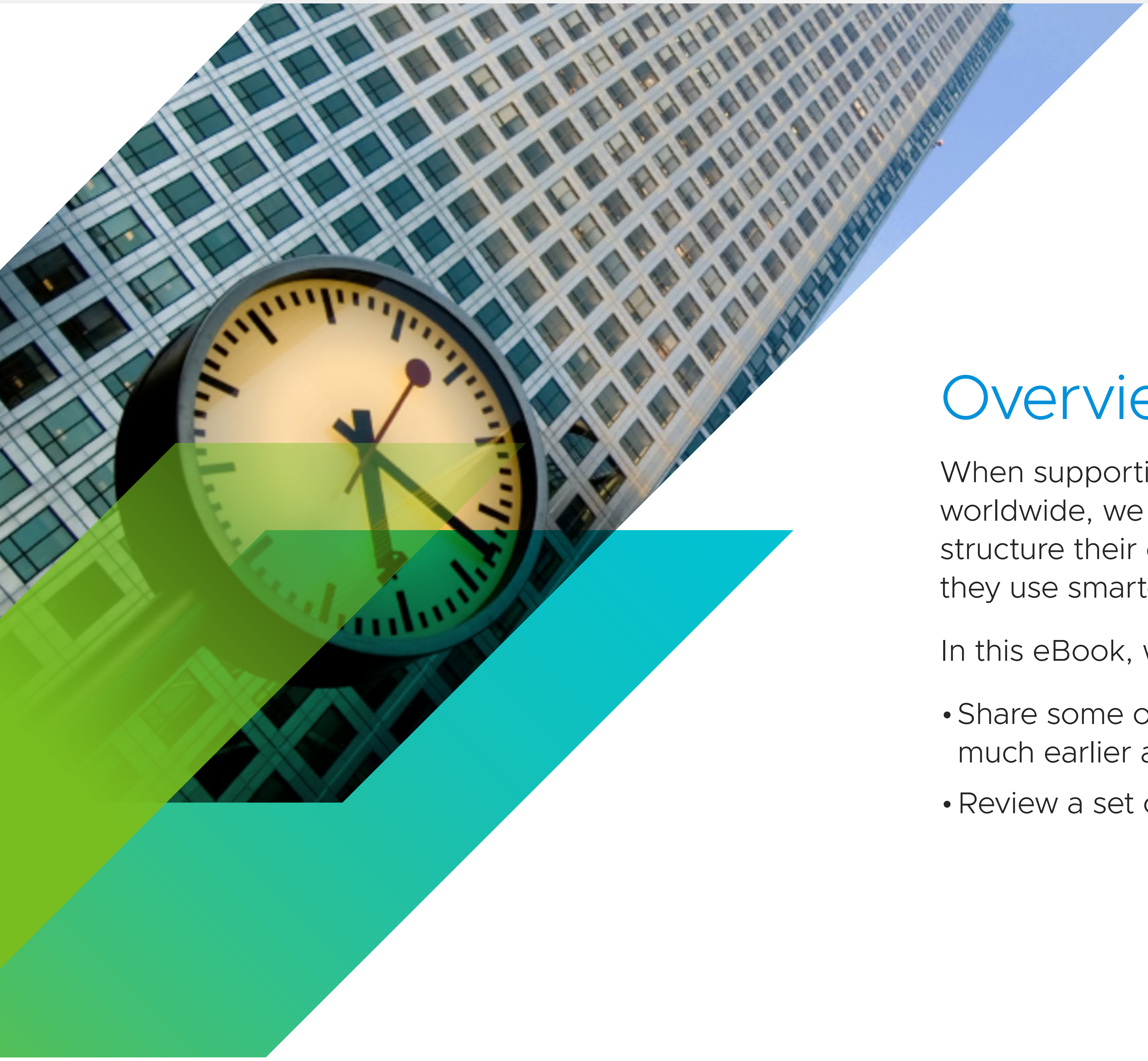
VMware Tanzu Observability

10 Examples of Smarter Alerting

How to use analytics to detect hidden anomalies in cloud application environments.

A guide for SRE, Dev and Ops teams who need to be proactive in finding problems before service is affected, without debilitating alert noise.

vmware®



Overview

When supporting some of the largest and most successful SaaS and digital enterprise companies worldwide, we at VMware get to learn from our customers on a regular basis. We see how they structure their operations, how they implement their monitoring and automation policies, and how they use smarter alerts to lower mean time to identify and mean time to automate.

In this eBook, we:

- Share some of the things we've learned in how these companies use analytics to identify problems much earlier and accurately.
- Review a set of examples where customers have leveraged these more actionable, smarter alerts.

