Ask-a-Muse: Killing the Immortal Thread

Posted At : June 9, 2009 11:18 AM | Posted By : Mark Kruger
Related Categories: ColdFusion, Coldfusion Troubleshooting

> *Muse Reader Joe Asks:*
> How do I kill a request? Every other day or so there will be a runaway process that cannot be killed. Clicking on the red exclamation in the monitoring tool does not give an error but it does not kill the request either. My question is how to kill this process?

Ah the immortal thread - like a god coming down from Mt. Olympus and laughing with his (or her) hands on his mighty hips (see why I chose "his"? ... "her mighty hips" ... well, I just didn't want to go there). Such threads are mind bogglingly frustrating. In actual fact, there are some requests spawned by ColdFusion that *may not be able to be terminated* by ColdFusion. For the long version read on McDuff.

The Natives are Restless

Stop and think for a moment how most requests are handled in ColdFusion. I think of a request as a traveler jumping through various gates. Take a form post from a typical forum application:

- **Request:** Hey I have some form data here from Ben Forta. What do I tell him?
- **ColdFusion:** Hang on request, nothing gets back to him till it's been validated.
- **Request:** Validated? Do you know who this is? BF wrote the book on you. I have to believe he knows what he's doing.
- **ColdFusion:** Yeah yeah... I love him too, but orders is orders. No validation, no return.
- **Request:** Geez... ok, validate me.
- **ColdFusion:** Hmmmm... let's see here, login session ok, length of message ok, post date ok. Looks good Request.
- **Request** Great! It's been like 4 milliseconds and I think he might be getting antsy. What do I tell him.
- **ColdFusion** Not so fast there RQ. According to my checklist your next stop is DB.
- **Request:** DB? You mean Carl right? Oh man... I hate that guy. He's always going on and on about he's the most sophisticated piece of software in the world and how his drive is bigger than everyone else's. You know my logging agent tells me size isn't really that important.
- **ColdFusion:** Whoa there big fella... too much information. You sound like a spam agent. Off you go to the DB and *make sure and let me know when you get back* - ok? [hands request off to DB] [sets up to handle return value from the DB]
- **Request:** [to the DB Server] Carl... hey Carl... I have some information from Mr. Forta here and ...
- **Carl (the DB):** Forta Schmorta, wait your turn. It's not like I have spare cycles just waiting around you know.
- **Request:** Yeah, ok... I'll just hang out here then... [hums the theme from Tetris and twiddles his memory registers]
- **Carl:** Do you mind? I'm trying to re-index a table and I have 4 locks on standby.
- **Request:** Sorry.... [waits silently for another 30 miliseconds]... uh Carl, do you have an idea when you might be ready for me?

- **Carl:** Ok, I'm ready - what's the big hairy deal?
- **Request:** Great! Ok, I have an insert and I need the identity field back for the server. Cool?
- **Carl:** Sheesh, is that all? Why don't you web requests ever give me a batch I can sink my teeth into? Not like the ERP, now there's a process I can work with. Here's your ID.
- **Request:** [Rolling His 1's and 0's] ... yes well, thanks Carl - see in a few million cycles. [triggering the callback for ColdFusion] ... Ok, here's the ID, what do I tell Mr. Forta?
- **ColdFusion:** Great, let me check my list.... validation, Carl, ... ah, here we go. You have to send an email.
- **Request:** (sigh) I guess Mr. Forta will have to wait another 100 milliseconds.

Now in the middle of my little play you might have noticed something. ColdFusion "hands off" the request to the DB and then "waits" for the DB to return the request back. This little square dance that happens under the hood is why some requests are bullet proof and can't be terminated. While the thread is handed over to the DB the DB has control over it's disposition. It can't be killed by ColdFusion because ColdFusion does not really control it anymore. ColdFusion is in a passive state waiting for the DB to release the thread back to it.

