BlueDragon 7 - Coldfusion's Little Sister is Growing Up

Posted At : February 28, 2007 12:51 PM | Posted By : Mark Kruger
Related Categories: Coldfusion's Future, Product Reviews

You probably already know about BlueDragon. It's a fine ap server from the folks over at **New Atlanta** (the makers of Jturbo and ServletExec). BlueDragon is an interpretive engine for CFML that is a direct competitor to Coldfusion. Last night I was privileged to see a presentations by Josh Adams from New Atlanta at the **Nebraska CFUG** on the upcoming release of Blue Dragon 7. I have always thought that one of the knocks on BlueDragon is that BD is forced to play "catch up". The folks at New Atlanta have to wait and see what features appear in Coldfusion and then hurry to implement them in the next version of BD. I was pleasantly pleased and surprised to see that the next version of BlueDragon may force Adobe to play a bit of catch up on their own. It has a number of quite innovative features and I was duly impressed. Here are some of the more impressive things I saw or heard about.

Thread Management

A request can spawn worker threads for asynchronous work. The request can "fire and forget" these threads or "rejoin" them (wait for each of them to finish). In the example that Josh used a call to a web service was done 4 times serially at a cost of around 1000 milliseconds (1 second). He then modified the code to create 3 threads and made each call to the webservice asynchronous to the request thread. It looked something like this:

```
<cfthread name="a">
        <cfreturn wsCall.getData("somevar")/>
    </cfthread>
    <cfthread name="b">
        <cfreturn wsCall.getData("somevar")/>
    </cfthread>
    <cfthread name="c">
        <cfreturn wsCall.getData("somevar")/>
    </cfthread>
    <!--- let the request thread do some
            work as well
     --->
    <cfset d = wscall.getData("somevar")/>
     <cfscript>
         /* joinThread() makes the request wait
        for the specified thread to finish.
        Cfjoin is the tag aquivelent
        */
         joinThread(a);
        joinThread(b);
        joinThread(c);
     </cfscript>

    <cfoutput>
        #a.returnvariable.rs.column#
        #b.returnvariable.rs.column#
        #c.returnvariable.rs.column#
    </cfoutput>
```

In the example, I'm imagining that the wsCall returns a query. A special "returnvariable" provides an entry point into data that is created by the Cfreturn call.

It's complicated but the results were impressive. The web service process went from more than 1000 milliseconds to around 300 to 400 milliseconds.

Of course you can also use cfthread to fire off worker threads and allow the request to proceed. One possible use of it that might not be evident at first is to avoid double submits. Sometimes the reason that double submits occurs is that the user becomes impatient with the length of time the request is taking. Using CFTHREAD you could offload the time consuming data processes to a separate thread and take the use right to feedback - much like a gateway or message queue. I have an upload tool that does a data import from an excel file that could greatly benefit from such an approach.

Cfquery "Background" Attribute

This is really just a specialized use of threads for handling database interactions. If you set *Background="true"* in your cfquery tag the server will spawn a separate worker thread for that database call and allow the rest of the request to continue. The example that Josh gave us was a logging routine that ran in the onRequestEnd() function in application.cfc. The logging function inserted some Cgi params into the database as a tracking tool for users of the site. Such routines are pretty common (Farcry has such a routine enabled by default). Setting background to true allowed this query to run "in the background" and queue up with other such queries without impacting the request - very cool.

Cfquery "CacheUntilchanged" (.NET and MSSSQL)

A nice feature of the .NET version of BD allows you to add the "cacheUntilChanged" attribute to a query (not sure if I spelled that correctly). This attribute works exclusively with MSSQL. If data changes on the DB Server the cache handler is notified that the cache needs to be expired. The next request for that data results in a call to the DB. Using this method you could ensure that queries are run only when they are necessary. That is *very* cool.

onMissingTemplate() Function in Application.cfc

Josh briefly showed us this tag in the application.cfc template. it allows you to take specific actions if the page requested is not found.

Conclusions

Overall I'd say that BD deserves another look. I was most intrigued that Josh indicated a new pricing structure may be coming for BD.NET more along the lines of the "standard" version. I understand they are trying to cater to the enterprise market but in my opinion the CPU pricing for BD.NET is too high of an entry point - especially with CF enterprise in the same space. According to Josh they will be offering a version of BD.NET that is more akin to their JX pricing. I think that is a move in the right direction. It is nice to see a product that competes with Coldfusion on it's own turf and seems poised for new success. Congradulations to New Atlanta for an outstanding effort.