IIS 7 Constraining Simultaneous Requests Limit?

Posted At : November 4, 2011 10:37 AM | Posted By : Mark Kruger
Related Categories: Coldfusion Troubleshooting

I have been doing some performance testing for a company with a large server farm over the last couple of weeks. Although the farm had 20 or more servers, we started with just one sever to try and get some numbers we could use to extrapolate the overall tolerance of the larger system. The servers were all Windows 2008r2 64bit, IIS 7, running CF 9 enterprise with plenty of RAM. We were also running Fusion Reactor to help introspect ColdFusion.

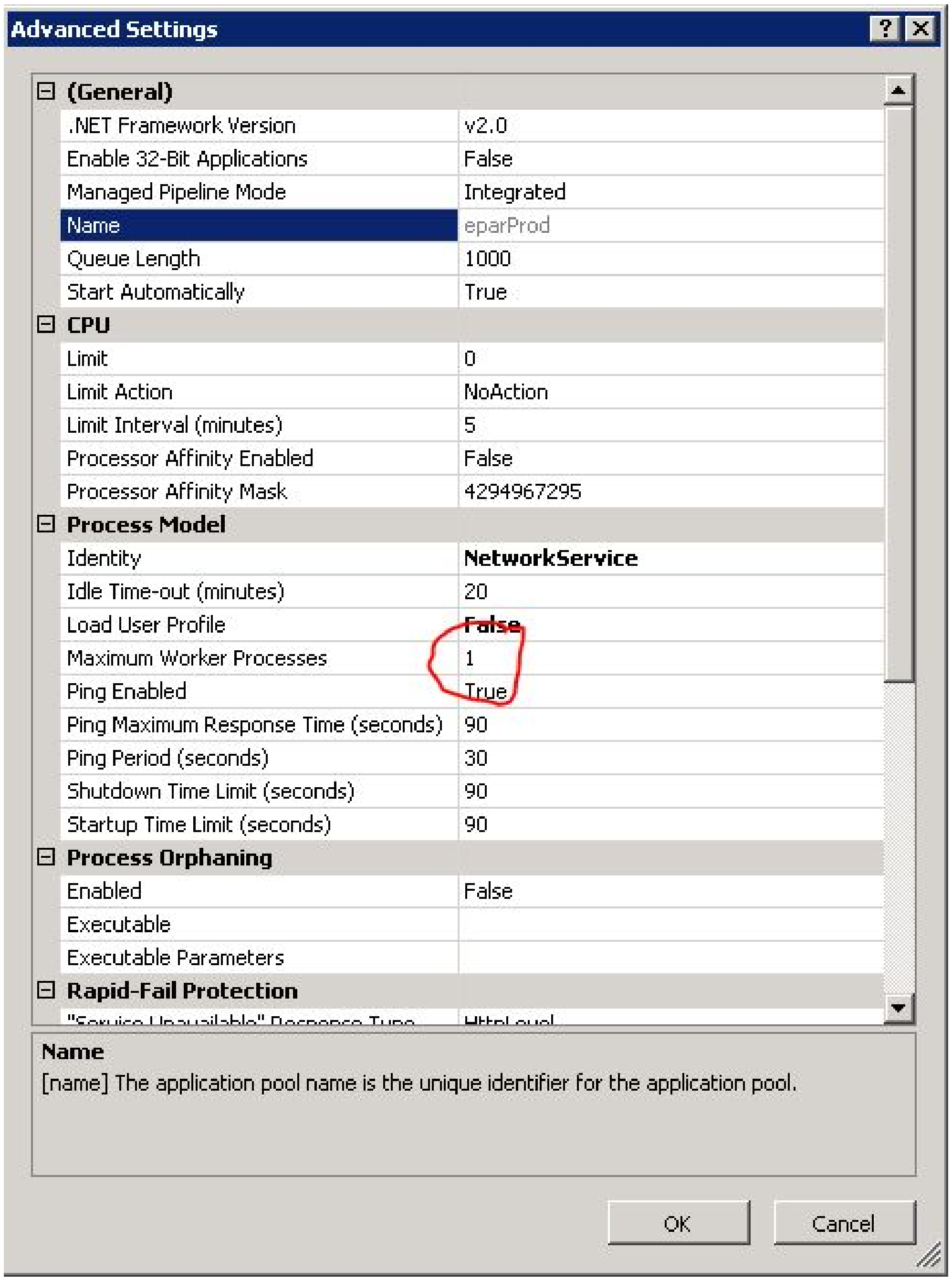
As I slowly poured on more load I noticed something strange that I had never seen before. Although my "simultaneous requests" setting was set to 48. I could not get ColdFusion to handle more than *25 active connections*. Under ordinary circumstances I could pound away at a server using my test framework and get enough requests active to *overrun* that simultaneous request setting and see the queue kick in. I was trying to max out the server but it was not behaving as expected. Active requests would "cap out" at 25 - as if my simultaneous request setting was set to 25 - but there were never any requests in the request queue. It was a head scratcher - but I kinda love those! Here's the skinny....

(NOTE: The comments on this post are important as well. And this **Follow up** post clears up some of the confusion.)

A quick look at Windows performance monitor "web service->current connections" showed it climbing well above 25 - 75, 80 and 90 connections. What I surmised is that instead of letting ColdFusion queue the requests in the JVM, somehow IIS was actually queuing them and passing a max of 25 concurrent threads over to ColdFusion.

The Fix

I'm not sure how or why, but I do know the what. The setting to change is the "Maximum Worker Processes" in your application pools. You find this setting by going into IIS manager, drilling down to the application pool you need to adjust and choosing "advanced settings". You'll see the setting on the properties dialogue. It will look like this:



Advanced Settings

(General)	
.NET Framework Version	v2.0
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	eparProd
Queue Length	1000
Start Automatically	True
CPU	
Limit	0
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	False
Processor Affinity Mask	4294967295
Process Model	
Identity	**NetworkService**
Idle Time-out (minutes)	20
Load User Profile	**False**
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Enabled	False
Executable	
Executable Parameters	
Rapid-Fail Protection	

Name
[name] The application pool name is the unique identifier for the application pool.

In IIS 6 this used to be called "web farms" but I never remember it constraining so precisely like this. I set it to a multiple of 25 that got me above my simultaneous request setting. So if I was using say 64 I would set it to 3 (3 times 25 = 75). Sure enough when I put my extra load on the server perfmon "current connections" were reduced somewhat as CF began to handle more requests, and CF active requests now spiked above 25 and were able to queue as I expected. Take note that when you do this you are allowing IIS to spawn additional threads (and allocate memory and cpu cycles etc) so you can have a positive OR a negative impact depending on your resources and the load you are expecting.

Perhaps one of my ingenious readers can clue me in on *why* I saw this constraint. I certainly did not see anything specific about the number 25 but that was exactly the number I was held to until I changed this setting. Very curious.