Typesafe

Bringing Reactive Applications to the Java Virtual Machine

Lucid Software uses Typesafe for next generation platform

When the Lucid Software team was searching for a new architecture foundation that would enable substantial user growth of their flagship product, Lucidchart, they knew that they needed a special toolset that was both productive and highly performant. Lucid Software found their solution in the Typesafe Platform.

About Lucid Software

Lucid Software is a software development company whose enterprise-level online diagramming tool, Lucidchart, provides a graphical web application in the browser that is on par with, if not better than, desktop software. Lucidchart enables companies to visually communicate processes and ideas in real-time across an entire organization. Lucidchart has a variety of shape libraries, which allow users to create diagrams like org charts, mind maps, UML schemas, ER diagrams, network diagrams, and more. Lucidchart's support of Visio file types allows for import and export in Visio format; it also gives users the option to import Visio stencils to create custom shape libraries for specialized use cases.

The Problem

Dealing with healthy growth in product usage is a problem that every company wants, but it can be challenging to provide an efficient and consistent experience for every user. The Lucid engineering team could see that their monolithic server architecture had too much overhead and some limitations that would make it difficult to scale in a cost-effective way. They realized that they needed to make a marked shift in technologies to ensure that the company could meet its performance, scalability, and reliability goals.

Lucidchart's growing pains were due to both architectural realities and to the limitations of certain technologies. For example, Lucidchart's monolithic application was not easily partitioned and distributed, and CakePHP introduced a lot of overhead per request, resulting in the minimum response times being too high under load. As it became clear that a significant portion of the code base needed to be rewritten, Brian Pugh, VP of Engineering, pushed for a complete survey of existing solutions to find the best tools to meet both current and future technical requirements.

The team determined that scaling up in a cost-effective way would be best achieved by adopting a Service Oriented Architecture (SOA) and making significant use of parallel processing to handle compute-intensive operations. Building such an architecture from the ground up is a significant task, so they wanted to leverage existing technologies and frameworks that were purposefully built for the task. The Lucid team narrowed in on PHP, Java, and the Typesafe Platform as possible technologies on which to base their future development.

The Solution

While the team enjoyed the productivity of PHP, they found the visibility into the runtime environment to be limited. On the other hand, the Java Virtual Machine (JVM) has many monitoring tools available with well-documented APIs (e.g. VisualVM, JMX, jmap, jstat). Another important factor the team considered was support for parallel processing, an area where PHP was lacking. While Java does provide the benefits of the JVM runtime and concurrency support, it can be difficult to utilize when massive scale and concurrency are required.

That left one option: The Typesafe Platform, consisting of Scala, Akka and Play Web Framework.

Scala is a general purpose programming language inspired by the long tradition of functional programming, which makes it easy to avoid shared state. Scala ensures the ready distribution of computation across cores on a multicore server, and across servers in a datacenter. This makes Scala an especially good match for modern multicore CPUs and distributed cloud-computing workloads that require concurrency and parallelism.

Play Web Framework is a framework based on a lightweight, stateless, web-friendly architecture that features predictable and minimal resource consumption (CPU, memory, threads) for highly scalable applications.

Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications on the JVM. Akka vastly simplifies the concurrency problem by providing developers with a higher level of abstraction in its Actor-based model. By utilizing Actors, developers can focus on writing business logic, instead of worrying about low-level tasks like state management, memory locking and thread management.

Since these technologies run within the JVM, the Lucid operations team had access to the monitoring tools they needed, while developers were able to take advantage of the numerous libraries available in the Java ecosystem. Furthermore, the engineering team continued to enjoy the highly productive development environment they had grown accustomed to with PHP, since Play supports hot deployment – no application servers to restart every time a change is made.

These benefits made The Typesafe Platform the logical choice on which to build out the next generation Lucidchart platform.

The Team

Lucid Software's developers are split into two teams. One team handles the client-side, and the other manages the server-side. The client team manages over 150,000 lines of JavaScript to provide Lucidchart's users with a uniquely rich and fluid SaaS experience. Those lines remained largely unaffected by the change in platform. The server-side team embraced the new toolset and invested the time needed to get up to speed. Scala and the Play Framework furnished better tools for re-architecting to the SOA model, provided a variety of options for parallel processing, and pushed developers to write high quality code, an example of which is below:

```
// page is the model for the Lucidchart representation of a page of a document (diagram)
// pdfPage builds PDF rendering of page as methods are called on it

// if any exception is thrown during PDF page generation, notification sent with stack trace
withNotify("Generate PDF Page") {
    // initialize PDF page
    pdfPage.mirrorY()
    pdfPage.scale(page.pdfScale, page.pdfScale)
    pdfPage.setBackgroundColorFill(fillColor)

    //render all blocks on page
    pdfPage applyTransform {
        pdfPage.translate(-viewport.x, -viewport.y)

        // Render text items
        page.body.foreach (textItem => renderText(pdfPage, textItem) )

        //render other block items
        page.items.foreach { item =>
            pdfPage applyTransform {
                pdfPage.rotate(item.rotation, item.boundingBox.center.x, item.boundingBox.center.y)

                pdfPage.alpha(item.opacity / 100.0)
                cropItem(pdfPage, item) {
                    item.renderData.foreach { data =>
                        renderData(pdfPage, data)
                        ...
                    }
                }
                ...
            }
        }
        ...
    }
    //render master items (items are on "master" page which can be applied to any other page)
    val masterItems = page.master match {
        case Some(master) => master.items.map ( _.forPage(page))
        case None => List()
    }
    ...
}
```

This code is a snippet from Lucidchart's PDF generation library. While the library on the whole is fairly complex, it takes advantage of Scala-specific features, creating clear and concise code by making use of closures (e.g. cropItem, applyTransform), infix notation (e.g. applyTransform), high order functions (e.g. map, foreach), pattern matching, options, and so on.

Scala has allowed Lucidchart to essentially create a DSL for PDF generation, since it lends itself so well to a translation of Lucidchart's hierarchical data model for representing a diagram. The result is a proprietary format transformed into accurately rendered images.

The Results

When migrating a large application, a key tactic is to introduce small changes early and often. Today, Lucidchart still has PHP code on the server side, but is constantly migrating services to make light work of an otherwise gargantuan task.

A poster child for this strategy is the image generation service, which creates a PDF, PNG or JPEG image of a Lucidchart diagram; it is used *extensively* in the application and seemed like an ideal target on which to practice. Rewriting this service in Scala, leveraging Akka and Play, has resulted in marked performance and stability gains. The improvements were primarily the result of moving from the single threaded, sequential approach that Lucidchart had in PHP, to a parallel processing approach using Akka's Futures and Actors. Now, multiple threads are used to handle a single image or PDF generation.

The Typesafe Platform has also helped deliver measurable value with ROI increases in reliability and performance. The results of moving the image and PDF generation from PHP to Scala, Akka and Play are:

- Average response time decreased by 53%
- Median response time decreased by 37%
- Overall failure rate decreased by 85%

Conclusion

Lucid Software was searching for a solution to key development concerns of scalability, reliability, and performance as they migrated from a monolithic architecture to a services approach. The Lucidchart development team's audit of available technologies found the Typesafe Platform to be the best available option, given its compatibility with the JVM, enhanced support for concurrency, and language features that encourage high quality code. This implementation led to significant improvements in performance of services that have been migrated to the Typesafe Platform, and the Lucid team plans to continue the migration to completion.

*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.***