ColdFusion Request Tuning Settings in Depth

Posted At : August 3, 2009 12:12 PM | Posted By : Steven Erat
Related Categories: Flex, Java, Adobe, Quality Assurance, Restaurants, ColdFusion

Undoubtedly, the ColdFusion Administrator settings for Request Tuning are critical to performance of Web applications running in the server. While reading the recent Adobe article on **Performance Tuning for ColdFusion Applications** I was surprised to find the content on this topic to be a little light. With that in mind, I set out to expand on the topic of the Request Tuning settings.

Foundations of ColdFusion Request Settings

To begin, let's look at how the ColdFusion settings were configured in earlier versions of application server. With the release of ColdFusion MX 6.0 through versions 6.1 and 7, all editions of the ColdFusion server had one setting for Request Limits. This was referred to as the "Simultaneous Requests" setting. This single setting throttled the number of running requests to be processed concurrently. Should the running pool be fully occupied by requests that are processing but haven't yet completed, the J2EE server underlying ColdFusion will hold requests in a queued request thread pool that are to be fed to the running request pool.

What's the Best Setting?

As with most articles you may read on the topic, there is no magic setting for the Simultaneous Requests value that would automatically produce the best throughput. Rather, determining the best value is done as an iterative process by carrying out real world load testing during application development and functional testing in the software development life cycle.

CPU-intensive or Heavy I/O?

ColdFusion applications tend to have a heterogeneous mix of templates that require intense CPU utilization, such as when performing string parsing or calculations in loops, or templates that require little system resources, such as those that are database intensive and spend time waiting on results or those that are I/O intensive such as applications that perform lots of network transactions with CFFTP or CFLDAP. Generally speaking, the Simultaneous Requests value will be higher for I/O-centric applications, and lower for CPU-intensive applications. The default value for the Simultaneous Request setting in ColdFusion MX 7 and earlier was set as 8 by default after installation. Few server administrators changed that value, for better or worse, but it was thought to be a reasonable starting point with the generic guideline of 3-5 per CPU.

One approach to determining the best value for the setting is to perform load testing, perhaps with tools such as the freely available and robust **Apache JMeter**, or commercially available and highly scriptable products like **Paessler Web Stress Tool** or **Borland Silk Performer**. The testing should be carried out on a platform and configuration as close as possible to how it will be run in production. As you perform testing while tracking request throughput, repeat the test iteratively while modulating the Simultaneous Request setting from low to high, then graph the results to find the setting for highest throughput on the curve. Very likely, you will notice CPU utilization will reach a high of 60-80% during the trial that produced the best results. When the

CPU is consistently running at higher values, the CPU maybe thrashing which is another way of saying that it is being used inefficiently during the heavy context switching of frequently changing which cpu threads are being executed. Throughput typically declines as the CPU begins to "redline". Tuning ColdFusion request limits to moderately utilize the CPU as a baseline allows room for occasional bursts of traffic (which can be simulated with ramp testing). An example of a burst might be when users first start the application at 9am during the start of their work day, or when the application is popularized through Digg or other social networking sites.

How *Not* to Tune The Request Limits

The Simultaneous Request setting has often been misunderstood as the total number of end users to the application that can be concurrently supported. When administrators tune ColdFusion based on that incorrect premise, they will sometimes scale up the value to be the total number persons they expect to be using the application at the same time, and as such they might set the Simultaneous Requests value to an unusually high value such as 100 or even 500. In the majority of all situations, values this high will negatively affect server throughput by thrashing the CPU during heavy context switching, as alluded to earlier.

Types of Requests

So which type of application requests are throttled by the Simultaneous Requests setting? Up to and including ColdFusion 7, all request types are throttled through the same request pool, including ColdFusion CFM template requests, CFC template requests that arrive via HTTP requests such as AJAX applications, Web Service Requests for exposed endpoints, and Flash Remoting Requests such as those from Flash or Flex presentation layers.

