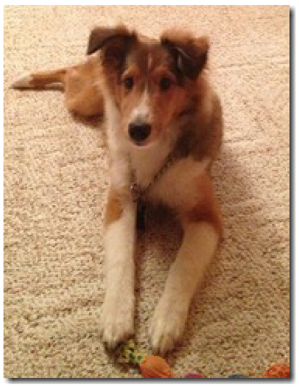
Re-Distributed Computing

Don MacVittie, 2012-09-11



This morning, Chaucer, our relatively new Sheltie puppy, took off with my slippers. I found one lying in his favorite spot almost immediately, but the other had gone missing. I wandered about the house with one slipper in hand while he hid in his kennel, knowing I was angry but not quite understanding why. It made me ponder the state of computing today, and where we're headed, because it made me ponder years ago with our last dog, doing exactly the same thing. One was a Shih Tzu, the other is a Sheltie, but puppies chew on slippers with rawhide ties, doesn't matter the breed or how many years in between.

Interesting thing about the state of the computing market, puppies – all puppies – do chew on slippers, and that rarely changes, even as we see change all around us in the high-tech world. The corollary? I sat down to learn Android development at a level most people aren't interested in learning months ago, and guess what? After a decade of progress and hype about new and different, I wrote in Java and C++, used some of the same libraries I used when writing for the Blackberry several years ago, and made calls to most of the familiar parts of Java and quite a few unfamiliar that haven't significantly changed. The same is largely true of Objective-C. I used it a very long time ago, and while there have been changes, it's still Objective-C.

The same is largely true of .NET development. There have been changes over the years, but for the last decade or so, they've changed .NET less than they've changed the command line (if you count PowerShell). Alternative languages like Ruby have come along, and those of us who geek out on this stuff have tried them, but to-the-hardware development is still done largely in C/C++, and high-level software development is still done largely in Java and .NET with a healthy dose of PHP. The databases organizations ask for have been stable for just about as long. No, I'm not ignoring the Hadoop and derivatives craze, regular app developers are rarely asked for that skill set as a primary skill – so far.

So what have we developers been doing over the last decade? Learning new platforms, of course. Learning new ways to integrate, of course, but largely, not stretching enough. There are day-to-day problems we're dealing with, and every dev has to learn new things just to do their job, but they're largely mundane or vertical things.

To some extent, the fragmentation of dev combined with the growth of software as the engine of business and a projected reduction in development jobs that never seems to have occurred all contribute to this scenario. And of course, the difficulty in displacing languages with millions or billions of lines of code impact it.

So does a slowdown in what is revolutionary, I think. after SOA, what next? Well, there have been a ***lot*** of developments, from Git to AWS, but in the end, development is much the same. PaaS changed that very little, though it was the most likely of the cloud technologies to do so. We also filter a lot, I think. Do you write multi-core code? Seriously? Most devs in the enterprise don't, even though there are documented benefits. That's "beneath the hood" so-to-speak.

So what has me excited? There are signs of significant change. SDN promises not just a change in platform, but hooks you can manipulate to make your code more reliable. It's not a Brave New World or anything, but it is food for improving App Dev without a seismic shift in development environments or methodology. The future of mobile device development promises more changes too, as more and more devices suck up more and more bandwidth. An interesting analyst note was that 4G wasn't helping because devices were being added to the network faster than the network speed was improving. That means optimized communications either in the app or in a device between the app and users, is going to continue to be important.

Things like Git fall firmly into the re-distributed computing category. It's a server. Always. You may not think of it as one, but if I have permissions, I can ask to clone the code in your repository. That's a server. With the added resiliency that so is every other dev machine. Which means the loss of a "server" is not likely to be catastrophic as long as best practices for SVN have been followed. Someone will have a new enough copy that only the most recent work will be lost. And of course, if you're replicating to GitHub, the likelihood of a catastrophic loss is even smaller. Not a huge change (it's still version control), but a change that stabilizes source issues in the loss of a single machine.

This whole micro-server concept will creep in elsewhere, I think. Making a single machine or VM the repository of knowledge critical to your business has never been a great idea, and our outrageous backup plans show we know that at some level. If we're replicating by virtue of daily business, it will increase network traffic in exchange for less single points of failure. Which means it will happen, because network bandwidth always increases, single points of failure are always a weak point.

As to languages, the death of Java has been predicted a lot, but I just don't see it happening any time soon. Java and .NET are both general purpose environments suited to the current state of enterprise computing. Mobile may produce something that truly changes things, but at this point it doesn't look like it. Low-level systems and those that need greater performance will likely continue with C/C++, though the number of environments that entails will continue to decrease. RoR and PHP will continue to eat a lot of web UI space because they're particularly suited to it, and we'll continue on our way. Command line scripting languages are almost never impacted by even seismic change, so nothing to see there.

In short, the places to innovate are the connections we can make to other parts of the network – applications and increasingly infrastructure. REST and SOA will continue to rule that space.

So we're re-redistributing. Don't get complacent. In high tech, when you cannot see the massive changes, that means they're happening incrementally. Continue to grow, stretch your boundaries. Try new things. Bring more to the table, because it is the external environment that is currently changing, and that needs to be accounted for in application development.

And if you haven't tried out Git, go check it out (pun intended), you'll find it intuitive and well documented.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com

©2016 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113