Queued Requests Hanging Coldfusion? Check Network Capacity

Posted At : May 17, 2006 11:40 AM | Posted By : Mark Kruger
Related Categories: Hosting and Networking, Coldfusion Troubleshooting

Let me set the scene. A client's server was set up with about 20 sites. One site in particular was quite busy. After what has been described as a "spontaneous reboot" the server began have problems. It would stay up with all the sites enabled except for the one busy site. As soon as that site was enabled, running requests would climb slowly till they reached the simultaneous request threshold, then queued request would climb until the server was unresponsive.

NOTE: There is a **follow up** to this post.

I took the following steps:

- I perused the cfusionmx/logs and the runtime/logs looking for clues. All I discovered were lots of timed out requests and some broken sockets. Since these occur as the *result* of hanging threads (although they can also be a cause) I did not see a smoking gun yet.
- Since this busy site made heavy use of the database I suspected that an underperforming database was the culprit. I found a lack of indexing on 3 tables - none of them large. I ran a trace file and used the wizard to make recommendations. It gave me no recommendations.
- I examined the query code and added indexes based on experience - still no luck.
- I went through the usual JRUN tweaks (the proxy service changes, JVM settings etc) - still no success.
- I added process isolation in IIS - still no luck.
- I discovered a verity collection. Verity collections with badly optimized indexes can cause behavior like this when they are used heavily. I purged and re-indexed. This too resulted in no benefit.

At this point I was pulling out my hair (which always scares the dog). It was around 10:00 and I was discouraged. I knew that the simplest answer is usually the right answer. The simplest answer in this case was *database performance or connectivity*. I created a test query and ran it rapidly in sequence. It performed fine for 8 or 10 times in a row. I enabled the busy site and hit my test query again before requests began to queue. It was fine for 4 to 6 requests, then suddenly it bogged down - no response. I checked and the queued request were still at 0. The running requests had increased by 1. So I had a hanging thread... but why?

Maybe the db connection is just flaky. I opened a command prompt and started a ping:

```
C:\>;ping mydb.mydomain.com -t
```

Everything was at <1ms, but after about 12 or 15 pings I got a request timeout. I watched for about 5 minutes and I saw a request timeout appearing sporadically. Something was inhibiting the response from the database.

The Solution

I took a look at network utilization for the NIC on the web server. Although it was high - it wasn't so high as to cause me to think it was *past* capacity. It ran at about 65 percent. Because of the nature of Ethernet and TCP/IP this is actually a fairly high

number (duplexing notwithstanding). It meant the NIC card was *near* capacity. I logged into the DB Server via RDP and took a look at it's network utilization (no problem - 20 to 30 percent). I checked the link speed. These 2 boxes were connected to the same switch. They were both running at 10 Megabits. The provider in this case restricts each box to 10 megabits. So, both the db server, the good web server, and the bad (or perhaps just misunderstood) web server were all running 10 megabits.

Modern switches will often try to negotiate for a higher speed. Perhaps the port for the "bad" web server was set to auto-negotiate and it was periodically crowding the NIC with negotiation requests. This doesn't explain why RDP continued to run fine, but perhaps the DB packets were more fragmented or came in faster and were queued up in the buffer. In any case, network capacity seemed marginal for the amount of traffic (about 20 to 40 visitors a second) I set the NIC to auto-negotiate with some fear - I thought I might end up being permanently disconnected. The auto-negotiation spooled the NIC up to 100 megabits. This resulted (to my everlasting delight) in a 100 percent improvement. Requests were handled and no longer hung. The server began to perform perfectly.

Hanging Request Lessons

Because of the way that Coldfusion deals with timeouts the dreaded "hanging thread" is often related to Coldfusion's access to external services. That "timeout requests after *N* seconds" setting only applies to threads that Coldfusion can terminate. When it "hands off" a thread to a database driver it's like the day after a first date. It's really hoping the database server calls back. When it doesn't, Coldfusion just sits in it's room waiting by the phone - a poor lonely thread, hanging there (it makes me tear up). So anything with the potential to interrupt or stall that call-back process has the potential to hang your server - including low level networking issues.

In this case, I believe that something is wrong with the switch. I believe that the auto-negotiation process "solved" something in the configuration of this specific port. Otherwise, why did the "other" web server (equally busy) continue to operate fine under load… a mystery for another time perhaps. In any case, I now have one more thing to check on my check-list. Luckily, it's a simple thing that I can eliminate as a possibility within minutes.

Additional Notes

The "spontaneous reboot" was not recorded in the event logs. I suspect that this was a network outage - probably a reboot of the switch or router. When I see a "broken pipes" type message in the stack trace I will now have another thing to consider - that the DB is losing connectivity.