Table of Contents

Introduction to Smarter Alerting

4 How do you know if a disaster is brewing in your cloud?

5 Evolution of alerting with the DevOps movement

7 Smarter alerting—derived from metrics-based anomalies

8 Better metrics—the basis for smarter alerts

9 Anomalies—something that's not normal, tells us when to act

12 Events and metrics—similar but different

13 Analytics—one way to find anomalies is better than the other

10 Anomaly Examples for Smarter Alerting

14 Range

15 Windowed deviation

17 Seasonal ratio

18 Constant rate

19 Group range

20 Group variance

21 Group sum

22 Ratio of two series (correlation)

24 Inverse-correlation

25 Consistent frequency

26 Plus, any Boolean combination or functional composition

Putting it All Together for Smarter Alerting

27 What is the shared theme across these examples?

28 Anomalies reflect unexpected behaviors, not values

29 Here's the secret to smarter alerts

30 Intro to Tanzu Observability Platform

33 How to create smart alerts in Tanzu Observability

34 Wrap-up

How Do You Know if a Disaster is Brewing in Your Cloud Application Environment?

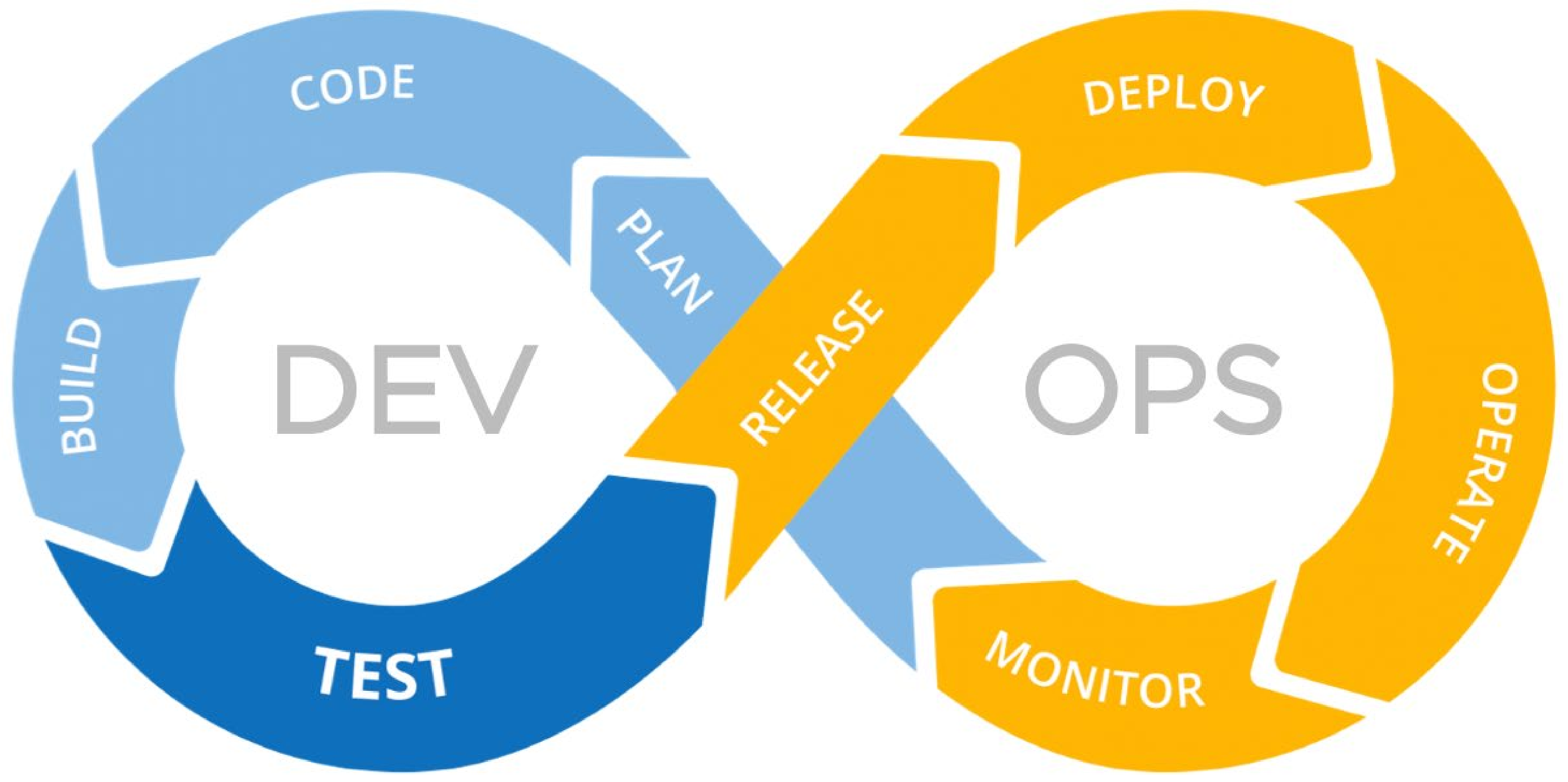
Historically, cloud application systems have been instrumented primarily with log files for identifying and diagnosing issues. There are two major issues with this approach:

- It tends to be backward-looking, with no way of projecting forward to prevent issues as they develop.
- Alerts derived from these systems are simplistic and event-oriented. They're one-dimensional. Because the alerts mostly say, "Something happened," with little supporting data, finding and repairing the causes can require significant debug effort.