A simple test

Don't believe me? Try this simple piece of code on your MS SQL server DSN.

```
<cfsetting requesttimeout="45">

<cfquery name="hangme" datasource="#mydsn#" result="res">
    WAITFOR DELAY '00:00:02'
</cfquery>

<cfdump var="#res#"/>
```

You will notice that the execution time of the query is at least 2 seconds (it might show 1998 milliseconds or something similar, but very very close to 2 seconds). That is what we would expect, after all the code tells SQL server to simply wait for 2 seconds before returning. Our total request execution time is a few milliseconds more than that - 2 seconds for the query and 2 seconds + 20 milliseconds for the overall request. So far this is what we would expect, right? But what happens when we use a requestimeout setting that is *less than* the SQL delay?

```
<cfsetting requesttimeout="4">

<cfquery name="hangme" datasource="#mydsn#" result="res">
    WAITFOR DELAY '00:00:06'
</cfquery>

<cfdump var="#res#"/>
```

What you might *expect* to happen is for the request to terminate after 4 seconds with a timeout error. But what *actually* happens is that the request will run a full 6 seconds, then throw a timeout error on "cfoutput". Cfoutput is used by the cfdump tag - so the timeout actually occurs when executing the code *immediately after* the cfquery tag returns.

In other words, ColdFusion knew that it needed to timeout at 4 seconds, but it had to *wait till the DB returned (6 seconds)* before it could throw the error and terminate the request. So ColdFusion can't terminate the request until after the DB returned *even though the request ran longer than the specified timeout*. This clearly illustrates the problem.

More Fun With SQL Threads

Here's one more tip that can actually come in handy. If the problem is that a DB query is hanging (blocking or running too long etc), you can kill the DB query and release the thread. Here's how you can set up a demonstration.

First, create a data source to your MSSQL server with a specific user that is not used anywhere else. This helps "find" the thread in question in the process viewer (in EM or studio). All you have to do is sort by users and it should stand out to you. Next, open EM or Studio and go to the process info (called "monitor" in MSSQL studio). Leave that screen up and run this code.

```
<cfsetting requesttimeout="500">

<cfquery name="hangme" datasource="#mydsn#" result="res">
    WAITFOR DELAY '00:04:00'
</cfquery>

<cfdump var="#res#"/>
```

This will create a request that hangs for 4 minutes. You now have a hanging thread and running request. Leave the page up in front of you and turn to enterprise manager. Find the process in question and double click on it (you might need to "refresh" the process list). You should see the SQL statement from your query (WAITFOR DELAY '00:04:00') in the little window with a button that says "kill process". Click on the button and confirm that you want to kill the statement. What *should* happen is that the ColdFusion request should return immediately with an error - something like DBMS returned an unspecified error. What you have done is terminated a thread that was unterminateable from within ColdFusion. But you did it by terminating it from the other end. You killed the process *in control* of the thread.

Additional Information

ColdFusion Guru **Charlie Arehart** helped with my understanding of this issue. He notes the following:

- Any tag with an external component to it (CFHTTP, CFQUERY, CFINVOKE, CFFILE, etc) would be a candidate for such a condition.
- To pin-point the issue, find the thread in question and do a stack trace. You can do a stack trace manually by starting CF from the **command line** (a huge hassle but doable). If you have ColdFusion 8 Enterprise the server monitor allows you to do a stack trace with "profiling" enabled. Finally, both **SeeFusion** and **Fusion Reactor** have mechanisms for trapping a stack trace on an individual thread.
- There's a handy **online tool** provided by Fusion reactor for analyzing a trace.

Charlie also mentioned a "special case" of a hung thread that is a result (according to the experts at Fusion Reactor) of a "reflexive bug in IE" which keeps the request open. There is a discussion of this bug at this **Google Group** address. I'm sure Charlie will

probably blog this information with his usual thoroughness at some point. Thanks Charlie!

Finally, let's take note that there *is* a programmatic way to terminate a thread in ColdFusion using the Admin API component. It should be noted that using this method will *not kill* a thread that is controlled outside of the JVM. This is the same command that is used from within the ColdFusion Monitor (clicking on the little red x). So if it does not work there it will not work here either. Still, in some circumstances it might do the trick.

```
<!--- authenticate --->
<cfset adminObj = createObject("component","cfide.adminapi.administrator")/>
<cfset adminObj.login([password])/>
<!--- get monitor object --->
<cfset sMonObj = createObject("component","cfide.adminapi.servermonitoring")"/>
<!--- kill thread --->
<cfset sMonObj.abortRequest("jrpp-100")/>
```

Now dear readers, I'm ready for you to all tell me your tips and tricks for finding and terminating various threads. Have at it (but be polite :).