Resolving flapping detectors in Splunk InfraMon

Because detectors that use static thresholds are easy to create and easy to understand, you have configured them to alert immediately every time a signal crosses a threshold. This provides the benefit of timeliness: if elevated values of the signal reveal problems to come, you are informed as soon as possible. However, even though you choose relevant signals and decent thresholds, the elevated value sometimes reveals only a transient stress on the system, and you now have alerts that fire and clear repeatedly in a short period of time. This phenomenon is known as “flapping” and you need to resolve it.

Configure a duration for the alert

For example, instead of triggering an alert immediately when CPU utilization exceeds 70%, you could require that utilization be above 70% for 2 minutes.

Configure a percent of duration for the alert

For example, you could require that CPU utilization be above 70%, for 80% of a 5 minute period. Be aware that using percent of duration conditions requires that you know the resolution of your detector (normally equivalent to the resolution of your data) to be able to accurately interpret its behavior. The reason for this is that the denominator for calculating the % of duration is the number of datapoints expected, and not the number of datapoints actually received. For example, let’s say that you are sending CPU utilization every 10 seconds. This means that in a 5 minute window you should have 5 x 6 = 30 measurements of CPU utilization, and that 80% of the 5 minute window means that 24 values must be above 70% for your detector to fire. However, if only 20 out of 30 expected datapoints arrive, and even if 16 of them (80%) are above 70%, then the detector will not fire. The rationale for this behavior is a combination of the fact that the detector (a) expects periodic data (in this case, every 10 seconds), and (b) makes no assumptions about the possible “goodness” or “badness” of missing data. As such, missing data can leave the detector in an ambiguous state, and the detector defaults to retaining its previous state rather than raising or clearing alerts based on assumptions. One important consequence of this behavior is that % of duration conditions cannot always be used for aperiodic data.

Configure the severity and notification for the alert

For example, you can have a minor alert (such as above 70% for 2 minutes) send a message to a Slack channel, while a critical alert (such as above 90% for 5 minutes) pages the on-call engineer.

Apply a smoothing transformation

Smoothing transformations allow you to obtain a signal with less fluctuation. They reduce the impact of a single (possibly spurious) extreme observation. One such transformation is the rolling mean, which replaces the original signal with the
mean (average) of its last several values, where “several” is a parameter (the “window length”) that you can specify. Averaging the last few sample values is a method of approximating the true signal.

Establish a clearing condition

By default, an alert clears when the original trigger condition is no longer true. For example, if your alert is triggered immediately by CPU utilization going above 70%, then it will clear when CPU utilization goes below 70% (again, immediately). This kind of clearing condition works well if your signal does not encounter the threshold frequently or in quick succession. However, if your signal is hovering in the vicinity of the threshold, then the reciprocal clearing contributes directly to flappiness and detectors that are noisy and not particularly helpful.

Next steps

These additional Splunk resources might help you understand and implement these recommendations:

- Blog: A deep dive into Splunk APM alerts
- Splunk Docs: Detectors and alerts
- Splunk Docs: Transformations and aggregations
- Tool: Alert assessor