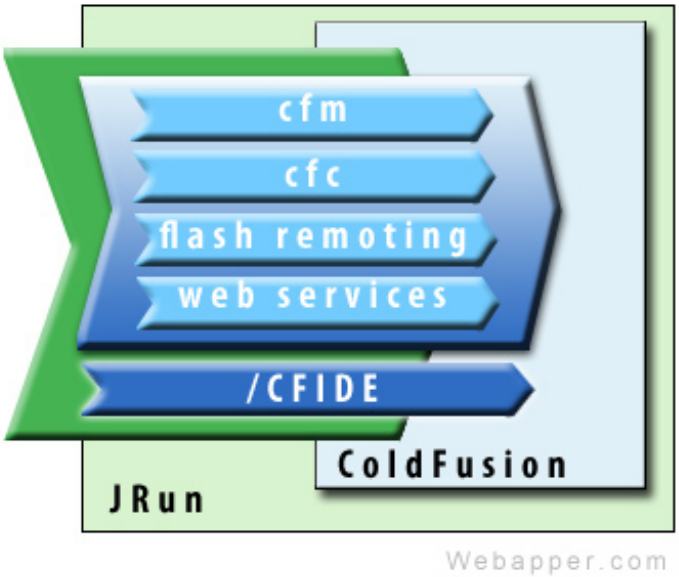
Where Do I Find The Setting?

The ColdFusion Administrator page titled Setting exposes this configuration option. The Simultaneous Requests value is directly tied to the JRun activeHandlerThreads value in the JRunProxyService section of it's corresponding jrun.xml file. For ColdFusion installations deployed to other J2EE containers such as WebSphere or Weblogic, the Simultaneous Request setting does not exist in the ColdFusion Administrator. Refer to the J2EE server documentation for request tuning settings. Commonly on those other J2EE servers, the ColdFusion server must compete with other J2EE applications having additional incoming requests for JSP and Servlets directly. Your tuning strategy will have to consider the total blend of requests across all applications running on that J2EE server.

Current and Future Versions of ColdFusion

With the release of ColdFusion 8 Enterprise Edition a new means of request tuning was introduced by a CF Admin page of the same title, Request Tuning. The ColdFusion 8 Administrator exposes 4 separate ColdFusion pools for granular request tuning. These pools discretely throttle CFM requests, CFC requests, Flash Remoting requests, and Web Service requests. These request pools are downstream of the underlying J2EE

server request pool. Note that the CFC throttle refers to CFCs accessed via HTTP directly, from AJAX applications for example.



ColdFusion 8 introduced 4 types of request pools that throttle independently. These pools are downstream of the JRun thread pool.

ColdFusion 8 Standard Edition, however, does not expose these 4 discrete ColdFusion pools, but rather it behaves exactly the same as ColdFusion 7 where there is a single Simultaneous Request setting.

Advantages and Disadvantages of ColdFusion Request Pools

The advantage of ColdFusion 8 Enterprise Edition is that if your application has different entry points such as an AJAX view for smart phones, a Flex view for desktop browsers, and a typical CFM/HTML view for clients not supporting Flash or JavaScript, then you can tune accordingly. If a link to the AJAX view of your application suddenly rises to the top on Digg and receives a burst of traffic, the other Flex view and CFM view may continue to be available without noticeable disruption.

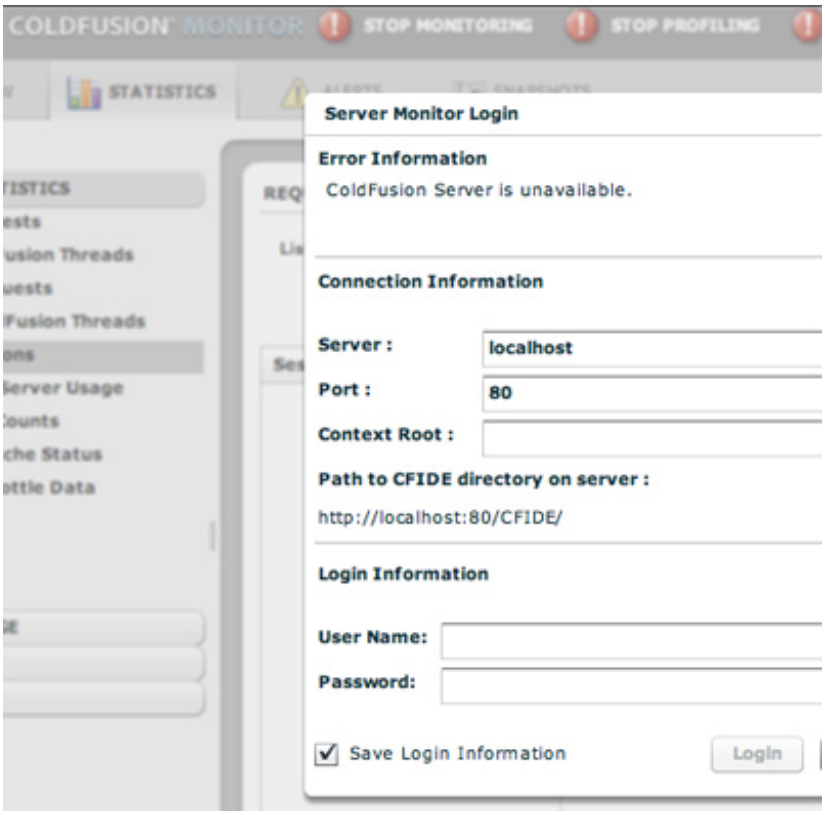
The reason I say *may* continue to be available is because, again, the ColdFusion request pools are secondary to the underlying J2EE server request pool. In this example, if the AJAX view linked from Digg is wildly popular, then you may receive so many requests that the CFC thread pool becomes full and queues, and then the JRun thread pool begins to queue. Once the JRun thread pool is queuing, then the CFM and Flex views to the application cannot be processed while they’re waiting in the JRun queue behind all the AJAXian CFC requests, even though the discrete CFM and Flash Remoting pool types are empty.

