What You Don't Know About a Meter:

Most people make a lot of assumptions when they are looking at performance data that can get them into trouble if they are not careful.

By: Bob Wescott

Summary

The more precisely you understand what a given meter is telling you, the more useful that data is. Here we look at some IO related data and focus on what we don't know to drive home the things you should know (and why you need to know them) before you begin working with any metering data.

This is an excerpt (with modifications) from: ***The Every Computer Performance Book*** a short, practical, and occasionally funny book I wrote on doing computer performance work.

What You Need to Know About Any Meter

For any meter you collect you need to know four things about it or it tells you almost nothing. Imagine you get a piece of data that states: **The application did 3000 writes.** Here are the four things you don't know about it and why you care.

1. The Time The Meter Was Taken

First and foremost, you need to know when the data was collected because no meter is an island. It is almost never the case that all possible metering data comes from one source. You'll have to dig into various sources and coordinate with other people. The way you link them together is time.

Since time is usually easy to collect and takes up very little space, I always try to record the time starting with the year and going down to the second. 2012-10-23 12:34:23. You may not need that precision for the current question, but someday you may need it to answer a different question.

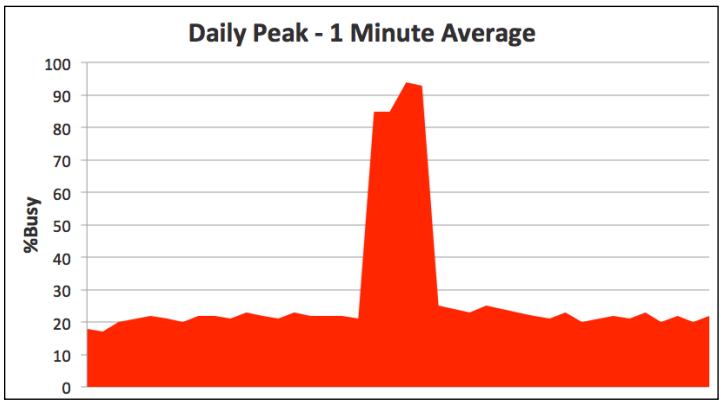
Adding the time can tell you: **The application reported 3000 writes at 3:05:10PM EST on June 19, 2012.** You can now compare and contrast this data with all other data sources.

2. The Sample Length of The Meter

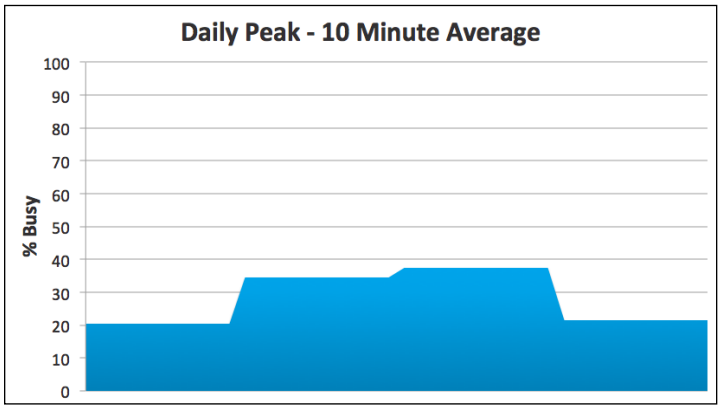
Any meter that gives you an averaged value has to average the results over a period of time. The most common averaged value is a utilization number.

The two graphs below show exactly the same data with the only difference being the

sample length of the meter. In the chart below the data was averaged every minute. Notice the very impressive spike in utilization in the middle of the graph. During this spike this resource had little left to give.



In the chart below the same data was averaged every 10-minutes. Notice that the spike almost disappears as the samples were taken at such times that part of the spike was averaged into different samples. Adjusting the sample length can dramatically change the story.



Some meters just report a count, and you've got to know when that count gets reset to zero or rolls over because the value is too big for the variable to hold. Some values start incrementing at system boot, some at process birth.

