Muse Review: Exploring CouchDB With Matt Woodward

Posted At : May 22, 2012 10:56 AM | Posted By : Mark Kruger
Related Categories: cfobjective

On Saturday I sat in on ColdFusion genius **Matt Woodward's** session on practical couchDB. I have experience with both Memcached and MongoDB so I thought I was prepared for the general sense of what you could do with CouchDB (which I had never explored). I assumed it was just another "no SQL" database. But Matt demonstrated some things that were new to me and I am intrigued enough to experiment with them - hopefully engendering a few more "CouchDB" blog posts. Here's a couple pros and cons gleaned from the presentation.

The Pros

For one thing it seems like (from the demo) that CouchDB's simple HTTP interface for getting, putting and updating records is a natural fit for ColdFusion and CF programmers. The storage engine uses Json (and really... who *isn't* using Json) which makes it easy to work with fairly complex data types. For example, getting content looks like this:

```
<!---MAK: get some user--->
<cfhttp url="#somedomain#:5984/dbname/keyOrGuid" meothod="GET" result="getUser"/>
<!---MAK: turn him into a CF data object --->
<cfset getUser = deserializeJSON(getUser.filecontent)>
```

Other actions use the standard HTTP verbs (PUT, DELETE, POST etc). HTTP status codes allow for traditional error handling. This seems conventional and easy to grasp. If a "document" (an individual record) is not found you get a 404 error. Various views and filters are possible by altering the URL or passing params in as a part of the request. It's a straightforward model that "fits" what a typical ColdFusion programmer already knows how to do - unlike MongoDB or Memcache which (while still easy) require some java construction.

The most intriguing thoughts bouncing around in the Muse' head have to do with performance. In its simplest form CouchDB is a listening HTTP port. Since it has clustering and replication under the hood (this was not demonstrated but it was made to sound easy :) you could easily cluster your content on multiple VMs or iron behind a load balancer - making scaling with traditional load balancing a piece of cake and eliminating the need for the application to use a teamed IP, know about failover, handle primary/delegate relationships etc. Indeed scaling databases is a great deal more challenging than scaling web servers, so having this capability should be a real plus.

The Cons

Not that I'm completely sold. I still worry about the overhead associated with HTTP requests. A cfhttp request is more expensive than a JDBC driver call and there's not "connection pooling". So the general connect and retrieve infrastructure is going to be more expensive than traditional DBs - at least that's my theory. In addition, you have to pull in the data as a string and deserialize it. Granted Json serialization/deserialization is light weight but it's still one more step.

What would really be cool (listen up Railo and Adobe Folks) is to allow a CF admin to

register CouchDB (or Mongo or whatever) as data-sources or pseudo datasources and provide a function or tag to get the data back already deserialized. That would be a true data enhancement that fits with the ease of the CF administrator. One of ColdFusion's strengths is this sort of "pre-configured" resources that are easy to access within the code. No SQL Dbs would be a natural extension of this I think. But back to performance...

I'm often given a sort of glib response to granular performance concerns (like the overhead associated with CFHTTP) by developers who say "how often do you really need to worry about millisecond level performance? The truth is - quite often. One of my CF Webtools roles these days is to dive into under-performing systems and try to find ways to maximize output and speed. In many cases the goal is to get another month or two out of existing hardware or design while a new approach can be devised. In such cases *everything* is in play. Consequently, when designing a *new* system that is intended to receive a lot of request traffic, it's important to make such decisions right up front or pay the penalty later.

Still, for intermediate caching, shopping carts, sessions, aggregate portals and many types of content CouchDB would be a huge step up from the hodge-podge of approaches usually seen. I was impressed with its ease of use and responsiveness.