Hierarchical Anomaly Detection in Load Testing with StormRunner Load

A fresh approach to cloud-based website load testing is proving more effective in identifying and isolating application performance anomalies.
Table of Contents

Hierarchical Anomaly Detection in Load Testing ........................................ 1
Limitations of Traditional Anomaly Detection and Isolation ....................... 1
Solving the Anomaly Challenge: Hierarchical Anomaly Detection .............. 2
Explore Hierarchical Anomaly Detection in StormRunner Load ............... 7
Hierarchical Anomaly Detection in Load Testing

Successful application load testing requires identifying conditions where an application is unable to maintain a consistent performance profile—so that issues can be addressed, improvements can be made, and service-level agreements (SLAs) can be met. However, current anomaly detection methods are problematic on several fronts, adding complexity and delay to the website load testing process and, in some cases, leading to inaccurate or misleading test results. Micro Focus® has been researching new techniques of automatic, hierarchical anomaly detection and isolation that enable users to quickly, intuitively, and effectively analyze load results in real time. A new multi-layered approach, available in the cloud-based StormRunner Load testing solution, is able to identify and alert users to abnormal application behavior, and provides the insight needed to determine the root cause of detected deviations. This paper examines the growing need for a more efficient and effective anomaly detection technique, describes the novel approach developed at Micro Focus, and provides an example that illustrates the value of the hierarchical approach given today’s era of ever-increasing application complexity and shorter cycle times.

Limitations of Traditional Anomaly Detection and Isolation

Anomaly detection is the ability to identify items or events that are not behaving in an ordinary or expected way. Many traditional anomaly detection techniques are based on unsupervised machine learning and/or statistical techniques—detecting anomalies in time series data with numerical values that are uniformly spaced in time. Anomaly detection capabilities are widely available in application performance management (APM) tools; typically they notify the IT operations practitioner of any critical abnormal behavior in production systems. The events monitored include a long list of system metrics such as CPU utilization, available memory, disk I/O, as well as application metrics such as response time and availability. Similarly, performance-validation tools help to identify application problems, although they do so while synthetic load is being generated in a closed and controlled environment as opposed to the real user loads under test in the APM arena.

The traditional approach to identifying abnormal application behavior is to define manually what is normal and what is abnormal by setting a threshold (e.g., CPU utilization > 85% is abnormal). This approach has several shortcomings. First, the practitioner may have no basis for determining what would be the ideal or even sub-ideal threshold. Second, the patterns of server behavior change rapidly as a factor of time, which creates a lot of overhead in manually tuning the threshold. Third, the manual setting and tuning of thresholds can translate into false alarms. And finally, application behavior can be defined by hundreds or even thousands of different metrics, and understanding what should be the threshold of each and every one of them borders on the impossible.
One anomalous metric—or even a group of several metrics, each with a single anomalous event—may be a temporary spike and is not enough to indicate a pathological anomalous behavior. Therefore it is important to look at groups of metrics and find out when they behave abnormally “together” for a significant duration. The larger the system, of course, the more difficult it is to identify the interdependencies among multiple metrics. Even upon successful identification of an anomaly, the anomaly is composed of a set of metrics that exhibited abnormal behavior “together.”

In many cases such a set may contain hundreds of individual metrics, providing a “haystack” in which to seek a “needle” of root cause. This is where metric isolation becomes necessary. It allows practitioners to identify which metric is most likely to lead the root cause of the anomalous behavior and be able not only to identify the problem but also to investigate where and why it all started.

**Solving the Anomaly Challenge: Hierarchical Anomaly Detection**

StormRunner Load implements a new and more effective “hierarchical” website load testing approach to anomaly detection. The first layer calculates anomalies on a given metric (referred to as metric-anomaly). Calculating anomalies includes two steps: (1) identify the metric distribution, (2) test each new metric value to
In many cases one metric holds all three dimensions; and in that case it is considered to be the immediate suspect. In other cases each factor is weighted and summarized to provide each metric with a responsibility rank score, allowing the practitioner to order the metrics according to the responsibility rank.

determine the probability that such a value is very low, in which case it is reported as an anomaly. Both steps (distribution and probability tests) should be adaptive where newer metric values get more weight in comparison to older values. In order for this approach to be effective in load and website performance testing, the anomaly detection algorithm must be efficient and fast to allow real-time feedback as the test progresses.