The goal of this eBook is to bring to you the notion of Smart Alerts—a smarter, more sophisticated approach to monitoring that can both get ahead of a looming major issue and make resolution easier and faster. We'll start by looking at what makes a Smart Alert, motivated by the concept of metrics-based anomalies. We'll then go through ten different examples of alerts so that you can apply to your monitoring strategy, improving uptime, performance, and drastically shortening the time to incident resolution.

How Alerting is Evolving with the DevOps Movement.

One of the more interesting operational patterns we see is the notion that companies aren't scaling up their Ops teams anymore. Rather, they scale out their Dev teams instead. All of our larger customers do this as there's just too much complexity to have a large team of 30+ Ops people handling the entire system. As they scale Dev, the developers end up writing their own alerts using 'self-serve analytics.' Why? The developer has intimate knowledge of their code. So, they instrument their code, they deploy patches themselves without any kind of interference from Ops, and they observe the impact of that code running live (with tools like VMware Tanzu™ Observability™). To keep a continual eye on how things are working, they also create informational notifications—more than alerts per se—just to get a deeper sense of how their code actually behaves in production. Some Dev teams do this for their piece of the overall system. SREs will do it for the whole system.



Now that usually is way too much information for the Ops teams, as they don't have the luxury to work on much more than 'severe' notifications or important events happening right now. Ops teams work on finding and fixing things that are truly actionable. Meanwhile, as more developers understand how their code runs in production, the more issues the company is able to avoid proactively before they become true operational problems.

So, we see Dev teams using a spectrum of more informational alerts, deeply observing their code in production—like a remote profiler or a debugger—and then Ops teams are watching actual problems right now. But to do both, you need a couple of things. First, you need a deep set of data across your entire stack. Then, you need a very powerful language to express anomalies that go far beyond simple thresholds. This brings us to anomalies and metrics-based analytics.

Smarter Alerts and Visualizations are Derived from Smart Metrics-Based Anomalies.

In order to unpack the statement above, let's look at some less-smart alerts so that we understand where they can fail. We'll then look at anomalies to see how they provide better information and give you more power to spot and fix issues quickly.

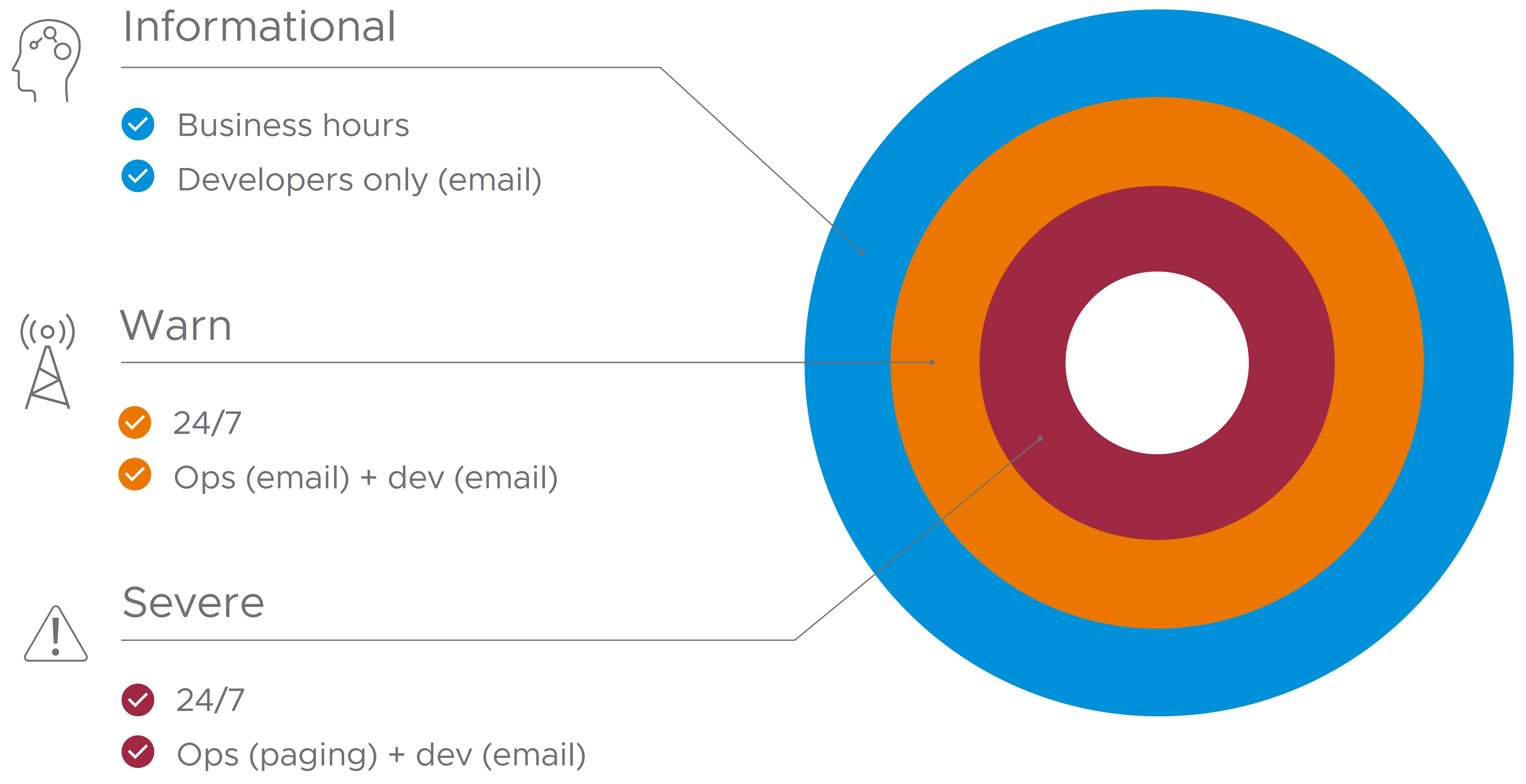
First, not-so-smart alerts are:

