From: www.cio.com

The Trouble with Coding Across the Clouds, Part 2

– David Taber, CIO

May 10, 2011

The best cloud applications have development (or at least scripting) capabilities for creating and extending the platform's database 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.

The Trouble With Coding Across the Clouds: Part 1

Even if the cloud platform's language weren't limited, though, there are plenty of situations where you just want to leverage an existing software asset or database that's in another cloud. Conversely, some other cloud might need to grab some data or manipulate objects inside your cloud-based CRM app. Of course you can have the clouds interact via web service or RESTful calls, but in many situations that isn't tightly coupled enough for the functional requirement.

What you'd *really* like to do is have your developers just start typing their favorite language in their current cloud, and get access to the objects and services of the other. Cool idea...but not so fast.

If we could just use that J2EE library...

There are three situations in Cloud application development and extension where you'd love to be able to leverage a software asset in a different language, running in a different cloud:

• You have a legacy asset that is known to work
(or can be very easily extended to perform a function) in a cloud A, and there's no reason to replicate it in cloud B.

• There are data and facilities available in a cloud A that simply can't be recreated in cloud B's environment (for example, a real-time trading algorithm or an ecommerce recommendation engine).

• You have developers who are very proficient in another language (for example, Java, C#, Ruby, Python, or PHP) native to cloud A, but don't want to learn the native languages of cloud B.

The lowest common denominator for this situation is to do a Web service or REST call (or even the sub-basement XML document exchange) to leverage the software asset outside of your application's cloud. While this can always be made to work, the latency and overhead can be annoying to users. Further, your developers may find retrofitting Web-service interfaces in their legacy code to be annoying. Just try it in COBOL.

A better answer is to use a set of libraries and language bindings that make calling another cloud's methods and objects pretty straightforward. These offerings, such as the Salesforce-VMware bridge, can really save developers a lot of time...with a few caveats:

• Java is not just a language: it's an enormous set of services and object libraries. VMforce leverages Java's Spring environment so that SFDC developers can get access to servlets, jsps, and other business-logic functions. But it doesn't really help for J2EE, J2ME, or the more advanced services and abstractions that the Java community developed over the last 15 years. If you're trying to leverage the heavy lifting server side components or the Java UI world, you're back at square one.

• Your developers are almost certain to have to write some wrapper code so that their classes are presented as a Web Service.

• The Java security model will map to some clouds' generic data and object access infrastructure, but most cloud applications use a security model that's focused more on the context of application usage. As I wrote in February, mapping security models can be quite daunting (particularly if you think of maintaining complex mappings over time).

• As with any cross-cloud software, your application has to have a recovery strategy for when the network times out or the other cloud simply isn't operational. The leading cloud application vendors have unplanned MTBF of 7500 hours or more, but your other cloud(s) services may not be as highly available.

My Java developers need to manipulate some Salesforce data...
In this use case, the developers in another cloud are trying to get access to some data and functions in your cloud application. Again, the lowest common denominator is almost always available via ODBC or your favorite integration server. This approach is pretty clumsy, however, as it depends on a lot of manual effort at development and configuration time.

A better approach is to have dynamic wrappers that understand changes to the data model in your cloud application, and present the relational objects as a set of classes at the language level in the other cloud. This is much easier for the developers to understand and work with, again with a few caveats:

• To what degree will one cloud's object and database customizations be automatically / dynamically handled at the "other end" of the object bridge? The bridge is unlikely to be perfect, but you don't want to have to restart the services every time somebody changes a field definition in the other cloud.

• What limitations of language or object model will come into play? Typically, when coding across clouds you have to live with the lowest common denominator. For example, if your Java code is depending on full SQL92 or proprietary extensions, your developers are going to have some work to do when accessing data from another cloud's application.

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.