Is Node.js Becoming a Part of the Stack? SimpleGeo Says Yes.

Tuesday, February 22, 2011 at 9:05AM

Todd Hoff in Example, Strategy, nodejs

This is an interview with Wade Simmons, an Infrastructure Engineer at SimpleGeo, a service making it *easy for developers to create location-aware applications*, on their increasing use of Node.js as a backend service component, replacing code that would have at one time been written in Java, Python or Ruby. Node.js is finding it's way into many stacks these days and I was curious why that might be. My experience writing several messaging systems is that programmers don't like the async model and it's a big surprise that a pure async programming model like Node.js, especially one that uses server-side Javascript, would be taking off. Wade was generous enough to help explain their reasoning behind using Node.js at SimpleGeo. I'd really like to thank Wade for taking the time for this interview. He did a really great job and provided a lot of insight on how the modern web stack is evolving in the crucible of real-life experience.

And here begins the interview with Wade Simmons:

At SimpleGeo we have been fans of event driven, non-blocking network servers since the beginning. Our core API server was written in Python on top of the Tornado web server (http://www.tornadoweb.org/). We also have internal servers that are written using Twisted (http://twistedmatrix.com/). Event driven frameworks make a lot of sense when your server is going to be mainly IO-bound, and a lot of our internal services are basically wrappers around different types of indexes and databases.

You are probably wondering why we are trying a different event based framework now when we have already developed and deployed code in others. The answer is there are some drawbacks in the other frameworks that we are hoping that Node.js may avoid.

Languages Not Designed to be Async Will Block

The primary problem that we had with Tornado is library support. Tornado provides a non-blocking HTTP client, but support for other protocols is limited or non-existent. This means that it is very easy to be tempted to use the plentiful libraries available for Python. Once this happens, you inevitably start to block the event thread and concurrent connections to the server will start to block on each other. This means you get the worst of two worlds: a single threaded server with none of the potential benefits.

Twisted Doesn't Mesh Well with Python

The Twisted framework for Python has much better library support, as many protocols have been implemented for it. There isn't as wide support as we would like, but at least it is much better than Tornado. Quite a few people at SimpleGeo weren't fans of Twisted though because it doesn't mesh well with the rest of Python. For example, the naming conventions for methods are camel-case instead of of the python standard "lowercase with underscores." The learning curve for Twisted can also be a little steeper as there are less examples to find online (although the ones that do exist are very nice and a few good books are available as well).

Cooperative Multitasking Model Easier to Debug

We also considered coroutine libraries such as Eventlet (http://eventlet.

net/). There is much debate within SimpleGeo about whether cooperative multitasking or coroutines are a better style. The team that decided to go ahead with using Node.js likes the cooperative method as it makes debugging concurrency issues much easier to test. You know exactly where your program will yield control to other tasks, so tracking down issues that have to do with shared state is much easier.

Async Browser Experience is being Transferred to the Server

I think one of the primary reasons that Node.js is getting traction is that people are used to the asynchronous programming style with JavaScript already. Even programmers that mostly do backend work probably dabble at bit with client-side JavaScript from time to time. JavaScript isn't the perfect language, but its functional abilities are better than most and there many documented ways for avoiding its pitfalls. Once you leave the world of the browser and no longer have to worry about backwards compatibility the good parts start to shine through.

Used Mostly as Glue Between Services

Currently, our primary use of Node.js is for the entry point to our API. We have many different internal servers, and our different API endpoints need to hit one or more of them for each request. The Node.js server does authentication, makes parallel internal requests then combines the results into the requested output format. This server also takes care of recording stats and logging detailed request information for later parsing.

Easy to Debug, Test, and Performs Well

So far, our experience has been that writing Node.js code has been very intuitive and also easy to test. Production deployments have been very stable and we have had no memory leak problems or other issues that you would expect with such a young platform (knock on wood). We have plans to use Node.js internally for a few other "glue" style pieces of our framework; components that mostly act as routers/translators between backends and clients. These are components that have or would have written in Python with Tornado / Twisted. Python still has plenty of uses in our stack, but Node.js is starting to seep in as we become more comfortable with it.

No Plans to Implement CPU Intensive Services is Node.js

We don't have any plans to do any CPU intensive stuff in Node.js. We are working on a database layer in Java on top of Cassandra and we will probably stick with that for the time being. We are implementing the spatial logic there instead of on remote network servers so we can do as much processing right next to the data as possible. I don't believe that there is enough control over the memory model in v8 (and some of the data structures can actually be pretty bloated) for doing anything intensive like a database in Node.js.

Code Contributed by SimpleGeo

Even though Node.js is young, the community has grown really fast and there are a large number of libraries to be found on github and npm (the Node package manager). SimpleGeo loves open source, so when we have found libraries lacking or missing we have made sure to release all of our improvements so that others may use them.

A few examples of libraries that have been created from our work:

- https://github.com/wadey/node-thrift
- https://github.com/wadey/node-microtime

- https://github.com/wadey/decimaljson

There are many more to be found on https://github.com/simplegeo.

Related Articles

- Quora: What language is SimpleGeo's API platform coded in?
- HackerNews Thread
