Typesafe

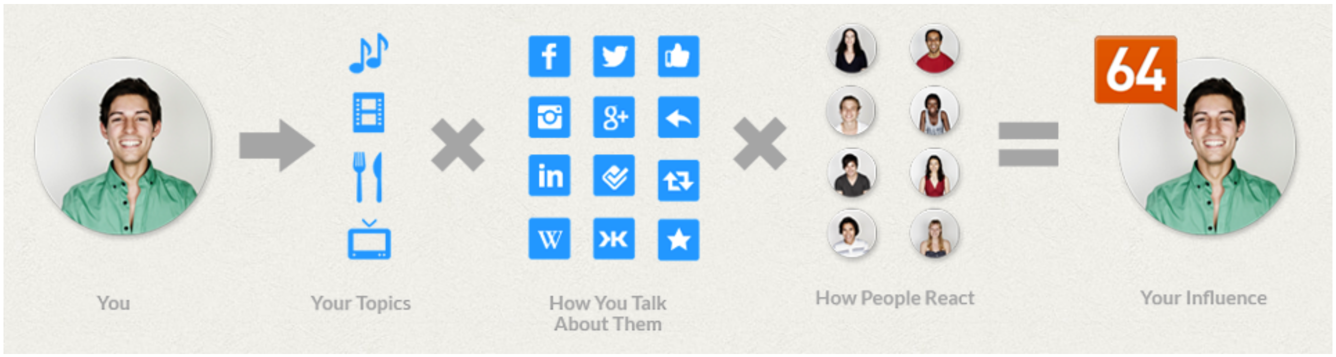
Bringing Reactive Applications
to the Java Virtual Machine

Typesafe gives Klout more Klout!

Before Klout could branch out into an important new market, they needed to ensure that they could overcome a rapidly growing technical hurdle first - data aggregation across several social networks and data stores in real time, with results instantaneously presented to end-users. Klout turned to the Typesafe Platform for the solution to its Big-Data problem.

About Klout

Klout's mission is to empower every person by unlocking his or her influence. Klout measures influence across several social networks and shows users how they impact the people connected to them. As social networks evolve, it's important for Klout to understand the new ways in which users are communicating with their audiences and update their analytical methods to integrate new features easily.



The Problem

At its inception, an elegant web-based interface was the only way end-users could interact with Klout. That all changed early in 2012 when the decision was made to expand the services reach into the mobile application realm. However, Klout's engineering team knew before they could attack this new marketplace their existing infrastructure needed some modernization. This infrastructure, which had served them well to this point consisted of a number of PHP servers that served up their website, along with Servlet, Spring and Java based applications that served up their legacy API. Since this architecture consisted of a set of one-off API servers, each with its own set of clients, a key goal was to unify the platform, as well as perform a technology upgrade to ensure a guaranteed level of performance as the platform scaled out. This goal also meant ensuring that the API layer was accessible not only from Klout's internal, web and mobile applications, but partner-based applications too - which broadened its potential user base substantially. It was clear that the infrastructure would have to be significantly re-architected to ensure that the scalability and consistency demands required of the infrastructure could be met.

Evaluating Solutions

Given the magnitude of the project at hand, combined with aggressive delivery schedules Klout did not have the luxury of time to conduct an extensive evaluation of available solutions. Thankfully, the engineering team brought with them a wealth of experience from prior work. Specifically the mobile project architect had experience using two technologies: Play Web Framework 1.0 for Java, and the Scala programming language. Play Framework is a very productive framework that is based on a lightweight, stateless, web friendly architecture that features predictable and minimal resource consumption for highly scalable applications. Scala is an elegant and concise programming language that integrates both functional and object oriented paradigms. Combining these two JVM based technologies would meet their requirements admirably.

Needing to look no further, with their platform chosen - all that was left to do was jump in and start building!

Crafting an Architecture

The Klout API is essentially a huge data mashup between a number of different data stores consisting of big data stored in HBase, along with user generated content in stored in MySQL.

Our legacy collection framework was written in Java and built on the java.util.concurrent library. As a consequence, data collection consisted of isolated nodes fetching user data sequentially with convoluted succeed-or-retry semantics in blocking threads. As our datasets grew with an ever-increasing user base, the inefficiencies of the system started to become apparent.

