WhitePages Rebuilds Core Parts of Application Stack with Scala and Akka to Improve Scaling

Replacing Perl and Ruby applications with Scala-based code and an Akka runtime environment, the WhitePages team delivers order of magnitude improvements in application speed and hardware utilization

About WhitePages

WhitePages is the leading provider of contact information for people and businesses in the U.S. With over 50 million unique monthly users and powering over 2 billion searches per year across sites that include WhitePages.com and 411.com, WhitePages offers consumers one-click access to more than 200 million adults and also provides them with the ability to edit and control their own listings. The company's suite of mobile products includes a top-50 mobile website and popular Android and iPhone apps with over 18 million active monthly mobile users.

The Challenge

WhitePages relies on an extensive service-oriented architecture that includes dozens of in-house and third-party services running on hundreds of servers and incorporating multiple software languages and frameworks. Scaling legacy languages in some of those frameworks began to be difficult and costly. "Our hardware costs to support Perl and Ruby-based technologies were ridiculous. We had a service that had grown to 80+ servers to handle 400 QPS," says Devin Ben-Hur, Senior Architect for WhitePages "And we were incurring significant latencies on services that relied on single-threaded applications written in those languages." Even using C-language plug-ins to try to speed up the slowest process, servers running Perl and Ruby processes that demanded higher throughput were spending 70% of their time serializing and deserializing data.

In general, WhitePages wanted to move to a Reactive development and application environment to enable easier development, iteration and scaling out of newer application components while simultaneously shrinking the required hardware and compute capacity footprint. WhitePages needed:

Revision 1.0

- **More efficient hardware utilization:** WhitePages wanted to reduce the number of boxes it was using to cut costs, reduce power consumption, and minimize operational and infrastructure management complexity.
- **Evented, asynchronous I/O processes by default:** to alleviate throughput bottlenecks, improve memory usage, and reduce latencies caused by non-reactive processes
- **Improved development capabilities**: allowing development on laptops rather than remote Linux servers allowing rapid iterations in the developer workflow

The Solution

Ben-Hur and WhitePages Director of Software Engineering, Robert Noble, spent time evaluating how to shift bottlenecked and costly portions of the WhitePages architecture away from legacy languages to a more modern, reactive framework. Ben-Hur considered Node.js, Clojure, Java, Erlang, and Go-lang. But he quickly narrowed down consideration to Scala. The other choices were eliminated because of:

- Ease of transition for experienced Ruby developers to a new programming language (Scala shares many of the Ruby coding idioms, whereas the learning curve for the other options was much steeper.)
- Maturity of the Reactive Asynchronous Programming Framework—the Akka framework offers most of the features of the most mature implementation (Erlang), without much of the accompanying baggage, such as a weak runtime and an awkward syntax.
- Enforced strong type system, eliminating a large class of errors which can be caught at compile time. (a weakness of Node.js, Clojure, and Erlang; this is less true of Go-lang and Java/).
- Maturity and completeness of available libraries. Running on a JVM brought a world of proven Java components.
- The size and quality of the community of practice—people openly using it, discussing it, and improving it. (These were marks against Node.js, Erlang, and Go-lang.)

"We wanted something that handled the reactive pattern well and would be both resilient and multi-threaded," explains Noble. "You could do it in Java but coding in Java is fairly cumbersome. Someone suggested Scala, and from there we fell in love."

The WhitePages team particularly liked the maturity and expressiveness of the asynchronous programming patterns supported by Futures and Actors provided by the Akka framework.

The Results

WhitePages began rebuilding core components of its stack, such as the search coordinator, with Scala, Akka and the Spray framework. "In Ruby and in Perl, it's possible to do evented I/O architectures but you end up with awkward coding patterns. Your libraries may not be compatible with evented architectures," says Ben-Hur. "With Akka, we got a Reactive, evented model out of the box. Using Futures and Actors was very easy." Using Akka for one crucial, high-throughput service, WhitePages reduced the number of servers required from sixty to five with improvements to latency, stability, and consistency. "The cost savings from that alone are tremendous," says Noble. "We can use that hardware for other things and also scale out more easily because scaling costs are so much less."

With the Typesafe stack WhitePages got:

- Radically more efficient hardware utilization on the most resource intensive services, including a 15x reduction in server count.
- A Reactive, event-driven, asynchronous non-blocking software stack right out of the box with no modifications required. This resulted in improvements of as much as two-orders of magnitude (at the 99th percentile) in latency on network-bound application processes.
- A more productive local test and development environment through the superior abstraction capabilities of the JVM that is at the core of Scala apps.

“With non-Reactive, non-evented languages built around process models, we tied up big chunks of memory and were limited to how much concurrency we could get on a server. With Akka and Scala, concurrency is just there. It gives us much better flexibility and a more efficient code base.”
Devin Ben-Hur, WhitePages

The WhitePages team is continuing to replace more of its diverse stack with Scala-based applications. Mobile backend services, B2B API services, and data ingestion and reasoning services are all slated to be moved to Scala/Akka from older programming languages. WhitePages found, too, that moving to Scala attracted great talent. “Scala is a language coders want to program in,” says Noble. “Java is a language they typically come from—but not one they enjoy working with. It’s a great recruiting tool for higher caliber programmers. And even more reason we’re glad we made the move.”

*The **Typesafe Reactive Platform** is a modern software platform that makes it easy for developers to build scalable software applications. It combines **Play Framework**, **Akka** runtime, the **Scala** programming language, and robust developer tools in a simple package that integrates seamlessly with existing Java infrastructure. Commercial support and maintenance is available for the Typesafe Reactive Platform through the **Typesafe Subscription.***