The case of the System process that consumed a lot of CPU

devblogs.microsoft.com/oldnewthing/20181112-00

November 12, 2018

Raymond Chen

A report came in through Feedback Hub that the System process was consuming high CPU. I was able to explain to the customer how to include a performance trace so the problem could be diagnosed.

In case you want to file a performance issue, the way to include a performance trace is to go to the *Additional details* section and click the *Recreate my problem* button. You will get additional options:

To help us understand what is causing the problem, please try to make it happen again while we follow along and capture data.
✓
Include data about Performance
✓
Include screenshots of each step
You will be able to review and edit the data before sending it.

⏱
Start capture

Check the *Include data about* box and select *Performance* as the category. Assuming the performance problem is ongoing, click *Start capture* and let it run for about 15 seconds, then click *Stop capture*. (If the performance problem occurs only when you perform a certain activity, then click *Start capture*, then do the activity that creates the performance issue, and then click *Stop capture*.)

That creates a performance trace that will be attached to your report.

Okay, let’s open the performance trace that this customer included. The tool for this is the Windows Performance Analyzer.

Since the problem is high CPU, the natural place to start is the Computation graph, which shows CPU usage.

Computation
(svg not supported on this site)

Yup, that sure looks like high CPU usage there.

Create an analysis page for that graph and zoom in to the period of high CPU. Here’s what’s using the CPU:

Line #	Process	Thread ID	Stack	Count	% Weight
1	▷ System			29,815	65.15
2	Idle	0	▷ [Idle]	510	21.35
3	▷ Taskmgr.exe (14412)				0.25
4	▷ MsMpEng.exe (5180)				0.07

And indeed most of it is going to the System process with 65.15%. The Idle thread is a distant second with 21.35%, and everybody else is noise.

So let’s dig into the System process.

Line #	Process	Thread ID	Stack	Count	% Weight
1	▼System (4)			29,815	65.15
2		9200	▷[Root]	1,605	3.51
3		19708	▷[Root]	1,576	3.44
4		▷18748		1,361	2.97
5		17480	▷[Root]	1,346	2.93
6		12132	▷[Root]	1,341	2.93
7		13020	▷[Root]	1,220	2.67
8		15064	▷[Root]	1,181	2.58
9		16364	▷[Root]	1,084	2.36
10		11376	▷[Root]	1,058	2.31
11		20444	▷[Root]	994	2.17
12		21000	▷[Root]	978	2.14

13		20648	▷[Root]		905	1.97
14		▷19076			895	1.95
15		8572	▷[Root]		757	1.65
16		13864	▷[Root]		743	1.62
17		17072	▷[Root]		685	1.50
18		16224	▷[Root]		653	1.43
19		▷15988			625	1.37
20		19592	▷[Root]		604	1.32
21		1784	▷[Root]		571	1.25
22		17872	▷[Root]		560	1.22
⋮		⋮	⋮		⋮	⋮

Hm, everything just flattens out. There's no big culprit sucking up all the CPU.

Are we being nibbled to death?

Let's look at two of those threads, maybe we'll discover something.

Line #	Process	Thread ID	Stack		Count
1	▼System (4)				29,815
2		9200	▼[Root]		1,605
3			▼ntoskrnl.exe!KxStartSystemThread		1,605
4			| ntoskrnl.exe!PspSystemThreadStartup		1,605
5			| ntoskrnl.exe!ExpWorkerThread		1,605
6			▷|- ntoskrnl.exe!IopProcessWorkItem		1,554
7			▷|- ntoskrnl.exe!KeRemovePriQueue		50
8			|- ntoskrnl.exe!ExpWorkerThread<itself>		1
9		19708	▼[Root]		1,576

10			▼|- ntoskrnl.exe!KxStartSystemThread		1,574
11			| ntoskrnl.exe!PspSystemThreadStartup		1,574
12			| ntoskrnl.exe!ExpWorkerThread		1,574
13			▷| |- ntoskrnl.exe!IopProcessWorkItem		1,538
14			▷| |- ntoskrnl.exe!KeRemovePriQueue		36

Okay, it seems that the threads are doing `IopProcessWorkItem`. That explains why the work is so evenly spread out: It's a thread pool.

Remove the *Thread ID* column because we don't care about which thread is doing the work. Now we can group purely by stacks.

Line #	Process	Stack		Count	% V
1	System (4)			29,815	
2		▼[Root]		29,810	
3		▼|- ntoskrnl.exe!KxStartSystemThread		29,794	
4		| ntoskrnl.exe!PspSystemThreadStartup		29,794	
5		| ntoskrnl.exe!ExpWorkerThread		29,699	
6		▼| |- ntoskrnl.exe!IopProcessWorkItem		28,742	
7		▼| |- contoso.sys!<PDB not found>		28,707	
8		▼| | |- contoso.sys!<PDB not found>		28,699	
9		▼| | | |- contoso.sys!<PDB not found>		28,588	
10		▷| | | | |- ntoskrnl.exe!RtlWriteRegistryValue		28,572	

Aha, basically all of the work items are going to the the Contoso driver, and that driver does very little work of its own. Of the 28,707 samples that showed that we were running a Contoso work item, 28,572 of them (over 99%) were in `RtlWriteRegistryValue`.

Basically, the Contoso driver was burning up all your CPU writing furiously to the registry.

The developers at Contoso replied that the customer was running a version of the driver that was over a year old. They suggested the customer upgrade to the latest driver and see if that fixes the problem.

I’m sure that upgrading to the latest driver will make the problem go away, but I’m not convinced that it’ll fix the problem. Because what’s probably happening is that the driver got into some sort of error state and is writing diagnostic information to the registry. That’ll go away even if you *don’t* upgrade the driver. All you have to do is reboot.

The real question is what sort of error state the driver managed to get itself into.

Raymond Chen

Follow