Upon identification of any new metric-anomaly, the set of all current metric-anomalies can be analyzed in order to correlate the metrics. For example, during a load test, a group of system anomalies can point to an underlying root cause in a specific infrastructure component. When implementing anomaly detection, StormRunner Load applies this technique to many types of metrics including system, server, infrastructure, and application metrics. It is also important to point out that the sampling rate for anomaly detection can vary depending on the specific metric. For example, it’s good enough to sample page availability every 1 sec where for CPU utilization the sampling rate is usually around 10 ms.

To determine the relevance of each metric, which we call “metric isolation,” StormRunner Load evaluates three dimensions to isolate a specific abnormal metric as more “responsible” than other metrics. Specifically, the three dimensions are (1) causality: which metric initiated an anomalous behavior first, (2) power: which metric was anomalous the most, (3) length: which metric was anomalous the most during the anomaly critical time. In many cases one metric holds all three dimensions; and in that case it is
considered to be the immediate suspect. In other cases each factor is weighted and summarized to provide each metric with a responsibility rank score, allowing the practitioner to order the metrics according to the responsibility rank.

The advantage of alerting the engineer to anomalies in real time is to give the engineer the opportunity to choose to terminate the test early. It is also important that multiple metrics are monitored for anomalies.

To better understand the advantages of the hierarchical anomaly detection and isolation, consider the example of a performance engineer running a regression load test for a change to an application using StormRunner Load. Once the execution of the test begins, various metrics about application performance are monitored. For each metric a normal behavior is estimated and presented adaptively as the test progresses. The performance engineer will typically define which metrics they want to monitor and for each of these metrics the system will present three primary aspects: its actual performance over time, expected normal performance over time (baseline); and, when an anomaly is detected, the system alerts the engineer that a deviation from normal behavior has been detected for this metric (see Figure 2). The advantage of alerting the engineer to anomalies in real time is to give the engineer the opportunity to choose to terminate the test early. It is also important that multiple metrics are monitored for anomalies. Consider the scenario where the metrics actively monitored by the performance engineer are all within the expected normal behavior, yet another metric not being monitored by the engineer (e.g., memory) begins to deviate...
StormRunner Load allows the engineer to see a list of six metrics, each with a number of blue bars to the right indicating the level of suspicion associated with the metric, sorted from the most suspicious to the least suspicious. From normal range. In this case, the testing system should alert the engineer about the unseen problem (see the blue circle in Figure 2) and indicate the number of SLA warnings, the number of anomaly alerts so far, and the number of errors in scripts. The number of anomaly alerts is another facet of informing the performance engineer that somewhere in the test something has become an issue and may require an active investigation and intervention.

Figure 3. Metric isolation

With anomaly detection and isolation implemented within website performance testing tools, the engineer can use the insight to speed their intuitive diagnosis and investigation into system performance. For example, when multiple anomalies are detected, it is possible to prioritize the anomalies by the most suspicious of being the initial cause of the anomalies. As shown in Figure 3, StormRunner Load allows the engineer to see a list of six metrics, each with a number of blue bars to the right indicating the level of suspicion associated with the metric, sorted from the most suspicious to the least suspicious. Upon clicking on any of these immediate suspect metrics, the engineer can further investigate the anomaly, zooming in first on a metric and then further on the relevant point in time.
As Figure 3 shows, an anomaly in one metric can be like a stack of dominoes: one started the fall but additional dominoes were then knocked over. The same approach is presented here, providing the performance engineer with a list of immediate suspects but also ranking which one was the most likely to initiate the anomaly. Clicking on a specific metric will open the metric view as described above. However, since this is a metric with an anomalous behavior a hazard icon will appear at the upper right of the metric box. When clicking on the Alert icon per metric a drop-down appears with the start and end time of when this anomalous behavior occurred, pointing the engineer to a lead for starting the investigation and diagnosis by clicking on the start or end time frame the metric timeline will zoom in on the relevant timeframe. With this automatic early alerting mechanism the performance engineer has a jump-start on diagnosis and treatment, saving time and money in the application delivery process. The tedious and time-consuming effort commonly required for analyzing and isolating performance problems has now become a simple and effective task.
Explore Hierarchical Anomaly Detection in StormRunner Load

Hierarchical anomaly detection is a new capability in StormRunner Load, the cloud-based, SaaS-delivered load and website performance testing solution. It enables teams to use powerful analytics to visualize and spot anomalies and performance problems and find root causes using real-time metrics. StormRunner Load makes it easy to plan, run, and scale Web and mobile testing, and enables teams to run tests in minutes from anywhere and any device.

Learn More At
www.microfocus.com/srl (30-day free trial)