UNDERSTANDING CFLOCKS: HOW THEY WORK / HOW THEY CAN HELP OR HURT

Charlie Arehart, Independent Consultant
CF Server Troubleshooter
charlie@carehart.org
@carehart (Slack, Github, X, Fb, Li, Skype, etc.)

Updated Sep 1, 2024

- In my experience, folks have a lot of misconceptions about use of cflock
  - (cflock or lock script statement)
- Your code may use them a lot (or a lot more than you realize)
  - or you may use few of them
  - or you may even truly use none
- Either situation could be problematic, under certain circumstances
- Sadly, CF (and Lucee) offers no insight into whether locks are hurting (or helping)

Charlie Arehart @carehart
charlie@carehart.org

TODAY'S TOPICS/DEMOS

- Basic example: what cflock does
- Key control mechanisms…which may NOT do what you presume
  - **Name** vs **Scope** locks
  - **Readonly** vs **exclusive** locks
  - Lock **Timeout**
  - **Throwontimeout**
- More examples: including why getting these things wrong can be devastating!
- Related matters, resources for more

Charlie Arehart @carehart
charlie@carehart.org

ME.ABOUT()

- I focus on CF/Lucee server troubleshooting, as an independent consultant
  - Assist organizations of all sizes, experience levels
  - Work remotely: safe, secure, easy via **shared desktop** (zoom, meet, teams, etc.)
  - **Solve most problems in less than an hour**, teaching you also as we go
  - **Satisfaction guaranteed**
  - More on rates, approach, online calendar, etc at *carehart.org/consulting*
- But to be clear, I'm not selling anything in this session! ☺
  - Just sharing my experience, and that of others
- Presentation online at *carehart.org/presentations*

Charlie Arehart @carehart
charlie@carehart.org

BASIC EXAMPLE: WHAT CFLOCK DOES, IN A SINGLE REQUEST RUN ALONE

- What would you expect this to do if you ran it, alone?

```
<cflock timeout="1">
        <cfset sleep(5000)>
</cflock>
```

- Here's a variant in cfscript (which can be done either of two ways):

```
//cflock(timeout="1") {
lock timeout="1" {
        sleep(5000);
}
```

Charlie Arehart @carehart
charlie@carehart.org

- And what would you expect it if you ran that in two requests at the same time?
- Let's try it…with some additional info offered to help clarify things

```
<cfoutput>
    <h2>I'm creating a lock and sleeping within it for 5 seconds</h2>
    <cfflush>

    And I'm willing to wait to obtain a lock for up to 1 second<br><br>
    <cflock timeout="1">
        <cfset sleep(5000)>
        Sleep within lock finished<br>
    </cflock>
</cfoutput>
```

BASIC EXAMPLE: WHAT CFLOCK DOES, IN CONCURRENT REQUESTS

Charlie Arehart @carehart
charlie@carehart.org

- So we see that locks are more a "*gentlemen's agreement*"/crossing guard
  - All about whether two or more requests try to obtain same lock at same time
- And there are available NAME and SCOPE attributes **giving more control**
  - Also optional TYPE (readonly or exclusive)
  - There's also a THROWONTIMEOUT to control what should happen upon timeout
- As you can see, these four are not required by CF
- Let's discuss these next
- Code examples are available online:
  - *gist.github.com/carehart/*

WE'LL EVOLVE THESE EXAMPLES

Charlie Arehart @carehart
charlie@carehart.org

KEY CONTROL MECHANISMS

Charlie Arehart @carehart
charlie@carehart.org

LOCK TIMEOUT

- As we saw, this does NOT control how long a lock is held!
  - It's how long a cflock is willing to WAIT to obtain the lock it seeks
- I have a more flexible example, letting us pass in sleep and timeout as URL vars
  - Let's demo
- **Key myth busted**:
  - **Lock timeouts do NOT control how long CF will HOLD (or is willing to hold) a lock**
  - They are solely about how long CF is willing to WAIT to obtain a lock
- Note:
  - Lucee doesn't even require timeout (seems to default to 50 seconds)
  - CF docs say timeout of 0 means use requesttimeout
    - I find instead it literally sets a 0-second timeout