- Event-oriented, meaning that they let you know that something of interest happened, but not why it happened.
- Often based on logs, which record past events for future analysis.
- Typically derived from some simple threshold being exceeded, but they reflect only "univariate" data: that is, a single raw data point or source in isolation.
- Tend to provide no severity nuance; it's either "The world is a beautiful place" or "The world as we know it is ending." Nothing in between.

The real-world problems that result from this are either too many false alarms or no alarm when one is needed (false positives and negatives). They can't be better qualified by correlating them with other metrics; that must be done by hand. And the result is that many people must get involved in trying to sort issues, and all that personnel and time becomes a big productivity drag. And the longer a problem remains unsolved, the longer your customer may have a negative experience that harms your reputation.

Better Metrics

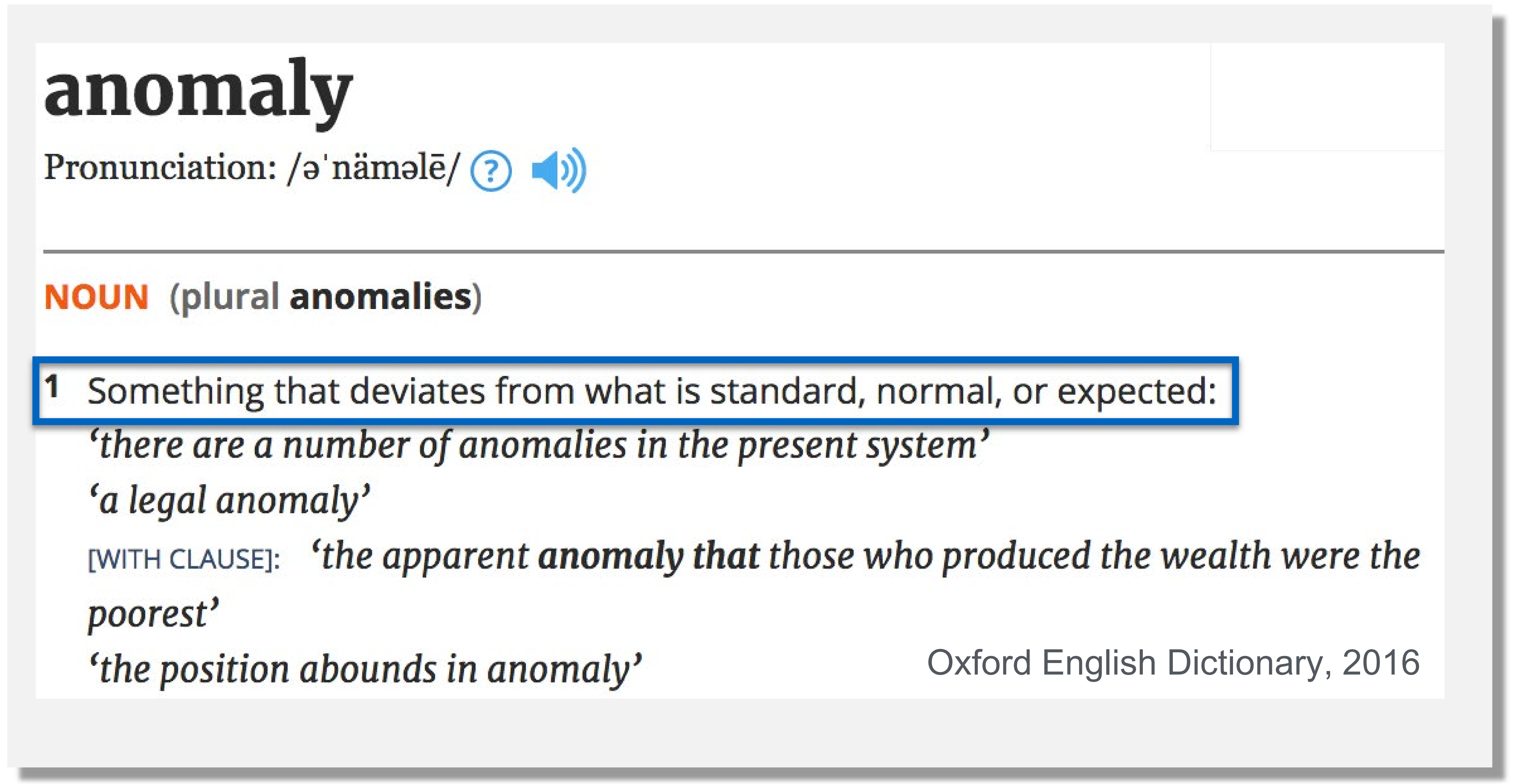
Let's look at some less-smart alerts so that we understand where they can fail. We'll then look at anomalies to see how they provide better information and give you more power to spot and fix issues quickly. What you really want is a way to combine different pieces of data into a "multi-variate" metric that provides more richness and accuracy. This requires a more expressive language to establish these relationships. It also lets you set up a severity hierarchy for self-serve support and graduated, targeted escalation only as needed.