Think of it as trying to get into a fast food restaurant during lunch hour in the city. If there’s a drink line, a sandwich line, and a buffet line inside and all you want is a Coke. If the buffet line is backed up and stretching out the door then you can’t get in to fill your cup even if the drink line is empty.

Come On In!

ColdFusion 8 recognized the potential for the individual request pools to become full, making the ColdFusion Administrator inaccessible. A (partial) solution to this problem is that any request starting with /CFIDE in the script name is permitted to bypass the ColdFusion request pools. This means that if the CFM template pool is full occupied and queuing, a request for the ColdFusion Administrator will still be accessible without queuing. Once again, however, this *backdoor* access for /CFIDE remains subject to the J2EE server pool and any queuing that may occur there.

A similar weakness exists regarding the ColdFusion Server Monitor should be noted. The UI to the Server Monitor is built from Flex and runs as a Flash application, and the data that is displayed in the UI is obtained via Flash Remoting requests, as observed by its posts to /flex2gateway servlet mapping. Requests to /flex2gateway do not participate in the /CFIDE bypass. Therefore, when the server is under load and the Flash Remoting pool becomes full and queues, then the Server Monitor is no longer able to retrieve data. Additionally, once the Server Monitor is running, if the JRun server is queuing, then again the Server Monitor may not be able to retrieve any data.



The ColdFusion Server Monitor requires a connection through the JRun thread pool to access the CF Flash Remoting

This weakness in the Server Monitor is where Webapper's SeeFusion monitoring tool shines. SeeFusion is installed as a Servlet Filter and listens on its own port. Requests to the SeeFusion monitoring interface are not subject to the restrictions of the ColdFusion request pools or the J2EE server request pools and is always available regardless of request volume.

Multiserver Considerations

Some admins may choose to install ColdFusion in the Multiserver Configuration to take advantage of JRun's failover and load balancing abilities, or to take advantage of application isolation since each ColdFusion server instance runs in its own JVM. When tuning the request pools, keep in mind during testing that the additive nature involved of the various request pools. If it was previously determined that a CFM Template Limit should be 8 on single instance of ColdFusion, then if you add more server instances then obviously they cannot all be set to 8. Instead the application should be retested for performance while tuning the various instances at the same time for balance and aggregate throughput. In fact, since each JVM requires some CPU just to manage its heap and thread pools, you may find that the number of total ColdFusion Simultaneous Requests per CPU decrease further. That is to say if you had one instance optimally set to a limit of 8 template requests, and you added another server instance for failover benefits, it would seem logical that you would set the template request limit from 8 down to 4 for each instance, but with the added overhead of an additional JVM you may end up discovering that the optimal throughput occurs with a limit of 3 template requests for each ColdFusion instance.

Other Differences Between ColdFusion 8 and Earlier

Recall that in ColdFusion MX 7 and earlier the Simultaneous Request setting was directly tied to the JRun activeHandlerThreads setting in jrun.xml. If you set ColdFusion to 5, then the whole JRun server had a request limit of 5. With ColdFusion 8, the request tuning pools are subsets of the JRun thread pool, with the default values of CFM/10, CFC/5, Flash Remoting/10, Web Service/5. (These values are held in the ColdFusion configuration file neo-runtime.xml.) The total default value is then 20 ColdFusion requests, and the JRun Master Request Limit for running JRun threads is set to a default of 50 where that value directly ties to the JRun activeHandlerThreads setting in the jrun.xml config file.