Charlie Arehart @carehart
charlie@carehart.org

NAME LOCKS

- Notice we didn't use a NAME or a SCOPE in our first examples
  - Locks without either are called "anonymous' locks
  - They affect only calls to the template in which they run
- Giving lock a NAME offers more flexibility:
  - Extends the locking to any request using a lock of that name
  - Name can literally be anything you want it to be
  - Let's see demo…
- Name can be generated dynamically (is just a string)
- Note that name locks are instance-wide (not app-specific)
- Before we discuss scope locks, let's first discuss that lock TYPE…

Charlie Arehart @carehart
charlie@carehart.org

LOCK TYPE: READONLY OR EXCLUSIVE

- You may have noticed we had no lock TYPE in our first examples
  - If not specified, the type is EXCLUSIVE
  - The other option is READONLY
- But we weren't "reading" anything in our examples (not even a variable)
  - The type is more for declaring the *intent* of what's going on in the lock
- The key is that if a given lock is held as EXCLUSIVE, it blocks all requests for that lock
  - If a lock is READONLY, other requests for that same READONLY lock can run
    - But any request for that lock as EXCLUSIVE will be blocked
- Let's demo…

Charlie Arehart @carehart
charlie@carehart.org

SCOPE LOCKS

- Now finally we can discuss SCOPE locks
  - Valid values are SESSION, APPLICATION, SERVER, and REQUEST
  - Extends locking to apply to any request using a lock of the same scope
- Essentially the same as a NAME being "the app name" or "the session id"
  - Let's demo (session and app scope locking)…
- Note: while a scope lock is held, another request CAN access that scope!
  - Let's demo (how scope is not "locked")…
- Request scope lock can be helpful with need for cflock in cfthread
- **Key myth busted:**
  - **Scope locks do NOT "lock access" to a scope**:
    - They are simply a different way of "naming" a lock, like any we've seen
    - They do not "lock" the scope!

Charlie Arehart @carehart
charlie@carehart.org

AREN'T WE "SUPPOSED TO LOCK"?

- Wait, weren't we all told we should "lock whenever we access a shared scope variable"?
  - This would be primarily session and application, but also server
- Here's the thing: this was true…when?...anyone?
  - Prior to CF6…at the turn of the century, when CF5 and earlier ran on C++
  - When CF6 came out, running CF on Java, the need vastly diminished
- **Key myth busted:**
  - **We no longer NEED to *ALWAYS* lock access to shared scopes…**
- …but aren't there times we SHOULD?….

Charlie Arehart @carehart
charlie@carehart.org

WHEN "SHOULD" WE CONSIDER LOCKING ACCESS TO SHARED SCOPES?

- We DO still need to consider locking our use of shared scopes…but when, primarily?
  - On a "RACE CONDITION"
  - Essentially, if two requests could run at the same time and modify the same variable
- This is especially an issue when a variable holds an incrementing value
  - Two requests COULD run at the same time, and while one reads and increments, the other does
- That said, if code always sets value, that's rarely an issue, like:
  - application.prod_dsn="mydsn";
- **Key conclusion:**
  - We SHOULD lock potential "race conditions"

Charlie Arehart @carehart
charlie@carehart.org

WHERE CFLOCKING CAN HELP

- Besides race conditions, where else could cflock help?
  - Anytime some action should somehow be single-threaded (only one request at a time)
  - This is a frequent use of named, exclusive locks
- That could be about writing to some file, or processing files
- Or even running some tag that's not well-suited to concurrent access
  - There was a time we needed to worry about that regarding cfindex/cfcollection
  - See interesting recent example with lucee 6 and cfschedule
    - *dev.lucee.org/t/lucee-6-1-0-243-all-contexts-settings-being-wiped/14243*
- Remember, though: cflock is a gentlemen's agreement
  - If a request tries to access "what is locked" (file, etc.) without using cflock, it can!

Charlie Arehart @carehart
charlie@carehart.org

WHAT ABOUT DATABASE LOCKING?