Anomalies—Something That's Not Normal, Tells Us When to Act.

An anomaly is something that's not normal.

It may not be a problem that your customer sees—yet—but something is happening that's not business as usual.



anomaly

Pronunciation: /əˈnäməlē/

NOUN (plural **anomalies**)

1 Something that deviates from what is standard, normal, or expected:

'there are a number of anomalies in the present system'

'a legal anomaly'

[WITH CLAUSE]: *'the apparent **anomaly** that those who produced the wealth were the poorest'*

'the position abounds in anomaly'

Oxford English Dictionary, 2016

Anomalies—Something That's Not Normal, Tells Us When to Act.

Anomalies tell us we need to act.

Ideally, we get a heads-up and can plan our actions before something bad happens.

But, on occasion, we find out too late, and preventive measures are replaced by rescue and recovery.

Planned Work

Unplanned Work (Post-anomaly)

Events and Metrics: Similar but Different

Event tracking, typically in the form of logs, has similarities with metrics—but the differences are critical. Both are timestamped and can provide any kind of message. The differences are that:

- An event records an occurrence: someone logged in; a load balancer marked down a service; a web page was requested. An event is not a measurement.
- A metric measures something. Perhaps the CPU load of a given host or the total number of orders that an application has completed. Metrics include numbers.
- An event log is basically a list, and each item has a very weak, if any, connection to other useful and potentially correlated information. Metrics, on the other hand, are functions of multiple primitive measurements; almost by definition, they imply a strong connection between different sources of measurement.
- A log file may contain duplicate or redundant entries that clutter the picture with no additional information. Metrics, by contrast, naturally avoid both duplication and conflicts, since metrics databases can filter duplicates, and a well-crafted metric can resolve its own conflicts.

Events (logs)

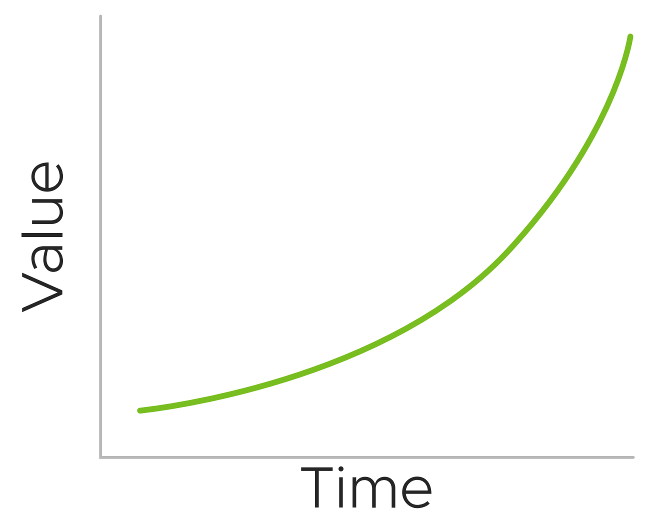
- An occurrence
- Timestamped
- Arbitrary annotations
- Duplicates valid
- List metaphor - Weakly connected

Metrics

- A (numerical) measurement
- Timestamped
- Arbitrary annotations
- Duplicates/conflicts invalid
- Function metaphor - Strongly connected

- Function machinery
- Columnize/Vectorize for speed/cost
- Visual interpretations @ scale

```
20160821/12:37:01 Login [bettysue@gmail.com]
20160821/14:12:45 Login [johndoe@gmail.com]
20160821/14:12:45 Login [johndoe@gmail.com]
20160821/15:11:28 Login [jane@gmail.com]
```

