Query Caching Run Amuck: Know Your FIFO Buffer

Posted At : September 20, 2010 11:08 AM | Posted By : Mark Kruger
Related Categories: Coldfusion Optimization, Coldfusion & Databases

Query caching is one of those underutilized features of ColdFusion that can exponentially speed up your application. It is also one of those misunderstood features that, when used *incorrectly* can be very disappointing. Let me say at the start that the Muse believes you *should use query caching*. If you don't believe I'm a fan then check out my post titled, **Good Developers Practice Safe Query Caching**. It's not a panacea, but it definitely has it's uses. Almost every application has some queries that can be cached - and saving round trips to the database is the holy grail of application tuning. But in this post we want to talk about naughty developers who cache irresponsibly... developers who do not understand the nature of the FIFO buffer.

FIFO Buffer?

No, it's not the mouse from "American Tail" (We have a Pwan!). It stands for "First In First Out" and it is one of the controlling features of query caching. Remember that query caching is controlled by two things. First, each query has a "time to live" sort of value controlled by either the cachedwithin or cacheafter attributes of the cfquery tag. Second, each query is subject to be removed from the cache based on the FIFO buffer and the limit set in the ColdFusion Administrator. So, for example, if you set the number of queries to cache to 500 and your application tries to cache query number 501, the buffer looks for the "oldest" query (the one that was "first added to the cache" - "first in") and removes it. The next time that first query is run it will be added to the front edge of the rolling buffer and the oldest query will be removed - and so on.

Now this fairly sensible framework where a limit is set to the total number of queries able to be cached and the oldest query is the first to go (also known as a "Logan's Run framework") is put in place to conserve the balance of resources on the web server. If an application were allowed to cache thousands and thousands of queries then pretty soon retrieving information from the cache would in itself become a bottleneck. In fact, the CF Administrator won't allow you to set the cache limit higher than 4 digits - making the effective FIFO buffer maximum 9,999 queries.

Make Caching Unusable

The problem rears its ugly head when you don't understand the nature of this buffer and you cache too aggressively. For example, let's say you have a search engine running against the library of congress. One of the most popular features is to search by author. You decide some caching might help you, so your search query looks like this.

```
<cfquery name="getBooks" datasource="#dsn#"
        cachedwithin="#createtimespan(0,1,0,0)#">
    SELECT *
    FROM     locBooks
    WHERE    author = '#form.author#'
</cfquery>
```

Now this seems quite sensible doesn't it? If I search for Hemingway (presumably a popular author) my query is cached and the next fellow looking for Hemingway will pull his information from the cache. I would guess that the locBooks table doesn't change that much so I'm pretty sure I'm in ok caching for a full hour. I could probably

even increase it to several hours - maybe even a day.

Does anyone see the flaw in my plan? If my little application becomes popular and dozens of searches are run each minute my buffer is going to fill up quickly. Each unique search takes up an entry in the cache no matter how obscure or trivial. Did Lindsay Lohan write her memoirs yet? Wasn't there a mathematician named Grenalda Smortsgrammer? How do you spell Kierkegaard? Many authors are going to be searched for *just one time* and you certainly don't want to waste space in the cache with them. Why? Because if the FIFO buffer fills up the queries you really want to keep in the cache will begin to get pushed out.

Not Just Theory

I'm not just blowing smoke here. I have a server with a large heap that contains an application that depends on caching very large queries of product data from Amazon. It began to underperform and upon examination I found caching being used for a search query like the one above - where *the parameters passed to the query were largely unique*. The FIFO buffer was set to the maximum of 9999, but it was filling up and my product data queries were being forced out by trivial search queries. Putting things back in order fixed the issue and now my application is singing a happy tune again.

Conclusion? When implementing a caching strategy make sure and take the size of the FIFO buffer and the uniqueness of your intended queries into account. The "Server Monitor" has a nice feature under Statistics/Database that will show you how many queries are being cached and allow you to browse them and sort by hit count.

The higher the hit count the more effective your strategy (meaning you are reducing round trips to the DB server). How do you know if you are failing? If you have mostly "1s" in the hitcount column and your buffer is full (the number of cached queries equals the maximum size of your FIFO buffer) then you are likely losing most of the benefit you might have accrued.

Personal Note

The muse has been failing his readers this year and not producing posts at a useful rate. On my list of goals for the fall is to post at 5-8 times per month, so stay tuned. Hopefully I will not disappoint.