UUID Magic With Java - When Speed is Critical

Posted At : January 13, 2011 2:32 PM | Posted By : Mark Kruger
Related Categories: ColdFusion

Note, this post is compiled from information foraged and provided generously by the inestimable **Brian Meloche** who's ColdFusion skills are (quite obviously) legendary. The Muse gives praise where praise is due - and with a nice dose of hyperbole to boot.

Many folks use UUID's for various reason. CF has a nice function built in that handles UUID creation - createUUID(). Try it out - use <cfoutput>#createUUID()#</cfoutput>. You should see a funky 36 character string that looks like this - *7C286425-CA3D-1B10-16A3CCD259C21FEE*. If you are new to ColdFusion or programming in general you might not realize what's special about the UUID. It is guaranteed to be *unique* - at least within reason. There is a statistical probability that a duplicate is possible, but your chance of finding one is about the same as being kissed by Ann Paquin or being intellectually stimulated by the show "Jersey Shore". If you want to know more about the uniqueness of UUID's check out this interesting (or mind numbing - depending on your perspective) **article** on Wikipedia.

Just how would you use a UUID? There are a myriad of ways. You could store it as a cookie to identify a "unique" visitor. You could use it to tie into your custom "roll your own" session management. You could use it as part of a seed for encryption. But probably the way most folks use it is as a primary key to the DB. Now I know that most RDBMS systems include a built in UUID function. If you are planning on programming for one DB (and there are often *very* good reasons to do so) then I recommend using the built in function. It's typically faster to have your DB Server create something like a UUID than passing a 36 character string to it via the JDBC driver. However, if you wish your code to be portable then it's likely you will be creating your own UUIDs using ColdFusion's built in functionality.

Slight Problem

If you are using a version of CF prior to version 9 then ColdFusion has a particular way of generate UUIDs that is tied to the clock and MAC address. It is capable of generating about 100 per second. There are times when that number might become a bottleneck on a high traffic site. For example, if you are at the peak of your traffic with 700 or 800 concurrent connections and suddenly an aggressive bot starts crawling your site. ColdFusion may not be able to quite keep up with that number of generated UUIDs (assuming your Bots are generating UUIDs through you code somehow – logging, sessions or whatever).

Fortunately there is a "JAVA" way to do UUIDs that is able to get around this issue. Actually (quoting Brian) there are 5 "classes" of UUID in the java.util package. CF Apparently uses a slower one of the 5 (presumably more compatible with disparate environments or whatever). The following Java code uses a different one of those classes. The code is pretty easy to figure out. In most cases you could simply drop this code in to replace your createUUID() code:

```
<cfset uuid = createobject("java", "java.util.UUID") />
<cfset newUUID = uCase(removeChars(uuid.randomUUID().toString(), 24, 1)) />
```

Another Brian, Brian Ghidinelli, published this post where he tests the speed of createUUID() against the Java code above and finds a nearly thousand times increase in speed using the Java UUID code.

Meanwhile Brian Meloche wanted to verify that ColdFusion 9 improves the performance of the original CreateUUID() so he ran the following test on both platforms:

```
Old:

<cfset timeIs = getTickCount()>
<cfloop from="1" to="1000" index="i">
    <cfset uuid = createUUID() />
</cfloop>
<cfset timeIs = getTickCount() - timeIs>

New:

<cfset timeIs = getTickCount()>
<cfloop from="1" to="1000" index="i">
    <cfset uuid = createobject("java", "java.util.UUID") />
    <cfset new = uCase(removeChars(uuid.randomUUID().toString(), 24, 1)) />
</cfloop>
<cfset timeIs = getTickCount() - timeIs>
```

What did he find? While ColdFusion 8 benefited tremendously from the Java version with a 200 to 1000 times improvement, ColdFusion 9 saw only a modest 10 times improvement. From this test I think we could reasonably conclude that using ColdFusion 9 you are not likely to run into any problems with UUID bottlenecks regardless of whether you use createUUID() or the Java version of the code.