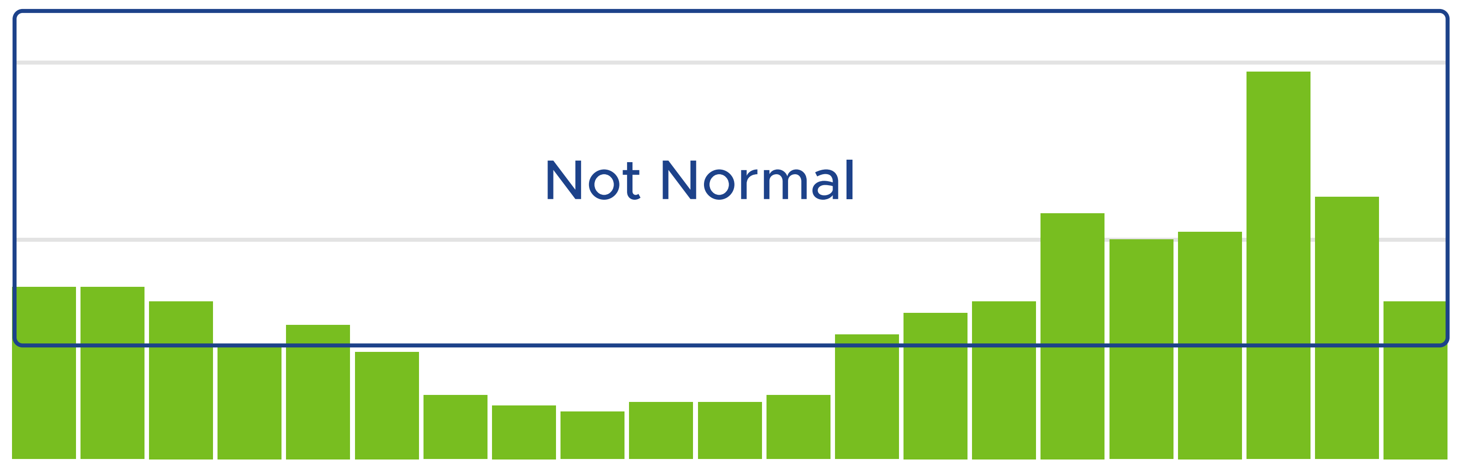


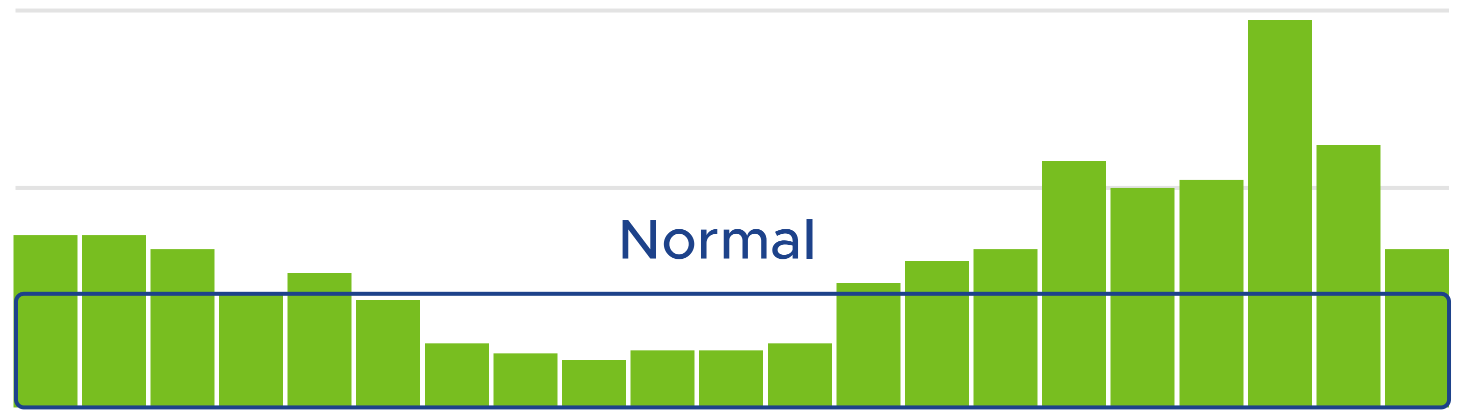
```
cpu.loadavg.1m host=app-1

[0.42,0.37,0.41,0.41,0.5,...]
```

Anomalies can be defined in two ways.

You can define an analytical function to make it easier to see anything that should be considered anomalous, or you can define what's normal and say that anything not within that normal range will be, by definition, an anomaly. The latter method is typically often a simpler approach, and what most engineers start with.

