Server stability: Of Diligence, Anecdote, and Fallacy

Posted At : November 24, 2004 7:07 AM | Posted By : Steven Erat
Related Categories: Java, Linux, Events, Quality Assurance, ColdFusion, Computer Technology, Macromedia

Today I continue my previous post and the CF Server Stability thread on CFTalk in reply to Micha's comments.

I reiterate; I don't believe that casting a broad net for specious reports, incomplete data, and hyperbole will serve to solve whatever particular difficulty ails a given server. I think most every case is rather unique, although where common threads are drawn they get turned around into Technote advisories, documentation, or possibly bug reports. I think each case needs to be founded in contextual information. In other words... Show me the data!

Customer applications really do vary widely, and we in Macromedia Support have the opportunity to see the full breadth of ingenious applications that emerge from the developer community. Realize that out of the box, the ColdFusion and underlying JRun server are tuned as best possible to meet the general needs of such a diverse audience, where their servers may run on any possible configuration within a broad spectrum of configurations limited only (but not always) by the published System Requirements. This encompasses the range of internet protocols, operating system quirks, other third party software integration. Further, it is assumed that there is some basic knowledge of best practices for tuning the CF and JRun servers, and it is expected that each application will undergo testing to determine the best server settings configuration for that specific app. Through Macromedia Devnet, Livedocs, and Technotes, Macromedia strives to help customer understand the performance options available.

Even so, it is expected that customers understand their applications and their context. Does a public shopping cart app need a SQL statement that is 10 to perhaps 50 lines long, composed of numerous WHEREs, LIKEs, INs, and subselects?

Let me illustrate through anecdote some common themes resulting from reports of server stablility or performance problems:

Mistake

Q: "I was reviewing your server settings, and I was wondering why general timeout wasn't enabled?"
R: "Oh, really? I thought I had set that. I know I set it appropriately on the other server"

Misunderstanding

Q: "Why is that your Simultaneous Request setting is set at 300?"
R: "Well when the server seemed to hang we thought it would help to just bump that up"

Not knowing the application

Q: "So I found that after you enabled the Log Slow Page setting that somepage.cfm is chronically reported to run 180-225 seconds. Can you account for what in that page is taking up the bulk of that time?"
R: "Oh, I really don't know. We inherited this application when the developer quit. How do we find that out?"

Unexpected

Q: "I've reviewed the thread dumps and found that all web threads are stuck on the same line of code a specific CFC"
R: "We load tested the application in development but didn't see this CPU spiking behavior. It's odd that under more load that this should happen, but at least we know where to look now"

Unsupported

Q: "So you're an ISP and your server is down and you have this HotSpot crash log. I found the crash was in this function call, and all Google hits for that function call point to Java on Debian... Oh, you *are* running on Debian, eh?"

ColdFusion/JRun bug

Q: "Your site is just fine under load all day, but at night it's completely idle until 4am when your other server spiders it... I found that the thread dumps contain absolutely no jrpp threads. That looks like a recent bug just discovered, and its good that you have a very reproducible case because we've been lacking that."

Sun bug

Q: "You found a HotSpot crash log with error code 4F530E43505002EF. Yes, we've found this on Sun's bug parade. Its a known issue in the JVM. You can turn off the HotSpot optimizer with -Xint, but that trades stability for performance"

My point here is that when you dig down into the server configuration, the application, and other raw data, we can most often attribute the problem to something very specific. Sometimes that turns out to be a bug, and that gets logged and helps improve the product, and most often a workaround or hotfix is provided. Sometimes it turns out to be configuration or code dependent.

Argumentum ad Misericordiam. Argumentum ad Populum. Aggregating all reports of server instability under one heading is not only futile, but fallacious. Be diligent in development, be thorough in staging, and be watchful in production. This will usually keep that $5000 consulting fee in your pocket, although a manicure may still be in order :)