- This is an interesting one: some people think they should cflock SQL calls
  - The thing is, nearly all databases already implement their own locking for us
- Again, your use of cflock **does NOT prevent others accessing DB**
  - It only affects others ALSO using the same lock (***name***/***scope***), subject to ***type***
- **Key myth busted:**
  - It's generally NOT appropriate to rely on CFLOCK to enforce db locking/contention

Charlie Arehart @carehart
charlie@carehart.org

THROWONTIMEOUT

- Finally, we can move on to the last attribute: THROWONTIMEOUT
  - It simply controls what should happen if a lock cannot be obtained in TIMEOUT time
- Default is "true": an error will be thrown, as we saw in very first demo
  - Let's demo with more recent "flexible" code, passing in low timeout for long sleep…
- So what happens if throwontimeout="false" (or "no", etc.)?
  - Anyone want to guess?
  - Does it run the code in the lock anyway?
    - NO!
- Let's demo…
- It literally SKIPS the code in the lock!!!
  - Which can be a TERRIBLE underlying cause of subtle bugs: why does xxx not exist?
- **Key myth busted:**
  - **Don't use throwontimeout unless you are VERY careful about and aware of it**

Charlie Arehart @carehart
charlie@carehart.org

- So we've seen that locks can be held for a long time
  - And technically code WITHIN a lock can HOLD a lock for a long time
  - Can be silent killer in a lot of CFML apps
  - Key takeway: **don't hold locks any longer than necessary**
    - Think twice about doing long-running actions within a lock, like cfquery, cfhttp, etc.
- Here's a related topic: consider nested lock checking
  - If you're locking to "do something that needs to be done"…
    - Check WITHIN the lock whether you "still need to do it". If not, skip "doing it"
  - helpx.adobe.com/coldfusion/developing-applications/developing-cfml-applications/using-persistent-data-and-locking/locking-code-with-cflock.html***#Nestinglocksandavoidingdeadlocks***

LOCK CONTENTION: ISSUE/OPTION

Charlie Arehart @carehart
charlie@carehart.org

- Wouldn't it be nice to know how long requests are HOLDING or AWAITING locks?
  - Or whether a request had a lock timeout, or ignored one?
  - Sadly, neither CF nor Lucee offer any means to track all these, not even in debug info
  - Lucee does add a **RESULT** attribute that can track the *executiontime*
- And CF/Lucee Admin also has no mechanism to control cflock mechanism
  - Nor any means to manage/monitor them
- I feel like this is a missed opportunity, and I've asked about it before
  - *tracker.adobe.com/#/view/CF-3036835*
  - That's from 2008! Many votes, nothing ever done :-(

HOW CF AND LUCEE LACK ANY DIAGNOSIS OF CFLOCKS

Charlie Arehart @carehart
charlie@carehart.org

- CF2016 added synchronized arrays/structs (see docs for arraynew)
- CF2018 added runAsync()
  - *helpx.adobe.com/coldfusion/cfml-reference/coldfusion-functions/functions-m-r/runasync.html*
  - *helpx.adobe.com/coldfusion/using/asynchronous-programming.html*
  - *modern-cfml.ortusbooks.com/beyond-the-100/asynchronous-programming*
- Lucee offers a cfdistributedlock, which supports locks across servers via Redis
  - *docs.lucee.org/reference/tags/distributedlock.html*

RELATED TOPICS WE WON'T DISCUSS

Charlie Arehart @carehart
charlie@carehart.org

RESOURCES

- Beware that some resources have misstatements that our examples can prove
- *helpx.adobe.com/coldfusion/cfml-reference/coldfusion-tags/tags-j-l/cflock.html*
- *helpx.adobe.com/coldfusion/developing-applications/developing-cfml-applications/using-persistent-data-and-locking/locking-code-with-cflock.html*
- *helpx.adobe.com/coldfusion/developing-applications/developing-cfml-applications/using-persistent-data-and-locking/examples-of-cflock.html*
- *docs.lucee.org/reference/tags/lock.html*
- *carehart.org/blog/2022/6/24/understanding_cflock_cost_part_1*
- *modern-cfml.ortusbooks.com/cfml-language/locking*

Charlie Arehart @carehart
charlie@carehart.org

SUMMARY

- CFLock (and the cfscript cflock/lock) are important tools, powerful
  - With power comes responsibility
- We've seen the options of anonymous, named, or scope locks
  - With optional type of readonly or exclusive
  - With timeout that controls how long to WAIT for a lock
  - And throwontimeout that can "ignore" a failed request for a lock
- Be careful out there
- Reach out to me with questions on talk/share feedback (direct or public)
  - Slack, Github, X, Fb, Li, Skype, etc. simply as @carehart
  - Email: charlie@carehart.org
- Again, presentation online at *carehart.org/presentations*

Charlie Arehart @carehart
charlie@carehart.org