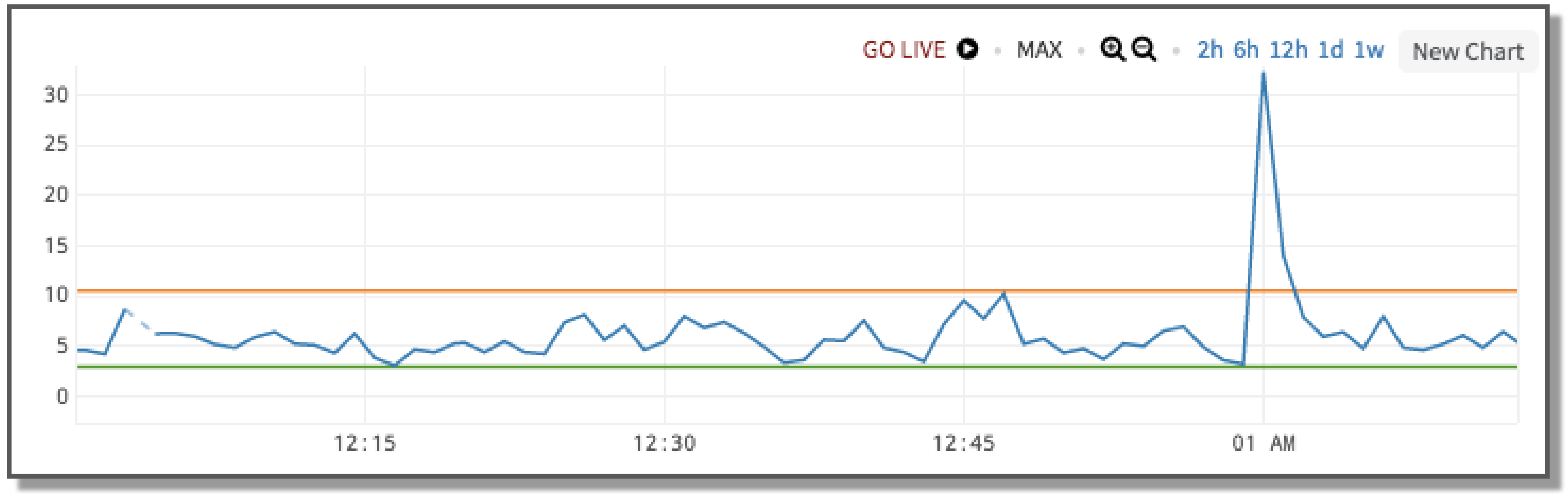




10 Anomaly Examples for Smarter Alerting with a Shared Theme.

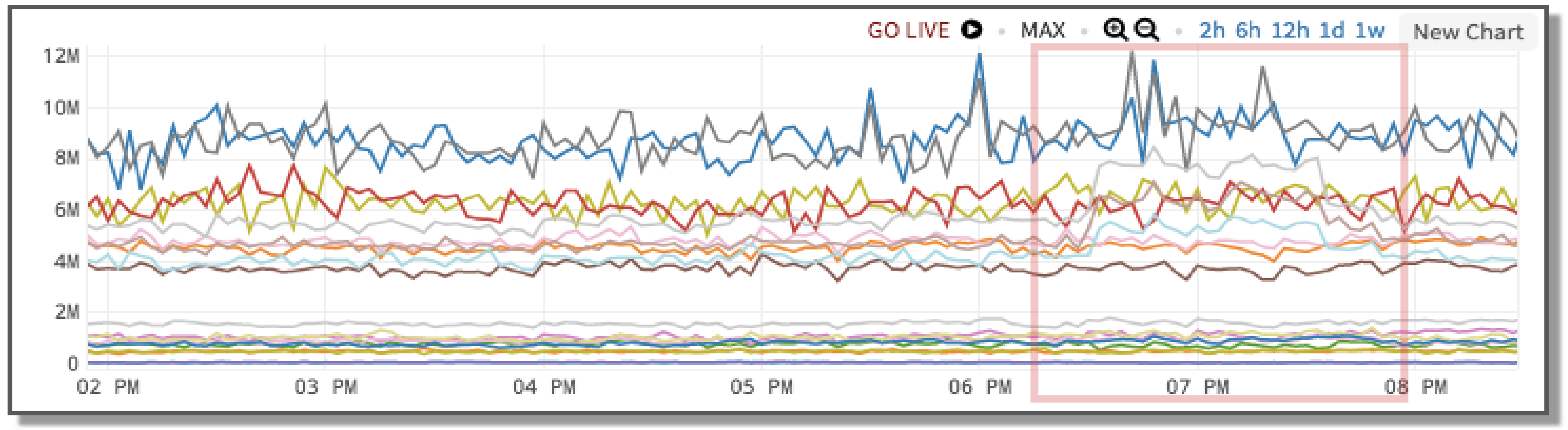
1. Range

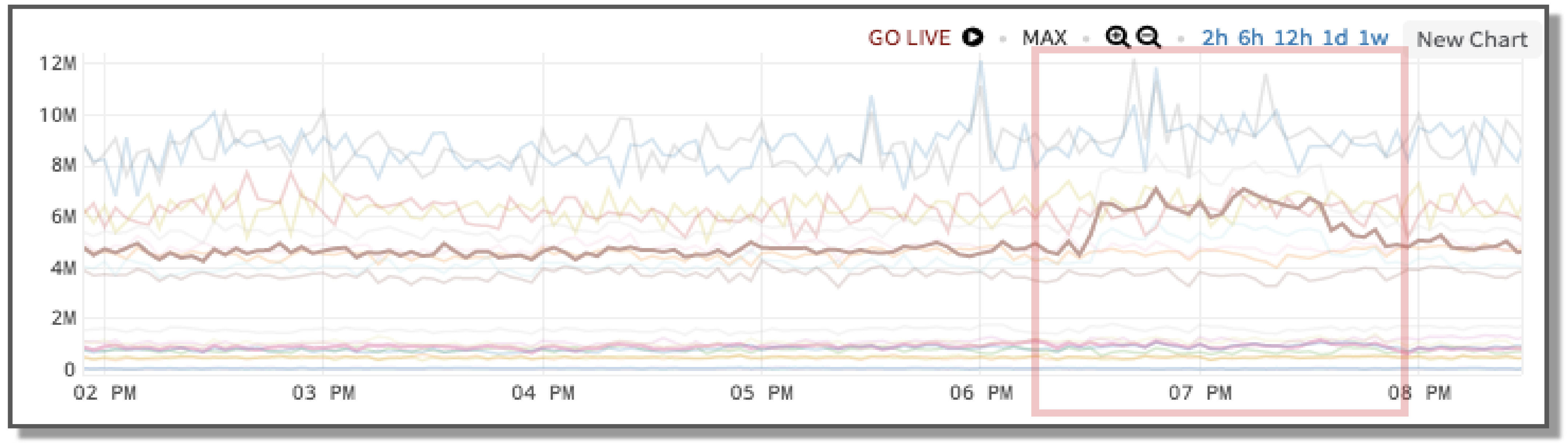
This is a simple anomaly that's visible to the naked eye. A metric has clearly departed from its baseline normal range, and that excursion is an anomaly.



