The Road to Hell is Paved with Average

The Road to Actionable Intelligence is Paved with Minimum, Average, 95th Percentile and Maximum

Ben Davies, Movìri Inc

In the next few pages we will discuss why on average, average is a below average metric. Then discuss that average becomes a better metric at smaller time slices.Then show an example where even at 15 second time slices, average failed to show actionable intelligence.

I find that significant actionable intelligence is missed with our obsession with average. It is not that average is not a good metric, but if that is all you use then you are paving the road to (capacity) hell. This is a case to encourage you to include maximum, 95th percentile, average, and minimum in your capacity analysis arsenal.

Average is usually an acceptable metric, but loses its usefulness as the 'time slices' become thicker, as compared to the monitoring target. Average of a day of hourly data is characterizing 24 discrete events to 1 number. That may be fine for hourly counts, but most capacity metrics, are events that are measured is in tens or hundreds of milliseconds. An average of hourly connections at five per second, is characterizing 18,000 discrete events to 1 number. Higher connection rates drives the discrete event count up quickly.

Here is a case in point. We got an alert from a monitoring tool for low idle workers. Let's be clear what the metric is showing. Chart 1 When 'nothing' is happening on the device, the idle workers are many (up on the chart), however as the system gets busier the idle workers are doing work, so there are fewer available idle workers to do new work (lower on the chart). When idle workers get to zero, there are no idle workers to service new work, and you are effectively down, that is, unable to process new work, until workers become idle again.

Looking at the chart, everything looks "fine" and the low idle worker alert was ignored. Hundreds of alerts over months were ignored with similar chart reviews.

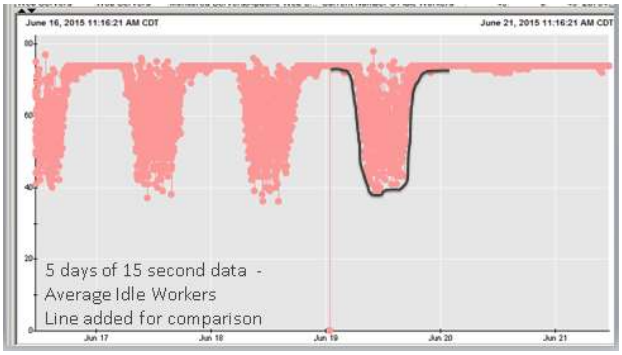

Chart 1 – Minimum idle workers 15 second slice

Turning on Minimum and Maximum shows a completely different story. This chart is much more interesting and **IS actionable intelligence**. Chart 2

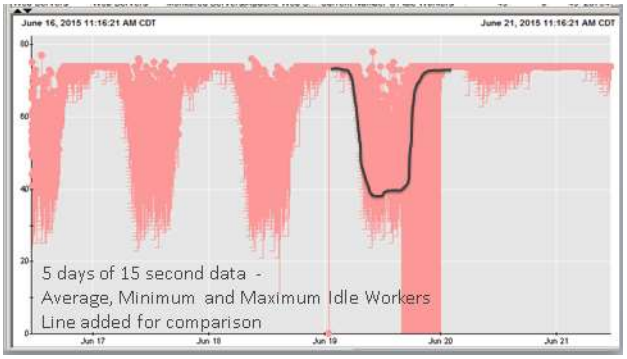

Chart 2 – Min idle workers with min and max shown

This chart clearly shows that for some period during each 15 second time slice, there were zero idle workers, that is, no workers available to service new work, even though the average over the 15 seconds were normal. And it was in this condition for HOURS.

The result of this is highly dependent on the application. When there are no workers to connect to, does the process gracefully fail and try again after some random

period, or does it fail not gracefully with an error presented to the end user? Each application will be different. In our case, errors were dumped into a log (that was unmonitored) and the user saw connection delays and poor performance. The monitoring system generated errors, but we have seen these were discounted out of hand.

The root failure of the initial analysis is the 'love of average'. "Average is all we look at. Especially," it was stated "in a 15 second slice, minimum and maximum would be little different. So there is no use to look."

Rubbish I say.

The analysis rule of thumb should be to use the best metric, at the best resolution, of the best type (min, average, max, etc). And you will not be able to determine this without looking at the options. Especially during an abnormal condition.

In the above example, we created a 'health check' chart based on hourly average compared with hourly minimum. Chart 3

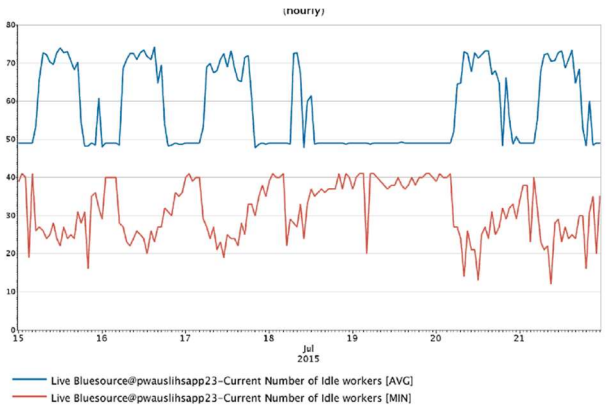

Chart 3 – health check chart

This chart makes it clear that while the average was 'normal' over the hour duration, at no time during the hour was the minimum lower than about 15 idle workers.

