinking about these operations in terms of aynschronous events, so a Deferred can be ignored until it triggers an event.

Single-threadedness

Most things in Twisted run in a single thread. This simplifies concurrency issues greatly. The single-threadedness can be circumvented by running servers in multiple processes – possibly on a single machine, or distributed throughout a network. Running a server herd can cause many different concurrency issues, for example syncing of data (mostly solved by our flooding algorithm) and data duplication. Because of the way the herd prototype works, it's extremely difficult to make all `IAMAT` commands atomic, so there are data race issues stemming from that.

4.1 Twisted v. node.js

Node.js is a framework for writing server-side web applications in Javascript. Like Twisted, Node.js is also

heavily asynchronous and event-driven. Twisted supports many different forms of network protocols, making it a good framework for developing an application that needs to communicate over multiple protocols. Twisted is also a far more mature framework than Node, since it is much older.

Overall, it seems that Twisted is a very good option for implementing a Wikimedia-style server herd. Twisted is a robust framework that produces highly scalable applications, and it's already used in production code at places such as Twilio, Apple, and Canonical.

References

[1] Java 7 vs Python 3 quad-core x64 comparison. http://shootout.alioth.debian.org/.

[2] https://wiki.python.org/moin/GlobalInterpreterLock