Diagnosing high CPU by studying profiling results, example

devblogs.microsoft.com/oldnewthing/20151113-00

November 13, 2015

Raymond Chen

A customer asked for assistance determining why their program demonstrated sporadic high CPU. It occurred only at a client location, so they couldn't set up a full debugging environment. They were able to convince the client to trace one of the spikes in a profiler. During this capture, high CPU was recorded for around 20% of the running time. Here is the drill-down of where most of that time was going.

Stack	% Weight
[Root]	19.36
|- ntdll.dll!RtlUserThreadStart	19.36
| kernel32.dll!BaseThreadInitThunk	19.36
| contoso.exe!__wmainCRTStartup	19.36
| contoso.exe!wWinMain	19.36
|⋮	⋮
| fabrikam.dll!Widget::~Widget	19.36
| fabrikam.dll!WidgetProxy::Disconnect	19.36
| fabrikam.dll!WidgetProxy::DisconnectAll	19.36
| fabrikam.dll!WidgetProxy::TransactCommand	19.36
| fabrikam.dll!WidgetProxy::WaitForResponse	19.36
| |- user32.dll!MsgWaitForMultipleObjectsEx	18.63
| | |- kernel32.dll!WaitForMultipleObjectsExImplementation	17.37
| | | |- KernelBase.dll!WaitForMultipleObjectsEx	16.55

| | | | |- ntdll.dll!ZwWaitForMultipleObjects	8.69
| | | | | |- ntoskrnl.exe!KiSystemServiceCopyEnd	4.76
| | | | | | |- ntoskrnl.exe!NtWaitForMultipleObjects	4.71
| | | | | | | |- ntoskrnl.exe!ObpWaitForMultipleObjects	4.02
| | | | | | | | |- ntoskrnl.exe!ObpWaitForMultipleObjects	4.02
| | | | | | | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | | | | | | |- ntoskrnl.exe!KiCheckForKernelApcDelivery	<0.01
| | | | | | | | |- ntoskrnl.exe!KiDpcInterrupt	<0.01
| | | | | | | | |- ntoskrnl.exe!KiInterruptDispatch	<0.01
| | | | | | | |- ntoskrnl.exe!NtWaitForMultipleObjects	0.49
| | | | | | | |- ntoskrnl.exe!memcpy	0.20
| | | | | | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | | | | | |- ntoskrnl.exe!KiDpcInterrupt	<0.01
| | | | | | |- ntoskrnl.exe!KiSystemServiceCopyEnd	0.05
| | | | | |- ntoskrnl.exe!KiSystemCall64	1.84
| | | | | |- ntdll.dll!ZwWaitForMultipleObjects	1.20
| | | | | |- ntoskrnl.exe!KiSystemServiceExit	0.56
| | | | | |- ntoskrnl.exe!KiSystemServiceRepeat	0.16
| | | | | |- ntoskrnl.exe!KiSystemServiceGdiTebAccess	0.13
| | | | | |- ntoskrnl.exe!KiSystemServiceStart	0.03
| | | | | |- ntoskrnl.exe!KiSystemServiceCopyStart	0.01
| | | | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | | |- KernelBase.dll!BaseSetLastNTError	6.57
| | | | | |- ntdll.dll!RtlNtStatusToDosError	6.33
| | | | | | |- ntdll.dll!RtlNtStatusToDosErrorNoTeb	6.05
| | | | | | | |- ntdll.dll!RtlNtStatusToDosErrorNoTeb	6.04

| | | | | | | |- ntoskrnl.exe!KiDpcInterrupt	<0.01
| | | | | | | |- ntoskrnl.exe!KiApcInterrupt	<0.01
| | | | | | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | | | | |- ntdll.dll!RtlNtStatusToDosError	0.29
| | | | | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | | | |- KernelBase.dll!BaseSetLastNTError	0.19
| | | | | |- ntdll.dll!RtlSetLastWin32Error	0.05
| | | | |- ntdll.dll!memcpy	0.15
| | | | |- KernelBase.dll!memcpy	0.01
| | | | |- ntoskrnl.exe!KiApcInterrupt	<0.01
| | | | |- ntoskrnl.exe!KiDpcInterrupt	<0.01
| | | |- kernel32.dll!WaitForMultipleObjectsExImplementation	0.60
| | | |- ntdll.dll!memcpy	0.16
| | | |- kernel32.dll!WaitForMultipleObjectsEx	0.04
| | | |- kernel32.dll!memcpy	0.02
| | | |- ntoskrnl.exe!KiInterruptDispatchNoLock	<0.01
| | | |- ntoskrnl.exe!KiApcInterrupt	<0.01
| | |- ntoskrnl.exe!KiDpcInterrupt	<0.01
| | |- ntoskrnl.exe!KiApcInterrupt	<0.01
| |- KernelBase.dll!GetTickCount	0.24
| |- KernelBase.dll!GetLastError	0.20
| |- fabrikam.dll!GetLastError	0.09
| |- kernel32.dll!GetLastError	0.02

From this chart, you can see that all of the time is consumed in `WidgetProxy::WaitForResponse`, and most of that time is in `MsgWaitForMultipleObjectsEx`. There is a lot of detail inside that function, so let's hide everything that contributes less than one percent.

Stack	% Weight
| fabrikam.dll!WidgetProxy::WaitForResponse	19.36
| |- user32.dll!MsgWaitForMultipleObjectsEx	18.63
| | |- kernel32.dll!WaitForMultipleObjectsExImplementation	17.37
| | | |- KernelBase.dll!WaitForMultipleObjectsEx	16.55
| | | | |- ntdll.dll!ZwWaitForMultipleObjects	8.69
| | | | | |- ntoskrnl.exe!KiSystemServiceCopyEnd	4.76
| | | | | | |- ntoskrnl.exe!NtWaitForMultipleObjects	4.71
| | | | | |- ntoskrnl.exe!KiSystemCall64	1.84
| | | | |- KernelBase.dll!BaseSetLastNTError	6.57

If we look only at the time spent in `WaitForMultipleObjectsEx`, 8.69%/16.55% = around half of the time is spent in `ZwWaitForMultipleObjects` (the kernel part of the function which does the waiting), and 6.57%/16.55% = around 40% of the time is the time setting the last error code.

That's odd. Why is this function spending nearly half of its time setting the last error code?

My theory: Because the call is failing with an error!

That explains the high CPU. The call to `WaitForMultipleObjectsEx` is failing, which means that instead of waiting, it returns immediately with an error code. The `Widget-Proxy::WaitForResponse` function doesn't quite know what to do in that case, so it shrugs its shoulders and tries waiting again. Eventually, whatever it's waiting for actually happens, and the call returns, but instead of waiting at low CPU, it accidentally created a CPU spin loop.

Raymond Chen

Follow