New Relic®

Improve Alerting Accuracy

New Relic's Apdex-Driven Approach Honed by Big Data

 | 188 Spear Street, Suite 1200, San Francisco, CA 94105 | www.newrelic.com

Table of Contents

Overview

Many people rely on Application Performance Monitoring (APM) tools to alert on performance degradation based on the end-user experience and application components data. But the challenge with alerting is that it needs to be accurate, meaning alert on true performance degradation with tangible user impact and business impact.

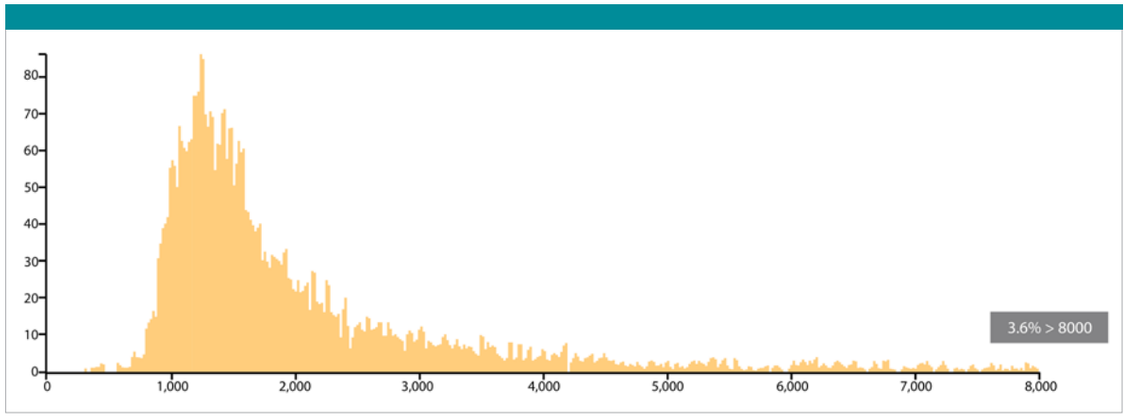
If the alert threshold is too low, it results in too many false alerts, causing operational folks to lose confidence in the alerting system. On the other hand, if the alert threshold is too high, IT operations could miss relevant performance degradation, resulting in significant user and business impact.

New Relic uses Apdex, an industry standard for measuring users' satisfaction with the response time of an application, to solve the complex alerting problem. This approach only continues to evolve and refine as we service and collect data from our customers' more than three million applications. In the following pages, you'll learn:

- How patterns in web performance data impact alerting
- What problems exist in today's alerting methodologies
- How New Relic's approach solves these challenges

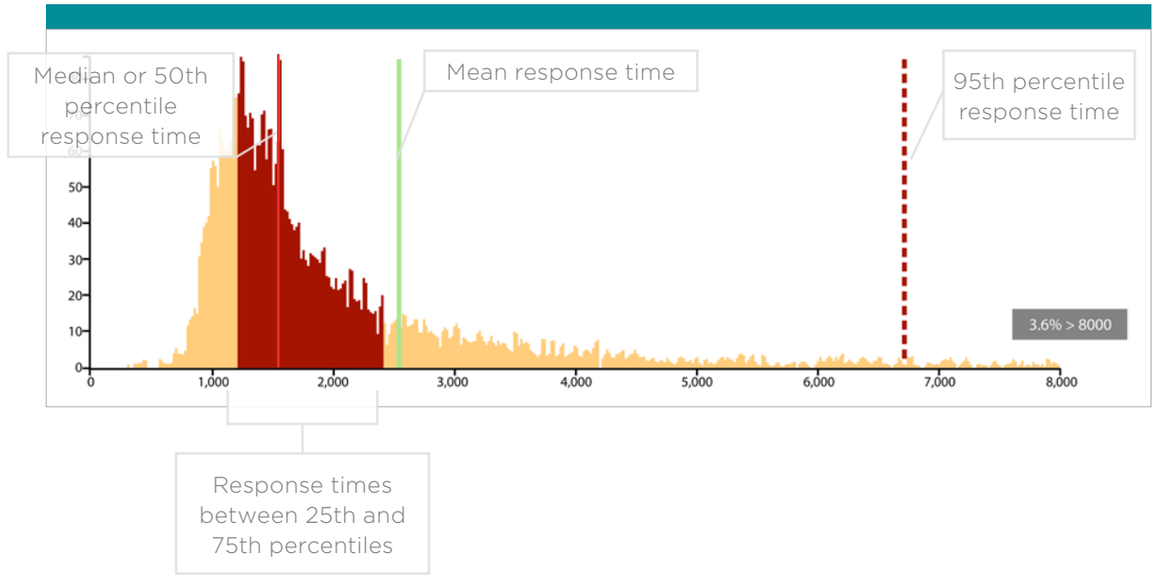
Understanding Web Performance

Based on the data that New Relic collects, we see a consistent performance profile of web traffic response times as shown in the figure below.



