Coldfusion Optimization and the Windows Legacy

Posted At : March 9, 2008 8:35 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Optimization

Coldfusion has been navigating the Internet waters in the good ship Java for some time now (6 or 7 years I think). If you are old enough in Internet years to remember Coldfusion 5 you probably know that Coldfusion was originally written for windows in C++ and ported over to Linux, Solaris and UNIX. These ports were not reputed to be particularly good and Coldfusion was largely considered to be a windows server application (and probably justifiably so). In 2002 with "CFMX" Coldfusion offered the Web world an application that was *not* just a windows application with a port to other platforms. Instead it was a *truly* cross-platform effort. Since that time Coldfusion has found it's way onto Linux in increasing numbers. In fact we are seeing more Coldfusion on Linux than ever before - particularly users who need Coldfusion Enterprise.

And why not? Coldfusion runs splendidly on Linux. We use CentOS at CF Webtools and it sings along happily with very few issues. In fact, it is possible to allocate about 50 percent more memory to the heap on a Linux machine (using the 32 bit JVM). That is a significant advantage that CF on Linux has over CF on windows. Here's a **blog post** by CF Webtools own Linux guru Ryan Stille on that topic. Meanwhile, here's a take on the server optimization from the good old days to today:

Coldfusion 4.x and 5.x Black Box Resources

One of the semi-icky things about CF 5 was that there were very few options for tuning it. Indeed it was programmed to latch onto as many resources as it needed to get the job done. In many cases this caused it to be a bit leaky and a good deal of the server problems that we had in the old days were related to CF's "black box" approach to handling system resources. Coldfusion was pretty much in charge of memory, threading and the like. The only way to influence it was to code carefully around the known nuances of the platform. Who can forget the constant locking that we had to implement for session and application variables - not to prevent race conditions (a challenge in any language), but to prevent server memory corruption. Lack of locking could actually crash the server (yikes!).

Starting with Coldfusion 6.1 and continuing with versions 7 and now 8 Coldfusion comes with a lot more internal options for fine tuning how it handles system resources. On a busy system these options can add dizzying complexity to the process of optimizing a server. One problem that people still have, however, is continuing to treat Coldfusion as if it were a "black box" application. They install it in the default configuration, they don't touch any of the settings, and they blame Coldfusion for being "un-scalable" when it starts to fail.

The Default Configuration

Here's Mark's best tip for optimizing your server. Never never never install it in the default configuration and expect good results. I'm not just talking about Coldfusion. This is true of *any* server. If you install Windows Server 2003 without hand picking the services you need - shame on you! A server is a special machine with specific purposes. It should not be tasked with *anything* that it doesn't explicitly need.

With regard to a Coldfusion server (versions 6-8) the default configuration is practically useless for anything more than a developer's workstation. For example, the

"simultaneous threads" setting is set to 8 by default. If you have a dual core dual proc machine you will want it to be more like 20 or even 28 (see my previous post on a **Case for Higher Simultaneous Requests**). And the heap settings are not just inadequate for a server, they are *woefully* inadequate. The default heap has no minimum amount and a maximum amount of 512 megabytes. Let's think about that for a minute.

A maximum of 512 megs means that if you install a beefy 4 gig machine that is intended to be a web server and nothing else - it will still *never use more than 600 or 700 megabytes of memory for Coldfusion* (512 plus some overhead like "permGenSize" etc). A typical server with the single task of web server should be configured to use the maximum amount of memory possible without impacting the rest of the server. And what about the minimum? No minimum means that the JVM is going to be tasked with "cleaning up" the heap to make it smaller all the time. In a typical optimized configuration the heap should be configured with little distance between maximum and minimum amounts.

Of course there are many other settings that need to be considered - Apache or IIS settings, specific changes to the jrun.xml file, specific switches for garbage collection etc. The main point is that you *should take the time to optimize your server after you have installed it.* Do not rely on the default configuration.

Adobe if You are Listening...

I realize that it is important to test settings. I know that every server is tasked with a different set of responsibilities and that one application that runs well with one set of options might suffer with another equally valid set of options. These are the reasons given for not providing some automated optimization assistance during the initial installation of Coldfusion. Let me just say that, while I agree with all of these points, I *do* think there is room for some helpful strategies during installation. For example, Adobe could add a "server tuning wizard" with all the standard caveats mentioned above. The wizard could ask about the servers resources, tasking and expected usage and then propose some "recommended" settings with a warning that experience may vary depending on usage. It could even leave it up to the sys admin to implement these settings. What do you think Muse readers? Is there room for some rudimentary help from Adobe in this regard? I'd be interested in hearing from some of my more informed readers on the subject.