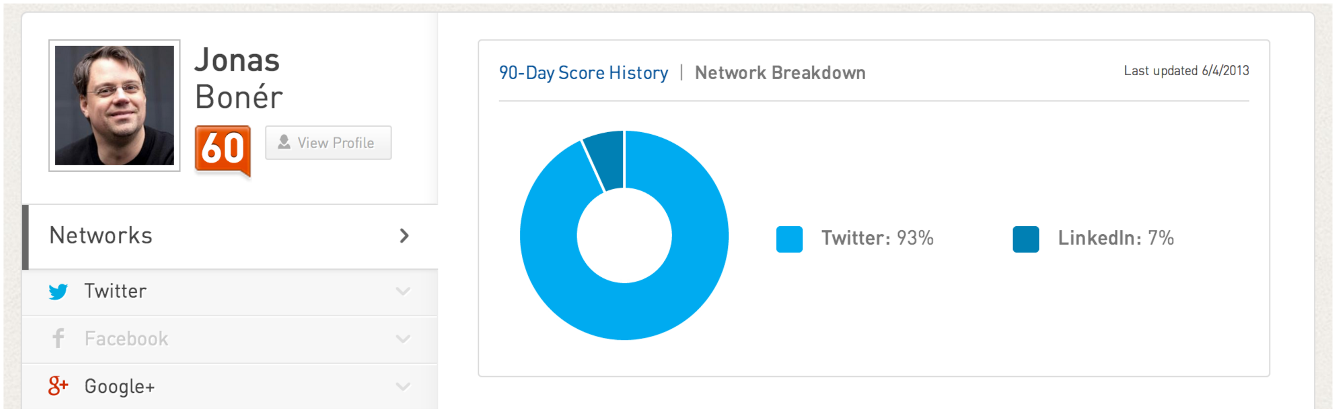
Naveen Gattu, Senior Software Engineer

Iteratees have become the cornerstones to Klout's new data collection architecture. Iteratees are the functional way to model producers and consumers of streams of data, and are easily parallelizable and highly asynchronous. Naveen furnished an excellent example of the kind of data collection done by the Klout service in a recent blog post. A user may post "I love Pop-Tarts® for breakfast!" to Facebook and receive 20 comments and 200 likes. This one activity consists of three components; the status message, 20 comments and 200 likes; these three components are independent of each other and can be collected asynchronously and in parallel. While a simple example, one can see value of performing this work in parallel when the volumes reach millions of users, with an ever-increasing number of posts across numerous social networks.

Having launched the API with Play 1.x, Klout quickly migrated to Play 2.0 as Typesafe announced its general availability shortly after the API launch. Play 2.0 offered the productivity of Play 1.0, along with the significant benefit of being built natively in Scala.

Today, the Klout service has four main codebases:

- The API written in Scala.
- The website written with Node.JS utilizing RESTful JSON served up via the API.
- A *Data Processing Pipeline* written in Hive and Hadoop.
- An iOS based mobile application written in Objective-C.



While the initial API project utilized Scala and Play, Klout has expanded their utilization of the Typesafe Platform to include Akka too. Akka is a toolkit and runtime for building highly concurrent, distributed, fault tolerant and event-driven applications on the JVM. Since Play was built on, and leverages Akka, many of its concepts were already familiar to the Klout team. Akka's Actors provide an elegant way to express business logic, without having to worry about the complexities of using Java concurrency. Typesafe's sbt plays a key role at Klout too, simplifying build management by tightly integrating the build process with the coding process.

sbt allows us to unify our codebase and avoids dealing with the Maven craziness; all of Klout's projects are multi-module builds, which also saves us from Maven hell!

David Ross, Senior Software Engineer

If Iteratees are the cornerstones of the Klout architecture, then Futures are its foundation. A Future is a data structure used to retrieve the result of some concurrent operation. This result can be accessed synchronously (blocking) or asynchronously (non-blocking). The Klout team simply loves Future composition in Play, as rendering a typical page can rely on hundreds of distinct data store lookups that need their results composed quickly and efficiently. This is one of the most important features that enables Klout to meet their massive scalability requirements.

At Klout, *everything* needs to happen asynchronously.

The Team

The initial team building out the infrastructure consisted of three developers building the API's, however that team has since expanded to about a dozen Scala developers. Like many companies, the team uses a mix of developer tools with some preferring Eclipse while others prefer Sublime. This team is currently managing a code-base that is 1½ years old that consists of approximately 100,000 lines of Scala code (including tests).

Obviously hiring talent is a concern for many companies, however at Klout, not many engineers have joined with Scala expertise. Scala knowledge and skills are obviously an added benefit, but Klout is happy to train smart, energetic engineers who are eager to learn.

Metrics and the Future

Klout's David Ross shared some interesting facts about their environment:

- Klout serves up an amazing *1 billion* API calls per day.
- Klout harvests about *1 terabyte* of data per day using Iteratees. Check out Naveen Gattu's fascinating blog post on how Klout uses Iteratees here.
- The Klout infrastructure currently consists of twenty or so stateless machines running the API service.

Always eager to ensure that they are doing things the right way, the Klout engineering team recently held a hackathon where the team decided to "do things the easy way" to solve a complex problem. It actually turned out that using Futures *was* the easy way.

The functional programming aspects of Play make it easy to get high performance out of the system.

David Ross, Senior Software Engineer

The Typesafe Platform has proven to be so successful at Klout that most projects built there now utilize it; Play is even used for non-web projects as it's so easy to get something going quickly.

*The **Typesafe 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 Platform through the **Typesafe Subscription.***