Should I be concerned that WaitForSingleObject is taking a large percentage of my performance test's execution time

devblogs.microsoft.com/oldnewthing/20160422-00

April 22, 2016

Raymond Chen

A customer designed a client-server system where the client and the server ran on the same machine (but with different security contexts). They were doing some performance tests of the data transfer portion of the system, and one of tests consisted of the following:

- Client opens a unit of content on the server.
- Repeat 100,000 times:
    - Client seeks to the start of the content.
    - Client reads entire content a line at a time until the end of the content is reached.
- Client closes the connection with the server.

When they ran this test, they found that the `WaitForSingleObject` function was consuming 6% of the total CPU time. "We expected overhead of calling the `WaitForSingleObject` function to be negligible. I suspect most developers don't take into account the full cost of `WaitForSingleObject`."

If you read the same content 100,000 times, it will quickly become fully cached, and the entire exercise becomes CPU-bound rather than I/O-bound.

CPU-bound operation is CPU-bound.

"Yes, we understand that this is an artificial scenario and that real-world applications will not use the server in this manner. The issue is that the `WaitForSingleObject` function is using a lot of CPU in this trace, whereas most developers probably consider `WaitForSingleObject` to be effectively free."

What you've done is taken what is presumably a complex operation and stripping it down to just overhead. Caching the entire workload in memory means that the actual work of reading the data is reduced to a series of `memcpy` operations, and since you're reading the data a line at a time, you're not copy a lot of data at each round trip. It's like using a large cardboard box to ship an index card. All of the cost is in getting the cardboard box, setting it up, packing it, sealing it, and mailing it.

Continuing the analogy: If you construct a sample test of your company's shipping system by shipping 100,000 cardboard boxes, each of which contains a single index card, then you're going to draw conclusions like "Tape and shipping labels constitute 5% of the weight!"

Well yeah, if you're shipping empty boxes, then tape and shipping labels are going to take up a measureable percentage of the weight, seeing as *you are shipping boxes that are practically empty*.

What you've got there, my friend, is a `WaitForSingleObject` stress test.

What is more interesting from a performance standpoint is not the percentage of overhead that goes to `WaitForSingleObject`. What you should be worrying about is the actual amount of time spent by `WaitForSingleObject`. From the customer's own data, it appeared that around 3 microseconds of the per-operation cost was being spent in calls to `WaitForSingleObject`.[1] That's the number you should be putting into your calculations to decide whether that overhead is preventing you from reaching your performance targets.

If you think about it some more, you may notice that the customer is worried that the `Wait-ForSingleObject` cost is too large a percentage of the CPU time, when in fact they should be worried that it is *too little* of the CPU time. Look at it another way: 94% of the CPU time is spent *inside the application code*. For example, when the `WaitForSingleObject` call returns, the server that was waiting on the handle has to figure out what the signaled handle means, determine which client is issuing the request and which piece of content is being requested, and route the request to the appropriate handler. The handler then performs any applicable security checks and parameter checks, and then it can do actual work: Calculate how many bytes of data need to be returned, locate those bytes, and copy those bytes. Finally, it has to do whatever is necessary to transfer that data back to the client.

Assuming that the actual work of determining what memory needs to be transferred to the client is, say, 1% of CPU (this is probably being generous), then that means 93% of the CPU is being spent in *application overhead*.

In the above analogy, the thing you should be noticing is not that tape and shipping labels take up 5% of the weight. What you should be noticing is that cardboard boxes take up 95% of the weight. That's the thing that's determining your shipping weight overhead. If you want to lower your weight overhead for shipping a single index card, don't try to get thinner shipping labels. Work on switching from cardboard boxes to envelopes.

[1] The customer didn't say how many times `WaitForSingleObject` was called per operation, so I don't know what the per-call `WaitForSingleObject` cost was.

Raymond Chen

Follow