How expensive is it to create a Windows performance counter?

devblogs.microsoft.com/oldnewthing/20230614-00

June 14, 2023

Raymond Chen

A customer had a program that created a lot of performance counters, although the program uses only a subset of them at any particular moment. How expensive is it to create a bunch of performance counters that aren't being used? Should they be careful to create them only when needed and promptly destroy them when finished? Or is it okay to create all of them at the start of the program and destroy all of them when the program ends?

Performance counters can be implemented in several different ways, and each way has a different cost.

One of the simplest performance counters is just a global 32-bit or 64-bit integer that tracks the number of times an event has occurred, or the total size of something. These types of counters are relatively inexpensive to keep up to date, since the code just needs to increment the value each time the thing occurs, or atomically add or subtract the size delta.

```
struct Widget
{
    Widget()
    {
        InterlockedIncrement(&TotalWidgetsCreated);
        InterlockedIncrement(&TotalActiveWidgets);
    }

    ~Widget()
    {
        InterlockedDecrement(&TotalActiveWidgets);
    }
};

HRESULT CreateWidget(Widget** result)
{
    *result = new(std::nothrow) Widget();
    return *result ? S_OK : E_OUTOFMEMORY;
}
```

These simple updates are so inexpensive that you may as well just do them unconditionally. The cost of of checking whether anybody is using them is comparable to the cost of updating the counter, so you may as well not bother checking.

On the other hand, other performance counters are more complicated to maintain.

For example, “Bytes sent over the network on behalf of application X” may have to do quite a bit of work to figure out which application to charge each network request to.

A counter that needs to be updated at each context switch would require a context switch hook to be installed to update the counters. And you don’t want to slow down context switches unnecessarily.

Some counters are backed by another data source, and starting a counter sets up a periodic timer to pull the values from the external data source and update the performance counter.

In other words, the answer is “It depends.”

Some performance counters are cheap. Others can be expensive. I don’t think there’s any requirement that each counter document how expensive they are, so if you want to play it safe, you should create counters only when needed and destroy them as soon as you’re finished.