Windows Service Increasing CPU Consumption

Asked 16 years, 4 months ago Modified 13 years, 3 months ago

Viewed 7k times

7

At my job, I have a clutch of six Windows services that I am responsible for, written in C# 2003. Each of these services contain a timer that fires every minute or so, where the majority of their work happens.

My problem is that, as these services run, they start to consume more and more CPU time through each iteration of the loop, even if there is no meaningful work for them to do (ie, they're just idling, looking through the database for something to do). When they start up, each service uses an average of (about) 2-3% of 4 CPUs, which is fine. After 24 hours, each service will be consuming an entire processor for the duration of its loop's run.

Can anyone help? I'm at a loss as to what could be causing this. Our current solution is to restart the services once a day (they shut themselves down, then a script sees that they're offline and restarts them at about 3AM). But this is not a long term solution; my concern is that as the services get busier, restarting them once a day may not be sufficient... but as there's a significant startup penalty (they all use NHibernate for data access), as they

get busier, exactly what we *don't* want to be doing is restarting them more frequently.

@akmad: True, it is very difficult.

1. Yes, a service run in isolation will show the same symptom over time.
2. No, it doesn't. We've looked at that. This can happen at 10AM or 6PM or in the middle of the night. There's no consistency.
3. We do; and they are. The services are doing exactly what they should be, and nothing else.
4. Unfortunately, that requires foreknowledge of exactly when the services are going to be maxing out CPUs, which happens on an unpredictable schedule, and never very quickly... which makes things doubly difficult, because my boss will run and restart them when they start having problems without thinking of debug issues.
5. No, they're using a fairly consistent amount of RAM (approx. 60-80MB each, out of 4GB on the machine).

Good suggestions, but rest assured, we have tried all of the usual troubleshooting. What I'm hoping is that this is a .NET issue that someone might know about, that we can work on solving. My boss' solution (which I emphatically don't want to implement) is to put a field in the database which holds multiple times for the services to restart during the day, so that he can make the problem go away

and not think about it. I'm desperately seeking the cause of the real problem so that I can fix it, because that solution will become a disaster in about six months.

@Yaakov Ellis: They each have a different function. One reads records out of an Oracle database somewhere offsite; another one processes those records and transfers files belonging to those records over to our system; a third checks those files to make sure they're what we expect them to be; another is a maintenance service that constantly checks things like disk space (that we have enough) and polls other servers to make sure they're alive; one is running only to make sure all of these other ones are running and doing their jobs, monitors and reports errors, and restarts anything that's failed to keep the whole system going 24 hours a day.

So, if you're asking what I think you're asking, no, there isn't one common thing that all these services do (other than database access via NHibernate) that I can point to as a potential problem. Unfortunately, if that turns out to be the actual issue (which wouldn't surprise me greatly), the whole thing might be screwed -- and I'll end up rewriting all of them in simple SQL. I'm hoping it's a garbage collector problem or something easier to deal with than NHibernate.

@Joshdan: No secret. As I said, we've tried all the usual troubleshooting. Profiling was unhelpful: the profiler we use was unable to point to any code that was actually

executing when the CPU usage was high. These services were torn apart about a month ago looking for this problem. Every section of code was analyzed to attempt to figure out if our code was the issue; I'm not here asking because I haven't done my homework. Were this a simple case of the services doing more work than anticipated, that's something that would have been caught.

The problem here is that, most of the time, the services are not doing anything at all, yet still manage to consume 25% or more of four CPU cores: they're finding no work to do, and exiting their loop and waiting for the next iteration. This should, quite literally, take almost no CPU time at all.

Here's a example of behaviour we're seeing, on a service with no work to do for two days (in an unchanging environment). This was captured last week:

Day 1, 8AM: Avg. CPU usage approx 3%
Day 1, 6PM: Avg. CPU usage approx 8%
Day 2, 7AM: Avg. CPU usage approx 20%
Day 2, 11AM: Avg. CPU usage approx 30%

Having looked at all of the possible mundane reasons for this, I've asked this question here because I figured (rightly, as it turns out) that I'd get more innovative answers (like Ubiguchi's), or pointers to things I *hadn't* thought of (like Ian's suggestion).

> So does the CPU spike happen immediately preceding the timer callback, within the timer callback, or immediately following the timer callback?

You misunderstand. This is not a spike. If it were, there would be no problem; I can deal with spikes. But it's not... the CPU usage is going up generally. Even when the service is doing nothing, waiting for the next timer hit. When the service starts up, things are nice and calm, and the graph looks like what you'd expect... generally, 0% usage, with spikes to 10% as NHibernate hits the database or the service does some trivial amount of work. But this increases to an across-the-board 25% (more if I let it go too far) usage at all times while the process is running.

