IIS 7 Max Worker Processes and ColdFusion Updated

Posted At : November 5, 2011 11:12 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Troubleshooting

In my previous post on this topic I indicated that IIS 7 seemed to be a constraining factor. That post lead to conversations with a couple of CF gurus (Charlie Arehart and Russ Michaels) who clued me in to a number of additional settings. If you are truly interested take the time to read the previous post and (especially) the comments before you read this post. What bothered me was that the issue I discovered (a cap on requests) can be affected by both IIS settings or JRUN settings (or both).

My conclusion is that the behavior I was trying to affect is actually the bug that Charlie pointed out to me on Adobe's site (found here). Charlie rightly indicates that this issue is under-recognized (I certainly had not run into yet). The *behavior* of this bug can be affected (fixed or mitigated) by adjusting IIS as described in my previous post as well as by using the Adobe-provided instructions. This lead to a bit of Muse head-scratching. How do these various processes really work together? This post hopes to clear that up (or at least add to our compendium of knowledge).

First, the short answer is that Charlie is right that the maxworkerthreads setting in the jrun_iis7_wildcard.ini file will solve this problem. Starting with a default setting of 1 for "Maximum Worker Processes" in the IIS7 application pool settings and uncommenting the maxworkerthreads and setting it to 50 (maxworkerthreads=50) or any number at least as large as your simultaneous requests setting will remove the cap of 25 maxworkerthreads just exactly as Charlie insisted. And I should know better than to doubt our resident CF scholar :)

To answer the *other* question however the maxworkerthreads setting is clearly *multiplied by the IIS7 maximum worker processes* setting. So when you play with these settings keep in mind that setting the maxworkerthreads to say 50 in your jrun_iis7_wildcard.ini file and then setting the maximum worker processes to 2 in the IIS 7 application pool settings will result in ColdFusion potentially handling *100 requests* concurrently (2 IIS worker processes times 50 max worker threads).

To prove this to myself I started with the following settings:

- IIS 7 Max Worker Processes = 1
- maxworkerthreads= 10 in jrun_iis7_wildcard.ini for my instance
- Simultaneous Requests = 32 in CF Admin.
- A single CF instance with a single site

Then I used my load test to put enough pressure on the server to max out the CF request queue (this was a baseline I knew from previous testing). I expected to be limited to 10 requests. Sure enough, as expected I capped out at 10 requests. Take note of how this works. This particular limit is bound to the IIS worker process. It's about the *connector* not the *application engine*. That means the requests queue in IIS - not in CF. In other words you will *not* see queued requests climbing in Fusion Reactor or the CF monitor. But if you use perfmon you *can* see them queing in IIS.

With my test running and observing my "active requests" steady at 10 with excess requests queuing in IIS and not in CF, I set the "max worker processes" to 2. My cap bounced up to 20. Requests were still queuing in IIS (and not in CF). I bounced to 3 and my cap held steady at 30. No CF queue yet (remember my simultaneous requests setting is 32). When I bounced max worker processes to 4 however my "active requests" made it to 32 but no further. Instead I now had a queue *in ColdFusion* of 8. With a max worker process of 5 I got a queue of 10 to 12 (that was the limit of my test tolerance).

Lessons Learned

Remember this default capping behavior is *per connector* not *per instance*. One of my servers running in multi-server mode has 24 or 25 sites on it - each of them connected to a specific instance with its own jrun_iis7_wildcard.ini file and its own application pool. So the total number of potential requests handled by that instance is a multiple of application pools x max worker processes x maxworkerthreads (or the default of 25 if that setting is commented out as it is by default).

To put it another way the overall number of connectors you are using against an instance will multiply the number of potential concurrent requests forwarded to ColdFusion. If you have a single instance with say 10 sites each with its own application pool and each connected to the same ColdFusion instance, you will end up with 10 worker process threads and therefore 10 times 25 (if using the default) maximum concurrent CF requests.

So shared servers with multiple sites might appear to *not be affected* by this bug. But if you think about it they *are* affected. With the default settings no app pool can get more than 25 requests active concurrently. That's a problem that could easily "hide" on a shared server (especially a busy one).

Is this useful in some way? It occurs to me that on a shared server you could create a sort of throttle behavior by adjusting these settings - limiting a specific site to only 10 concurrent requests sent to JRUN at a time. Let's say you have a particular site (a batch instance for example) with a tendency to gobble up too many threads, but you don't have enough resources on the server to give it its own instance. Instead you might limit the amount of work that the particular instance can do on behalf of that site by setting maxworkerthreads to some number (say 10) and max worker processes to 1.

Incidentally I do *not* object to having more than one "worker process" (w3wp.exe) in IIS as long as you don't go crazy. Indeed, in IIS7 each "site" is created with a new application pool (unless you reuse them) and each application pool spawns its own w3wp.exe process. These worker processes tend to have a small footprint overall and using 2 or 3 instead of 1 is a fairly modest configuration change that might actually improve your performance in some cases.

In any case it is interesting to see how these settings work together and I feel like exploring this process has added some additional information to our growing arsenal. Thanks to both Russ (a.k.a. "Snake") Michaels and the inestimable Charlie Arehart for poking at me on this and helping me figure out the relationships between these various settings and processes.