FPComplete

Bump Use Case Study

Perhaps you've seen the commercials, where two people bump their phones together and transfer a photo. It's pretty cool, right? The solution behind that is called Bump, the brainchild of rising star Mountain View CA based Bump Technologies.

Like most startups, Bump faced numerous challenges bringing their exciting solution to market, not the least of which was to reduce the errors they faced in other programming languages, while improving the quality and productivity of their development process.

Unlike other languages Haskell ensures that companies can enable their development and production teams to maximize their productivity, performance, quality, scalability and agility in ways that no other programming language can.

Only Haskell's model of functional programming provides all of these benefits and a growing number of companies, like Bump, are realizing these benefits for their applications. These benefits result in substantial savings over product and service lifecycles, allowing developers to concentrate more on innovation and less on operational issues.

What is Bump?

Bump was originally designed to help people share business cards more easily and ensure contact information wasn't lost. Over time, the solution grew into a means to readily share contacts, photos, files more quickly than through existing channels like file sharing services or email.

Bump Technologies, founded in 2009, launched their product in 2010, virtually skyrocketing in popularity overnight – in no small measure after it became the billionth app to be downloaded through Apple's iTunes service and Apple began promoting the tool to a wider audience. It has since become the tenth most popular free app in iTunes around the globe.

Behind the gentle bump of two phones is a blend of technologies utilizing colocation to facilitate the connections. The colocation functions combine GPS, to tell the Bump servers where each phone is, timing of the bumps, along with latency compensation to adjust for different cellular network carriers' speed. Bump also adjusts for elements like the light, sound and background environmental factors where the phones might be located, as well as accelerometers for how hard or soft the bumps are.

Python Constrictions

In the early days of the solution's development, Bump wrote their code in Python, a programming language that most of the founding and development teams were familiar with, which allowed them to get rolling quickly.

As their development continued to grow and they needed to scale their application, their team started having significant challenges, including getting a rash of serious single points of failure, particularly in shared memory functionality of the application.

The tipping point came for the Bump team when, in addition to high error rates, the volume of Python coding relative to the size of the team eclipsed their ability remember all of the code in their heads and share it accurately.

Thankfully some members of the team had experience in Haskell's method of functional programming and enticed their team to begin developing code in it, in part to avoid needing to hire a large team of Java programmers. They knew it would be an excellent means to drive down the error rate and eliminate the volume of code, through the inherent clean and terse nature of Haskell.

"We had some initial challenges with our non-functional programmers learning Haskell and increasing their productivity, but ultimately these growing pains have been outweighed by considerable long-term time savings that came from eliminating all of the errors we had before, while being able to readily duplicate code with Haskell's simple structure," noted Bump's lead developer, Jamie Turner. "II definitely is an good alternative to traditional development tools"

The longer that Bump Technologies used Haskell, the more they found that it was fun to program in, works well, runs long and flat in memory, with few problems out of the gate. Their team fully expects to use the language even more as their company continues to grow and their needs for low error rates and high scalability are magnified.

Haskell's capabilities enabled Bump Technologies to:

- Quickly develop prototypes with few errors.
- Easily extend code and scale their application as customer demand grew.
- Simply and reliably deploy changes or new features without introducing errors.
- Dramatically reduce time spent on testing and debugging.

"Haskell has proven to be extremely multicore and multiprocessor-friendly, supporting the scaling needed with our rapid growth. We knew we could code extensively without problems," Mr. Turner further explained. And by using the STM functions of Haskell, they also avoided challenges from locks or mutexes.

Breaking the Constriction

Bump Technologies found that Haskell provided such a robust and reliable base, which was incredibly scalable, that they could easily grow their application on their server environment from one based on six to eight cores, to one that is more than 250 cores, stretched out over 70 to 100 machines.

"The Haskell compiler is very reliable and refactoring in the language is, well, glorious," Jamie Turner gushed. Bump found that the compiler function gave them excellent confidence in their software redevelopment as they grew their solution, especially by providing multi-year life of their code other solutions like Python couldn't offer, while scaling up to support millions of users.

Coil-Free Code

Haskell's reliable, extensive type system and checking functionality ensured that Bump eliminated the prevalence of coding errors and limits to scaling they were encountering in Python, thus producing the highest quality code – while cutting down the development time when taking their prototype into production.

The modularity and libraries in Haskell facilitate extensive code reuse and parallel development, further enhancing the productivity programming teams can achieve, while greatly increasing reliability and reducing errors.

Plus, Haskell's native functionality, by eliminating shared values and interdependencies, increases scalability and performance, ensuring functions can be run in parallel on multiple processors and cores.

Ultimately Bump Technologies found that:

- Significant time savings in the overall development cycle, led to greater overall productivity.
- Haskell's compiler caught the few development errors that occurred, greatly increasing their commercial application's performance.
- Robust code ensures long-term application reliability and future changes can be made without compromising performance for their customers.
- Reduced development costs and easy application innovation drives high ROI.