That made Ian's suggestion the logical silver bullet (NHibernate does *a lot* of stuff when you're not looking). Alas, I've implemented his solution, but it hasn't had an effect (I have no proof of this, but I actually think it's made things worse... average usage is *seeming* to go up much faster now). Note that stripping out the NHibernate "sections" (as you recommend) is not feasible, since that would strip out about 90% of the code in the service, which would let me rule out the timer as a problem (which I absolutely intend to try), but can't help me rule out NHibernate as the issue, because if NHibernate is causing this, then the dodgy fix that's implemented (see below) is just going to have to become The Way The

System Works; we are so dependent on NHibernate for this project that the PM simply won't accept that it's causing an unresolvable structural problem.

> I just noted a sense of desperation in the question -- that your problems would continue barring a small miracle

Don't mean for it to come off that way. At the moment, the services are being restarted daily (with an option to input any number of hours of the day for them to shutdown and restart), which patches the problem but cannot be a long-term solution once they go onto the production machine and start to become busy. The problems will not continue, whether I fix them or the PM maintains this constraint on them. Obviously, I would prefer to implement a real fix, but since the initial testing revealed no reason for this, and the services have already been extensively reviewed, the PM would rather just have them restart multiple times than spend any more time trying to fix them. That's entirely out of my control and makes the miracle you were talking about more important than it would otherwise be.

> That is extremely intriguing (insofar as you trust your profiler).

I don't. But then, these are Windows services written in .NET 1.1 running on a Windows 2000 machine, deployed by a dodgy Nant script, using an old version of

NHibernate for database access. There's little on that machine I would actually say I trust.

c# nhibernate windows-services .net-1.1

Share

Improve this question

Follow

edited Sep 15, 2011 at 15:44

Bill the Lizard

405k ● 211 ● 572 ● 889

asked Aug 25, 2008 at 14:45

TheSmurf

15.6k ● 3 ● 41 ● 48

Do you have any code that detects if the service is still "working" on a previous work cycle? – hova Aug 25, 2008 at 15:16

Is it possible for your service timer to be active more than necessary? Say, the timer is activated and the loop is not executed yet, and 2 minuets pass. Will the timer get activated again? A lot of things can go wrong but all of them are related specific to your code. This is not normal behavior of a service. – Jaywalker Nov 24, 2010 at 10:04

7 Answers

Sorted by: Highest score (default)

3

You mentioned that you're using NHibernate - are you closing your NHibernate sessions at appropriate points (such as the end of each iteration?)

If not, then the size of the object map loaded into memory will be gradually increasing over time, and each session flush will take increasingly more CPU time.

Share Improve this answer

Follow

answered Aug 26, 2008 at 12:06

Ian Nelson

58.6k ● 20 ● 78 ● 104

> I don't believe we are. That's somewhat of a pattern in our NHibernate-related development. Sessions lay around for a LONG time. That's an interesting possibility, thank you.
> – TheSmurf Aug 26, 2008 at 17:56

3

Here's where I'd start:

1. Get Process Explorer and show %Time in JIT, %Time in GC, CPU Cycles Delta, CPU Time, CPU %, and Threads.
2. You'll also want kernel and user time, and a couple of representative stack traces but I think you have to hit Properties to get snapshots.
3. Compare before and after shots.

A couple of thoughts on possibilities:

- excessive GC (% Time in GC going up. Also, Perfmon GC and CPU counters would correspond)
- excessive threads and associated context switches (# of threads going up)

- polling (stack traces are consistently caught in a single function)
- excessive kernel time (kernel times are high - Task Manager shows large kernel time numbers when CPU is high)
- exceptions (PE .NET tab Exceptions thrown is high and getting higher. There's also a Perfmon counter)
- virus/rootkit (OK, this is a last ditch scenario - but it is possible to construct a rootkit that hides from TaskManager. I'd suspect that you could then allocate your inevitable CPU usage to another process if you were cunning enough. Besides, if you've ruled out all of the above, I'm out of ideas right now)

Share Improve this answer

Follow

answered Aug 28, 2008 at 16:27

Mark Brackett

85.6k ● 17 ● 111 ● 155

2

It's obviously pretty difficult to remotely debug you're unknown application... but here are some things I'd look at:

1. What happens when you only run one of the services at a time? Do you still see the slow-down? This may indicate that there is some contention between the services.
2. Does the problem always occur around the same time, regardless of how long the service has been

running? This may indicate that something else (a backup, virus scan, etc) is causing the machine (or db) as a whole to slow down.

3. Do you have logging or some other mechanism to be sure that the service is only doing work as often as you think it should?
4. If you can see the performance degradation over a short time period, try running the service for a while and then attach a profiler to see exactly what is pegging the CPU.
5. You don't mention anything about memory usage. Do you have any of this information for the services? It's possible that your using up most of the RAM and causing the disk the trash, or some similar problem.

Best of luck!

