Typesafe

Bringing Reactive Applications to the Java Virtual Machine

Flowdock scales messaging with Akka

Utilizing familiar tools to build a revolutionary new team collaboration product dictated looking outside the development teams' comfort zone for some highly performant and scalable platform components. Flowdock worked with Scala and Akka from Typesafe to address that need.

About Flowdock

Flowdock has been built and developed with love in Helsinki, Finland since 2009. The company's goal is to make a tool that transforms teams into a singular thinking organism with one memory and brain melding its members; and if that doesn't work, they at least want to make working in a team a wonderfully seamless and highly productive experience.

The Business Problem

Having begun life as a Ruby on Rails consulting shop in 2005, Otto Hilska had experienced first hand the difficulty in communicating quickly and effectively across a distributed team of busy consultants. Drawing from this experience, Otto conceived a vision for a low-friction collaboration tool that would act as a shared team inbox with group chat. Teams using the tool would be able to stay up-to-date, react in seconds instead of hours, and wouldn't miss a thing. By integrating activity from project management tools (Pivotal Tracker, JIRA), version control systems (GitHub, BitBucket, Kiln), customer feedback channels (Zendesk, email lists) and many other sources into an easily consumable stream, teams can then work the issues together, and react in seconds.

Flowdock was born.

Choosing the Tools

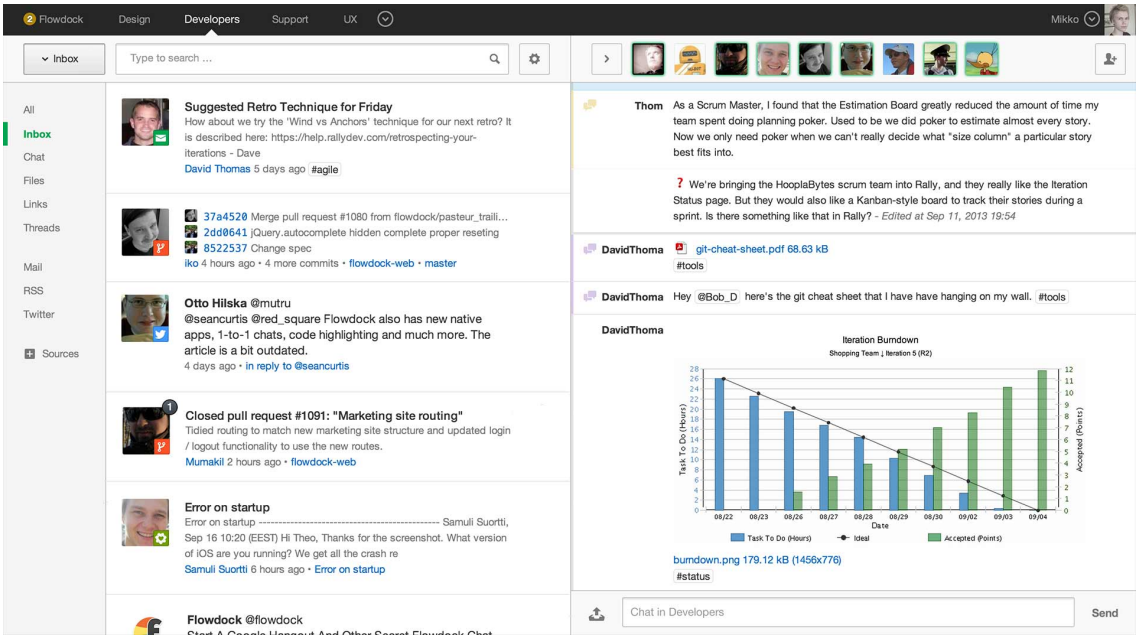
Given the team's extensive experience in Ruby on Rails, Otto and his team decided that Ruby would be a good toolset to use for building the majority of this new product. Since Flowdock is primarily a "single page" application, it also made sense to use client-side JavaScript and CoffeeScript extensively. Behind the

info@typesafe.com

Revision 1.0

front-end Flowdock application, on the server lives a highly modular architecture that consists of forty-plus services and components.

One of the first services that needed to be built, however, was *not* a candidate for Ruby. The core-messaging module was clearly a component that would need to be highly performant and allow for seamless scaling as the projected traffic that Flowdock's server side infrastructure would have to handle was substantial.



Otto looked at the available technologies in the marketplace at the time:

- Node.js was just becoming available, however, the 1.4GB heap size limitation was, and still is too constraining to be practical.
- Erlang looked appealing, however, the lack of Java interoperability and proprietary runtime made it an untenable solution. Erlang did, however, get Otto's team interested in the concept of the Actor model, which they felt would be a good programing model to utilize.

The desires to live within the familiar world of the JVM, and to utilize the Actor model lead Otto and his team in the direction of Scala and Akka from Typesafe. Scala is a general purpose programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It smoothly integrates features of object-oriented and functional languages, enabling Java and other programmers to be more productive. Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications. These two components used hand-in-hand allow developers to build massively scalable, highly concurrent applications.

Nobody within the Flowdock team had prior experience with Scala, so they learned it as they built out the core-messaging service. While the scope of the service changed over time, a small team of three developers deployed it to production within a couple of months, and then continued development iteratively, totaling nine months in all. The core-messaging service comprises approximately two thousand lines of Scala code in all, much of which is attributable to Scala's concise nature and the rich functionality of the Akka framework. This allowed the Flowdock team to focus on the value-add business logic that they needed to create, rather than worrying about managing low-level plumbing tasks.

The Outcome

While the core-messaging team worked in Scala and Akka, the rest of the team worked on the client and other server components in Ruby.

The messaging service has been running reliably for more than three years already with no real issues. It can handle 1700 messages/second routinely and can scale far beyond that if necessary.

It's the reliability of the whole thing - that's how its success is measured for me. Flowdock is a team communication tool that people spend, on average, more than seven hours a day actively interacting with team members in. If it's down, people get really annoyed.

Otto Hilska, Founder, Flowdock

Most of the downtime Flowdock experiences is due to networking issues – even the connections inside the U.S. have proven to be quite fragile. To work around those issues, the Flowdock team has built a large infrastructure of servers and redundant networks across the U.S. and around the world. Should a connection between any two servers go down, messages are routed through the redundant service. The Flowdock network infrastructure is built upon Amazon, Rackspace and some dedicated hosts.

Our goal is to recover from any issue within ten seconds.

Otto Hilska, Founder, Flowdock

Conclusion

Otto feels that the Typesafe Reactive Platform was a good choice, since the JVM ecosystem is well known, and there is a plethora of third-party libraries to leverage if need be. A good example of where this interoperability was critical was with their processing service that sanitizes HTML and ensures that it's safe to show to a user. If an image comes in on a stream and it contains HTML, Flowdock automatically strips out the script action and any other potentially dangerous content. This functionality is notoriously difficult to get right, so the team wanted to leverage existing libraries to help them. Had they used Erlang, they would have had to write that functionality entirely themselves.

Moving forward, the Flowdock team is interested in the cluster configuration functionality that's been delivered with the latest release of Akka. As of this writing, this functionality is not in use because Flowdock built out their own service architecture that takes care of detecting failures and restarting nodes.

In February 2013, Flowdock was acquired by Rally Software. The Flowdock team is Rally's first R&D outpost outside of the US and with additional resources available, the team is looking forward to continuing to innovate on the Flowdock platform.

*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.***

info@typesafe.com

Revision 1.0