Average, on average, is but an average metric. It must be compared to another metric to confirm its validity. I wonder how many of our chronic issues have remained unresolved due to our obsession with average. And our unwillingness to confirm average is approperate.

My favorite metric for finding actionable intelligence is min and max. When the extremes are, well extreme, then the system is in an exceptional state. If it performs in an exceptional state successfully, then it is much more likely to work well in its average state. However, min and max, especially max is susceptible to exceptional exceptions. So, using a 95th Percentile type metric is generally better to determine compliance to a standard.

What we mean by this is your nonfunctional requirements should be state like this: "95 percent of all port 80 traffic to the xxx service must have a response time of 3,450 milliseconds or less during normal operation periods." This lends itself to the 95th percentile metric and give a threshold that is measured in milliseconds. If it were measure in tenths of a second or in seconds the implied tolerances around the threshold are changed. Violation by a 0.5 of a second is viewed differently than a violation of 500 milliseconds. Except the violation is the same in either case.

So, if all we use normally is average, maybe it is because we forgot what the options are. Typically, there are: minimum, mean, median, average, maximum, and standard deviation. There are very specific definitions for each word and its use as a mathematics term or statistical term. The data set will help determine which is the most appropriate for an analysis exercise. And my point here is that you should use several rather than blindly using average. Or at least confirm average is the most appropriate metric.

It has been assumed that we were talking about a resource utilization as a percentage, but there are others. Counts per period, degrees (heat), location coordinates, or other measures (distance, volume, mass).

Average over a minute is a fine metric. Average over five minutes is adequate. Average over fifteen minutes is only sometimes useful, but average over an hour is the default for long term analysis and misses significant opportunity to discover actionable intelligence.

Frame average in a different way.

Average over an hour only delivers 8 to 10 data points a business day. Compared with average work done per hour may give an average relationship but missed nuance in a system with 200 millisecond transaction times. Or phrased another way, a one hour average summarizes 18,000 events per hour ((One 200 millisecond transaction) * 5 in a second *60 seconds in a minute *60 minutes in a hour). Having more than one transaction at a time just makes this number bigger. Is 18,000 events averaged to a single point really the metric you wish to base your career on?

Be Careful on Min and Max

Many systems take an average of say five minutes and give that to the monitoring system. These metrics are given to

the reporting engine which allows you to show min and max. However, that is the minimum five-minute average and the maximum five-minute average and is NOT the min and max you may have expected. You can prove this my plotting min, average, and max then scaling back the time slices. If the 'detail level' have the lines overlay, you are reporting the minimum average, and maximum average.

Depending on the analysis this may be an acceptable situation. If not, look at having the monitoring system report the five-minute average in addition to the five-minute minimum and five-minute maximum. But what about the additional storage requirements. Actually, the storage impact can be minimized as the monitoring system or reporting engine will (can) quickly age out the five- minute data and consolidate the min, average, and max at the higher data slices. Giving a true min, true average, and true maximum.

Summary

The idea here has been that there are many more metric choices than average. And while average is a work horse and generically useful, it is over used. You are encouraged to regularly use other metrics to at least confirm average is appropriate for a given analysis. There are more effective measures to assists with the quest to find actionable intelligence. You should go out of your way to use them.

Specifically, during troubleshooting, average is likely to mask actionable intelligence. When systems are in crisis, maximum and minimum are more likely to point to a problem than average.

More detail on the metric in our example.

The metric is a JVM metric called minimum idle workers. And was monitored by CA Wily Interscope. The metric is literally the number of configured 'workers' sitting about waiting for a connection. As connections arrive they change states to busy workers until the connection is terminated where they go back to idle workers.

In the example, on average these behaved in a completely normal way even when there was a problem. What the technicians did during the troubleshooting was to restart the JVMs, which resolved the problem. The conventional wisdom was that the JVMs are misbehaving in some way that no one could figure out, but once it was found that resetting the JVMs resolved the issues investigation stopped.

When we pursued the minimum idle worker alerts, we interrogated the monitoring data and 'found' the show minimum and maximum check box in Wily. This revealed the other chart. Once this was observed we told everyone about it and got very hostile responses, from all but one important application owner. They used the data to determine which device was 'freaking out' and causing the burst of connections that 'flooded' the idle workers but did not get intended work done. A code fix or two mitigated the issue and the application went from three a week 'all hands on deck calls' about poor performance, to zero 'all hands' calls for a year. Simply by having more complete information that pointed them to the real problem. Phrased another way, by not relying on average, and looking at other metrics, the problem was seen in a way that helped resolve the underlying issue.

For the record, the JVMs had nothing to do with the actual problems. They were victims of the upstream device flooding them with bogus connections or the downstream not releasing connections appropriately. Resetting the JVMs forced resets of these bogus connections both upstream and downstream which seemed to clear the problem, at least for the moment. As they were where the corrective action seemed to happen, they were wrongly blamed as being the problem.