Share Improve this answer

Follow

answered Aug 25, 2008 at 15:24

akmad

19.7k ● 2 ● 30 ● 25

2

I suggest to hack the problem into pieces.
First, find a way to reproduce the problem 100% of the times and quickly. Lower the timer so that the services fire up more frequently (for example, 10 times quicker than normal). If the problem arises 10 times quicker, then it's related to the number of iterations and not to real time or to real work done by the services). And you will be able to do the next steps quicker than once a day.

Second, comment out all the real work code, and let only the services, the timers and the synchronization mechanism. If the problem still shows up, than it will be in that part of the code. If it doesn't, then start adding back the code you commented out, one piece at a time. Eventually, you should find out what part of the code is causing the problem.

Share Improve this answer

Follow

answered Aug 25, 2008 at 16:40

Andrea Bertani
1,637 ● 12 ● 11

1

'Fraid this answer is only going to suggest some directions for you to look in, but having seen similar problems in .NET Windows Services I have a couple of thoughts you might find helpful.

My first suggestion is your services might have some bugs in either the way they handle memory, or perhaps in the way they handle unmanaged memory. The last time I tracked down a similar issue it turned out a 3rd party OSS libray we were using stored handles to unmanaged objects in static memory. The longer the service ran the more handles the service picked up which caused the process' CPU performance to nose-dive very quickly. The way to try and resolve this sort of issue to ensure your services store nothing in memory inbetween the timer invocations, although if your 3rd party libraries use static memory you might have to do something clever like create an app domain for the timer invocation and ditch

the app doamin (and its static memory) once processing is complete.

The other issue I've seen in similar circumstances was with the timer synchronization code being suspect, which in effect allowed more than one thread to be running the processing code at once. When we debugged the code we found the 1st thread was blocking the 2nd, and by the time the 2nd kicked off there was a 3rd being blocked. Over time the blocking was lasting longer and longer and the CPU usage was therefore heading to the top. The solution we used to fix the issue was to implement proper synchronization code so the timer only kicked off another thread if it wouldn't be blocked.

Hope this helps, but apologies up front if both my thoughts are red herrings.

Share Improve this answer

Follow

answered Aug 25, 2008 at 15:37

Ubiguchi

3,086 ● 1 ● 25 ● 24

> The threading thing isn't an issue; that's something I dealt with about 18 months ago, and I am confident that only one timer invocation is running at any given time. Good thought though, thank you. I will look into the AppDomain bit. Although I'm not aware of any memory issues, if I can just ditch the entire execution context after each timer hit, that may kill whatever issue we're having. Thanks for the suggestions. – TheSmurf Aug 25, 2008 at 15:41

1

Sounds like a threading issue with the timer. You might have one unit of work blocking another running on different worker threads, causing them to stack up every time the timer fires. Or you might have instances living and working longer than you expect.

I'd suggest refactoring out the timer. Replace it with a single thread that queues up work on the ThreadPool. You can Sleep() the thread to control how often it looks for new work. Make sure this is the only place where your code is multithreaded. All other objects should be instantiated as work is readied for processing and destroyed after that work is completed. STATE IS THE ENEMY in multithreaded code.

Another area where the design is lacking appears to be that you have multiple services that are polling resources to do something. I'd suggest unifying them under a single service. They might do seperate things, but they're working in unison; you're just using the filesystem, database, etc as a substitution for method calls. Also, 2003? I feel bad for you.

Share Improve this answer Follow

answered Aug 25, 2008 at 15:46

user1228

Thanks for that. Unfortunately, I do not have control over how many services are running. I totally agree that most of this should be consolidated. Unfortunately, the PM is convinced that running six services to do this work is more efficient. :S I've also been harping about upgrading to a new .NET

version for about two years. That hasn't had any effect. And now we have even more code (most of it ASP.NET), which will make the upgrade even harder now than when I first suggested it. I'm fairly confident that work is not stacking up. If a service is already working when its timer fires, th
– TheSmurf Aug 25, 2008 at 16:03

> Good suggestions, but rest assured, we have tried all of the usual troubleshooting. What I'm hoping is that this is a .NET issue that someone might know about, that we can work on solving.

My feeling is that no matter how bizarre the underlying cause, the usual troubleshooting steps are your best bet for locating the issue.

Since this is a performance issue, good measurements are invaluable. The overall process CPU usage is far too broad a measurement. *Where* is your service spending its time? You could use a profiler to measure this, or just log various section start and stops. If you aren't able to do even that, then use Andrea Bertani's suggestion -- isolate sections by removing others.

Once you've located the general area, then you can make even finer-grained measurements, until you sort out the source of the CPU usage. If it's not obvious how to fix it at that point, you at least have ammunition for a much more specific question.

If you have in fact already done all this usual troubleshooting, please do let us in on the secret.

Share Improve this answer

Follow

answered Aug 26, 2008 at 18:57

Joshdan

156 ● 1 ● 3