Some meters calculate the average periodically on their own schedule, and you just sample the current results when you ask for the data. For example, a key utilization meter is calculated once every 60 seconds and, no matter what is going on, the system reports exactly the same utilization figure for the entire 60 seconds. This may sound like a picky detail to you now, but when you need to understand what's happening in the first 30 seconds of market open, these little details matter.

Now, adding the sample length to what we already know about this meter, can tell you: **The application reported 3000 writes between 3:00-3:05:10PM EST on June 19, 2012.**

3. What Exactly Is Being Metered

As the old saying goes: "When you assume, you make an ass out of u and me." Here we have two undefined terms: application and writes.

An "application" is usually many processes that can be spread over many computers. So we need a little more precision here. Where did those 3000 writes come from? Just one process? All processes on a given system? All processes on all systems?

"Writes" can be measured in bytes, file records, database updates, disk blocks, etc. Some of these have much bigger performance impacts than others.

Even within a given metering tool, it is common to see the same word mean several different things in different places. Consistency is not a strong point in humans. So don't assume. Ask, investigate, test and double check until you know what these labels mean. The more precisely you understand what the meter measures, the more cool things you can do with it.

Now, adding the specifics about what is being metered can tell you: **All application processes on computer X reported 3000 blocks written to disk Y between 3:00-3:05:10PM EST on June 19, 2012.**

4. The Units Used in The Meter

Lastly, pay attention to units. When working with data from multiple sources it is really easy to confuse the units of speed (milliseconds, microseconds), size (bits vs. bytes), and throughput (things per second or minute) and end up with garbage. It is best to try to standardize your units and use the same ones in all calculations.

Since 5 minutes is 300 seconds, we can calculate the application was doing 10 writes/second = 3000 writes / 300 seconds.

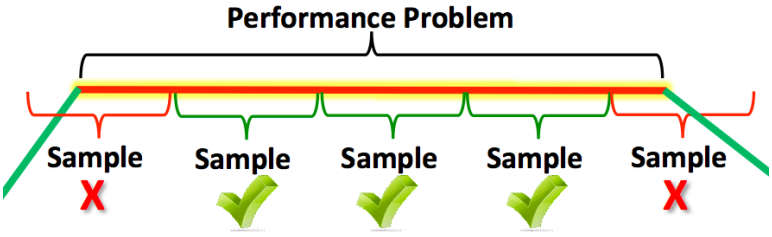
So finally we can tell you that: **All application processes on computer X wrote an average of 10 blocks per second to disk Y between 3:00-3:05:10PM EST on June 19, 2012.**

Extra Credit: Metering At The Right Frequency

You often have choices to make about how frequently to collect metering information. There is no "right" answer to this because it depends on what you are doing.

The more frequently you meter, the higher the resolution is on your performance data (e.g. more pixels in the image) and the higher the cost of metering. For internal meters, the "cost" might be just a few CPU cycles and a little disk space. For external third party testing, the "cost" might be real money. Know the cost of your metering and find a good balance between cost and getting the data you need.

My rule of thumb is you need to meter at a frequency that will get you *at least* two or three good samples during the performance problem you are trying to study. The diagram below shows why it is always important to record the time of the sample. The samples at the "edge" of the problem you are studying are not as helpful as the ones that were taken during the "core" of the problem.



In conclusion

Now we really know something about what's going on, when it happened, and have data specified in a common unit we can compare and contrast with other metering data. With this information we can ask better questions and understand how work is flowing though this part of the system.

Here is a brief example illustrating how important correct use of units can be. I know of one company that had to give away about Five Million Dollars worth of hardware on a fixed price bid due to a single metering mistake by the technical sales team where kilobytes were confused with megabytes. Ouch.

This short, occasionally funny, book covers Performance Monitoring, Capacity Planning, Load Testing, and Modeling.

It works for any application running on any collection of computers you have. It teaches you how to discover more about your meters than the documentation reveals. It only requires the simplest math on your part, yet it allows you to easily use fairly advanced techniques. It is relentlessly practical, buzzword free, and written in a conversational style.

Paperback: http://amzn.com/1482657759

On the iPad: https://itunes.apple.com/us/book/id607999070

Book's Website: http://www.treewhimsy.com/TECPB/Book.html