This is called an Erlang distribution that can be approximated by a log normal distribution. This pattern is pretty consistent across varying intervals of time—be it two-minute, hourly, daily, weekly, or annual time periods. This profile is also consistent for a specific user transaction, such as a Search function or "Add to Shopping Cart" within an application. The reason for this distribution is that these systems nearly always fit the model of a queuing system with arrivals as a Poisson process.

Now let's take a look at that same histogram with markers to indicate response time.



Given this understanding, let's take a closer look at the various approaches that APM vendors have taken in the past.

Static alert threshold based on mean

The most simplistic approach is to set the threshold to a static value and alert on every transaction instance that breaches the threshold or a specific number of breaches in a given interval of time. Setting the threshold based on mean would definitely cause an increase in false alerts. Hence most users of traditional APM solutions do not use alerts actively to perform their daily jobs, or they set the thresholds way too high to prevent false alerts. As you can see from the graphic on the previous page, the mean is close to the 75 percentile, implying that a large percentage of transactions will breach the threshold based on the mean.

Simple rolling averages over a period of time

Using rolling averages over a period of time to drive alerting is an improvement on the previous approach, as it increases the weight of most recent performance in computing the average response time. The fundamental nature of web traffic doesn't vary between these intervals and the profile of performance within each timeframe remains consistent and follows the same pattern—the Erlang distribution. However, almost 25 percent of the response times will be more than mean for any time-period within which the mean is calculated. Alert noise and eventual fatigue will result if the mean is used as a threshold, which is why most customers set the thresholds much higher or ignore the alerts altogether. The other downside of this approach is that the system fails to alert for transactions that gradually get slower and slower, because the thresholds progressively get higher and do not reflect the user expectations of performance.

Dynamic baselines of response time based on averages

Some APM vendors use the approach of computing a dynamic threshold based on hour of day and/or day of week or month. In this approach, the APM tool computes a different threshold for the average response times for that specific hour and the average is set as a threshold for that representative period. Since this approach also uses average to compute the threshold, it ends up giving more weight to the outliers and setting the threshold higher than what a typical user would expect performance to be at. Again, this goes back to the characteristic of the performance profile being an Erlang distribution.

While this is an improvement in reducing false alerts, it misses the fundamental point of APM. These dynamic baselines mask predictable application performance inconsistencies as normal. Is a consistent two-second response time during a Monday morning equivalent to a 10-second response time on a Friday evening? If this trend is consistent across multiple weeks for a specific application, the dynamic baseline marks this behavior as "Normal" and will not alert you.

A Better Approach to Alerting Driven by Apdex

New Relic adopted Apdex since its early days to accurately alert on poor performance without missing degradations and causing false alerts. We have refined this approach over the last several years by working closely with more disruptive, emerging companies who by definition have fewer resources and cannot waste time and critical resources on false alerting. By looking at big data streams across our APM product portfolio, we have developed unique, highly differentiated best practices on setting the right values for Apdex configuration. Before we dive into the specifics, let's first get a better understanding of the Apdex measurement.

What is Apdex?

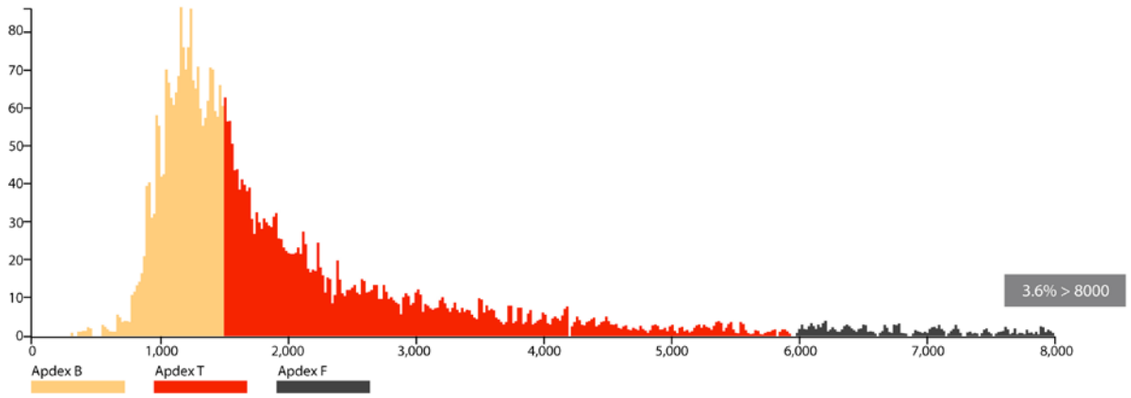
The Apdex method converts many measurements into one number on a uniform scale of 0 to 1 (0 = no users satisfied, 1 = all users satisfied). The resulting Apdex score is a numerical measure of user satisfaction with the performance of enterprise applications. This metric can be used to report on any source of end-user performance measurements for which a performance objective has been defined.

