From: www.cio.com

The Trouble With Coding Across the Clouds: Part 1

– David Taber, CIO

May 02, 2011

Most cloud applications have development (or at least scripting) capabilities that allow for deep customization plus some level of database access and computational capabilities. But even the best of the cloud applications must put in limiters for their platform/development environments: an app isn't a general purpose run-time or generic object container. For example, the development language must be made safe for a multi-tenant deployment, and must be well-behaved so that user code can't take down the virtual machine, database, or overall application. Further, some kinds of language constructs must be limited to prevent resource hogging and deadlocks. (Indeed, if you think about the billion lines of user code that Salesforce.com has running in their cloud, keeping quick responsiveness and good uptime stats is a non-trivial task.)

Using Salesforce's APEX as an example, the language can handle most business logic without too many tricky work-arounds. But inevitably with a cloud-based development environment, there will be certain requirements that push against the platform's limits. For example, there might be a terrific library in J2EE that does just what you want, but J2EE is simply not available in the cloud platform where you need it. Even if you need just a couple of methods out of that library, it may turn out that some of its underlying functions can't be successfully ported over to the native language of your platform.

Let me take a real world example: license key generation. Software vendors like to use license keys to enforce their end-user contracts. The CRM system holds the license keys (as part of the customers' Assets), and it would be nice to generate the keys entirely within the CRM application. So the software organization asks to port the keygen system into the CRM app. The key generation uses encryption methods that are available as part of the CRM platform, but it also uses a nifty library that does arbitrary precision math. Arbitrary precision, as in: multiply two 400-digit numbers and take the square root modulo 7. That nifty library does this math entirely with strings, using recursions that are most lovely.

So even though you can port all the logic to the CRM system, the computational load of actually running the keygen will blow through the governor limits that must be enforced on the CPU, heap size, and number of queries. These run-time limits frequently hit analytics (think BI), optimization (think stock portfolios), really big lookups (think the Named Account Model or multi-channel distribution strategies)

and other business applications touched by the CRM.

The answer, of course, is to call a service that does the data crunching outside, in an adjacent cloud (hence the title for this article). Unfortunately, there aren't any easy design patterns for these situations, because:

• Some of the data you need to access might not be able to move, because of policy, organizational politics, security, or other reasons. In other situations, the data might need to be moved because of other system changes that are underway.

• If your other cloud has to crunch on a lot of data that must live in the CRM database, you might want to ship abstracts, summaries, rollups, or bit-map representations of the data, rather than raw records.

• Depending on the problem, it may be easier to transfer data from a remote resource, or to port code to another cloud and use a function- shipping strategy.

• The other cloud may be easier to work with in RESTful protocols like JSON, even though the problem lends itself to WSDL and SOAP.

• Depending on the nature of the computation, it may make sense to do virtually all of the work inside the CRM system, and then call just the tiny method needed from the remote cloud. Conversely, it may make sense to do virtually all the work remotely, and call it as a full-fledged service.

• If the computation needs to evaluate system state (e.g., workflows, locks, or data changes), the amount of network traffic (and the resulting latency) can be a serious factor.

• Security, testing, and deployment considerations can't be ignored. If the computational problem is important enough, it's going to be around for years. So it has to be managed over time and survive with minimum hassle even when there are developer, administrator, and organizational ownership changes. (Think about whether that upcoming reorg will mean your developers will even have sufficient access to that other cloud in the future.)

So the first step is to determine the best architecture for your particular application, figuring out what elements of data should be transferred and refactoring your classes across the cloud geography.

Next week, we'll look further into coding across the clouds.

David Taber is the author of the new Prentice Hall book, "Salesforce.com Secrets of Success" and is the CEO of SalesLogistix, a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel, and India, and David has over 25 years experience in high tech, including 10 years at the VP level or above.

Follow everything from CIO.com on Twitter @CIOonline.