When I try to calculate a performance counter manually, the answer is off by a factor of 100

devblogs.microsoft.com/oldnewthing/20160219-00

February 19, 2016

Raymond Chen

A customer was having difficulty calculating a performance counter manually. According to the formula in the documentation, performance counters that are of type `PERF_PRECISION_100NS_TIMER` should be calculated as $(N_1-N_0)/(D_1-D_0)$. But when we do that, the results we get are a factor of 100 smaller than the values reported by PerfMon. Are we expected to multiply the result by 100?

Let's take a step back and look at the various types of performance counters.

One general category is the counter that simply reports an instantaneous value. You ask for the amount of free memory, you get the amount of free memory at the moment you ask.

Another is a counter that accumulates a value over time. You ask for the number of bytes written to disk, and you get the number of bytes written to disk since the performance counter started keeping track. To extract something more useful like "bytes written to disk per second", you are expected to read the value, wait one second, then read it again, and then subtract. That gives the number of bytes written in the period of time that elapsed between the first and second reading.

If you want to report "bytes written per second" over a different period of time, you read the value, wait a little while, then read the value again; then you subtract the two readings and divide by the amount of time that elapsed between the two readings. Fancy math people might say that you are expected to *differentiate* over time, because what you are calculating is a derivative: $(f(a+h)-f(a))/h$.

For example, suppose you want to report "bytes written per second", but update the value ten times per second. You would read the value, wait 0.1s, then read the value again. Subtract the two readings to determine the number of bytes written during the 0.1s interval. To convert to this to bytes written per second, divide by 0.1s.

Via dimensional analysis: Bytes read ÷ seconds elapsed = bytes read per second.

A special case of the accumulated value is the temporal accumlated value. These are values that indicate how much time was spent performing some activity. If you subtract the two readings, then divide by the time between readings, you determine what fraction of the time was spent performing that activity.

For example, "CPU idle time" accumulates over time. To find out how idle the machine was in the past second, you read the idle time, wait one second, then read the idle time again. Subtract to determine the amount of time spent idle during the past second. And if you want to measure over a larger or smaller interval, you divide by the amount of time that elapsed between the readings.

Many of the accumulated values actually report two values: The accumulated value and a timestamp. The timestamp records the internal time at which the accumulated value was obtained. This is important for two reasons: For one thing, it takes time to read the counter, and you don't want that time to skew your calculations. But more important is that your code to wait exactly one second between readings can have latency or wobble, either because that is in the nature of the timer you chose, or because something happened outside your control, such as getting pre-empted by a hardware interrupt or taking a page fault. For high-speed counters, these delays can create large swings in calculated values. Using the timestamps reported by the counter ensures that your calculations are accurate.

Okay, so now the question: Why is the calculation off by a factor of 100?

The PerfMon program assumes that a differentiated accumulated time value represents a percentage. If you are differentiating a time value, then you are calculating the ratio of two time values, which is dimensionless. "During the 1-second interval that just ended, what fraction of the time was spent doing X?" This naturally lends itself to being expressed as a percentage rather than a direct ratio. It's easier to understand "X is happening 15.3% of the time" rather than "X is happening 0.153 of the time."

The customer explained that they were interested in the total amount rather than an average over time. In which case, they can take the reading at the end and the reading at the start, subtract the two values, and don't divide by the time that elapsed. That will tell you that the CPU was idle for *N* seconds between the two readings. I'm not sure what use that is to them without knowing how far apart the two readings are, but if that's what you want, that's how you get it.

Raymond Chen

Follow