The Apdex formula is represented as the number of satisfied samples, plus half of the tolerating samples, plus none of the frustrated samples, divided by all the samples:

Apdex T = (Satisfied Count + Tolerating Count / 2) / Total Samples

In the above equation, the subscript T is the target time, and the tolerable time is assumed to be four times the target time. So it's easy to see how this ratio is always directly related to users' perceptions of satisfactory application responsiveness.

Apdex is like a histogram with just three buckets: Satisfied, Tolerating, and Frustrated. The buckets represent requests that are satisfying the user, requests that users just tolerate, and requests that fail to meet the users' expectations completely. The bucket intervals are 0, T, and 4T, where T is a parameter you choose in advance. Here's what that looks like if you shade the Apdex buckets in our histogram:



Key Benefits of Apdex

Why do companies rely on Apdex to know how their software is doing?

- Apdex overcomes the fundamental flaw of using average to set thresholds for performance that follow an Erlang distribution (which can be approximated to a log normal distribution).
- Apdex provides one uniform measure across transactions and applications. This helps IT and business stakeholders easily know if their customers are satisfied or not by eliminating the need to know what a good response time is for each transaction.
- The simplicity of Apdex leads to proactive action and work streams to improve performance of specific transactions and applications.
- Apdex eliminates alerting noise and saves IT Ops considerable time by not having to chase random outliers.
- Apdex provides an easy way to set and manage user experience with the ability to set different T values for different key transactions.

DEBUNKING MYTHS ABOUT APDEX

1. Setting up Apdex is time consuming and complicated.
It only takes minutes to set up Apdex, and the metric is based on your view of user expectations. Contrast that with the inordinate amounts of time developers and infrastructure teams spend building these type of capabilities in house, and it's really a walk in the park.

2. Apdex can detect business impact only if more than 5 percent of transactions are slow.
This is absolutely false and reflects the views of folks with limited understanding of New Relic's implementation of Apdex. New Relic computes Apdex every two minutes irrespective of the number of users in the system. That means that Apdex is comparing the performance profiles across two-minute intervals, which is independent of a specific percentage of transactions being slow.

New Relic also offers the flexibility to set different response time thresholds to configure the Apdex score. For instance, the T value for a key transaction such as "Browse a product category" or "Search for a specific product" might be set to one second, but a report that pulls all users that bought a certain product today might be a less frequent data-intensive transaction for which the appropriate T value might be seven seconds.

3. Apdex-based alerting causes false alarms due to static thresholds.
User expectations for a specific kind of transaction are very consistent (static), irrespective of the time of day or number of other users on the systems (holiday season). Now, given that the user expectation is consistently high and not dynamic based on extraneous factors, how do we prevent false alerts?

- Because random low static thresholds lead to a large number of alerts generated, New Relic sets different thresholds for different transactions within the same application (as described in previous pages).
- To prevent sudden spikes in alerts during consistent performance degradation or during an outage, New Relic aggregates alerts into incidents and does not inundate the user's inbox with more alerts on the same issue.

Getting Started with New Relic Alerting

New Relic has monitored millions of applications across various industries since 2008. During that time, we've developed the following best practices for setting up Apdex:

1. Install the New Relic agent and start collecting data.
2. Identify the transactions that are most important to your business. Most companies typically have between five and 10 key transactions.
3. Create your Key Transactions in New Relic. By default, New Relic APM will set your T value to 500ms.
4. After a week of collecting data, look at the New Relic histogram view for each of the Key Transactions, see where 85 percent of your customers are falling, and change the value of T based on this data.

The Apdex measure is calculated for each of the Key Transactions every two minutes for the previous five-minute interval. And New Relic also does the same for the entire application. After configuring T for your application, you are all set to benefit from the real-time scoring of every user transaction against the true expectations of your users, and not a random algorithm that computes the dynamic expectation of your user community.

For additional resources and more information about using Apdex and New Relic's alerting policies, visit docs.newrelic.com.