Can Performance Suffer With Cfqueryparam?

Posted At : November 18, 2008 9:18 PM | Posted By : Mark Kruger
Related Categories: ColdFusion

I heard an excellent presentation by CF giant **Charlie Arehart** yesterday. It was one of the "unconference" sessions title CfMythbusters. Later that day I was priveledged to share the mic with Charlie and talk bout CF Troubleshooting. Anyway, while discussing cfqueryparam Charlie said something that made me sit up a bit and say "huh?". It has long been the contention of myself and others that the use of Cfqueryparam benefits both *security* and *performance* when used against an RDBMS like SQL server or Oracle. While this is a generalization it usually holds true. Charlie, however, illustrated to me a case where cfqueryparam might be *detrimental* to performance and he was so convincing I thought I would share it with you.

*NOTE: Check out the comments for some cavaets and opposing viewpoints. Also note that the tip on constants may not hold water. See **this discussion** on Brad Wood's blog for more insight on that item*

Excution Plans

First a quick primer to refresh your memory. The reason that cfqueryparam is often faster is that it allows an RDBMS to take advantage of pre-compiled execution plans. Consider this query.

```
<cfquery ...>
    SELECT     fname, lname
    FROM    users
    WHERE ID IN (<cfqueryparam
                    cfsqltype="CF_SQL_CHAR"
                    value="#idlist#"
                    list="yes"/>)
</cfquery>
```

The first time it sees it the DB server will pick out the best way to execute this query and save the query pattern allong with the execution plan. The DB Server might say to itself:

> *DB, everytime you see a query that matches "select fname, lname from users where ID IN (*an array of ids*)" make sure and use the xyz index.*

Once that plan is in place the DB listener simply looks for the same pattern query. When it sees "select fname, lname from users where id in (*an array of ids*)" it says "Aha!" (or possiblye Eureka! if you are using the Greek character set) I know what to do with this one. So cfqueryparam *does* usually result in performance gains because it allows the DB server to figure out way in advance the best way to execute your queries and then use that method (the "cached" execution plan) without the need to figure it out again and again. Seems very simple right? What could go wrong?

One possible Problem

Even given the facts above there are times when the use of cfqueryparam *does not* improve performance? At least one reason why has to do with that idea mentioned earlier of the *first time*. Think about it for a minute and it will become clear. If the

efficiency of the execution plan is determined by the first time the query pattern is established, then that first query is important. It becomes the basis for any subsequent queries that match the pattern. But as you might have guessed, the most efficient way to run a query *does sometimes depend on the variables you pass in to it*. Take the example above. If the first time it is run the "idlist" is sequential as in 1,2,3,4 - then hypothetically the system might say to itself, "DB, the id is the primary key, clustered and unique - let's use a table scan" (again, hypothetically - this is not a discussion of DB execution plan algorithms).

Now consider what would happen if a subsequent query is a non-sequential list as in (55,18,20038,72,1). Even though an index might be a better choice for this second query, the cached execution plan will stick with the first plan. Do you see the problem? Subsequent queries that might benefit from a different plan are going to be shoehorned into whatever the DB came up with to begin with. The DB is going to sit in it's rocking chair like an old geezer and say, "I a-been a-doing it this way for 10 hours and I ain-a-gonna change fur you or anybody!"

The Fix

I'm still wrapping my head around this issue, but here are a couple possibilities. First, make sure your queries use the same "type" of data in the same way. For example, I've seen function that are set up for search also be used to retrieve individual records. Consider the likely groups of params that will be passed into the query and build your queries to service that particular type or group of params. Don't be so married to code reuse that you build enormous queries that do everything for you in one query when 2 or 3 might actually be more efficient (do you really need to join the same table 4 times?). Finally there is the Calvinist method. Add index hints to predetermine the execution plan for the DB engine. I'm looking forward to some suggestions from the savvy CF/SQL readers in the audience.

An Additional Free Tip

NOTE: I used to have a tip here about making sure that static variables (i.e. "active = 1" where also paramed to insure the use of the plan cache. Now it turns out I was mistaken about that. See the comments below for some more definitive information. The comment from Christoropher Secord explains a way to test when your cache is updated or not (very useful - thanks Chris).

Hopefully this post gives you one more way to analyze your queries and a new insight into cfquery. As always, if you have constructive input I welcome and look forward to your comments.