Notice that in ColdFusion 7 the total size for the JRun thread pool activeHandlerThreads has a default of 8, and that the ColdFusion 8 default JRun activeHandlerThreads setting is 50, about a 6-fold increase. The overall JRun defaults were increased in ColdFusion 8 based on the performance improvements observed with the upgrade to JVM 1.6 and the other internal ColdFusion server performance enhancements. Its wise to be aware of the difference in these default pool sizes because an upgrade without load testing may result in inadequate default request tuning settings.

As mentioned earlier, on other J2EE server types such as WebSphere, the ColdFusion 7 Simultaneous Request setting is not available, however with ColdFusion 8 the individual CFM/CFC/FR/WS pools are accessible for ColdFusion Enterprise on those non-Adobe J2EE servers.

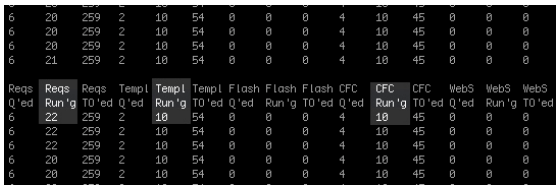
CFThread

The use of CFThread in ColdFusion 8 applications requires its own discussion apart from what's covered in this article. Notice that CFThread requests do not arrive via HTTP Requests from end users, and therefore are not subject to the ColdFusion Request Limits. The ColdFusion Administrator provides a separate setting for CFThread. In Enterprise Edition the default value for CFThread limit is 10 but may be increased upwards as needed. In Standard Edition the hard limit for CFThread is 10. It

has been suggested in a comment on **Ben Nadel's blog** that for applications spawning high volumes of CFThreads that you may need to adjust the jrun.xml JRunSchedulerService setting for activeHandlerThreads, not to be confused with the setting of the same name in the JRunProxyService section. I haven't yet validated that statement, so you may want to do so yourself. If you begin to tune the JRunSchedulerService settings, be aware that the maxHandlerThreads should not be larger than the activeHandlerThreads setting for the JRunSchedulerService because those threads do not die when idle, so that type of request pool does not shrink, but can only grow until the max limit.

Watch it!

With the addition of these new ColdFusion thread pools in ColdFusion 8, the commandline utility cfstat has been enhanced with 12 new columns to report the new values (including running, queued, and timedout). To see these columns in ColdFusion 8 cfstat, use the -x switch for eXtended metrics, as in cfstat -x 1 to output extended data every second.



Reqs Q'ed	Reqs Run'g	Reqs TO'ed	Templ Q'ed	Templ Run'g	Templ TO'ed	Flash Q'ed	Flash Run'g	Flash TO'ed	CFC Q'ed	CFC Run'g	CFC TO'ed	WebS Q'ed	WebS Run'g	WebS TO'ed
6	20	259	2	10	54	0	0	0	4	10	45	0	0	0
6	20	259	2	10	54	0	0	0	4	10	45	0	0	0
6	20	259	2	10	54	0	0	0	4	10	45	0	0	0
6	21	259	2	10	54	0	0	0	4	10	45	0	0	0
Reqs Q'ed	Reqs Run'g	Reqs TO'ed	Templ Q'ed	Templ Run'g	Templ TO'ed	Flash Q'ed	Flash Run'g	Flash TO'ed	CFC Q'ed	CFC Run'g	CFC TO'ed	WebS Q'ed	WebS Run'g	WebS TO'ed
6	22	259	2	10	54	0	0	0	4	10	45	0	0	0
6	22	259	2	10	54	0	0	0	4	10	45	0	0	0
6	22	259	2	10	54	0	0	0	4	10	45	0	0	0
6	20	259	2	10	54	0	0	0	4	10	45	0	0	0
6	20	259	2	10	54	0	0	0	4	10	45	0	0	0

Here cfstat reveals 22 running requests total, but only 10 cfm and 10 cfc requests. The extra 2 requests were from accessing the ColdFusion Administrator which bypasses the new ColdFusion request limit pools.

To the Future

In July 2009 Adobe released a public Beta version of ColdFusion Server 9. The Request Tuning settings remains unchanged from as they were in ColdFusion 8 at this time, so this article still applies as of this writing.

Other Facets of Server Performance

Request Tuning is just one aspect of performance tuning considerations. Continue reading the Webapper blog for posts having overviews and recommendations of JVM tuning, and architectural suggestions.

Acknowledgements

Many thanks for the technical review provided by:

- **Andy Allan**
- **Mike Brunt**