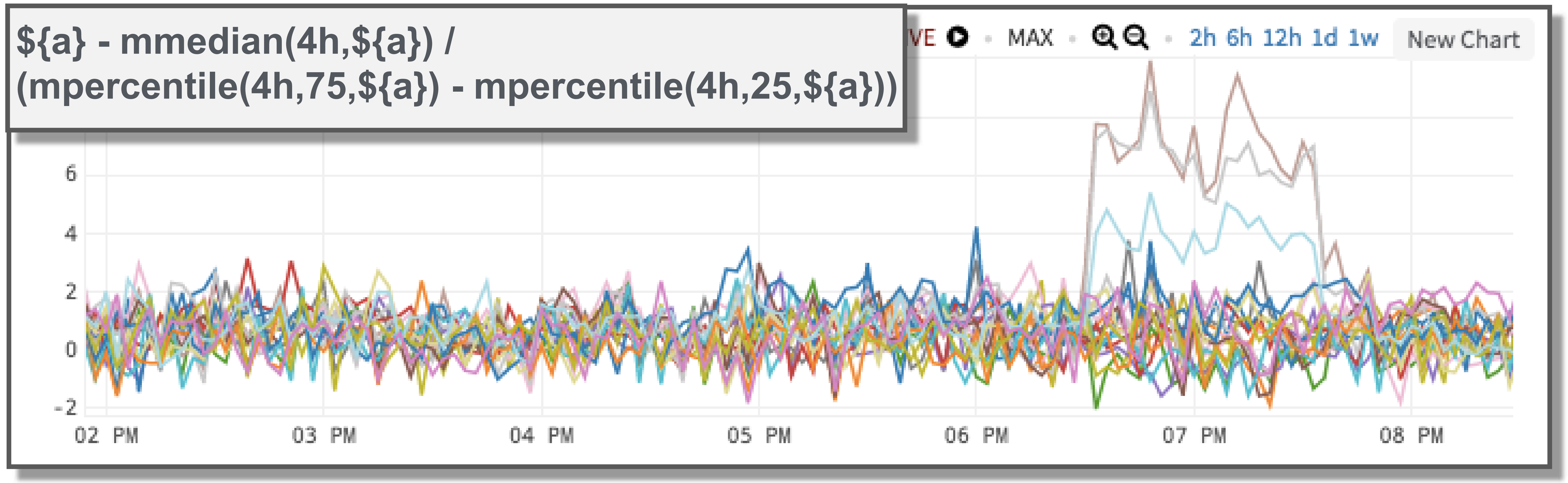
2. Windowed deviation

Because of the number of series reported in the chart below, it's impossible to pick out the anomaly visually. If you highlight the offending series, it's easier, but we were able to do that only when we already knew the answer. What if we don't already know the answer?





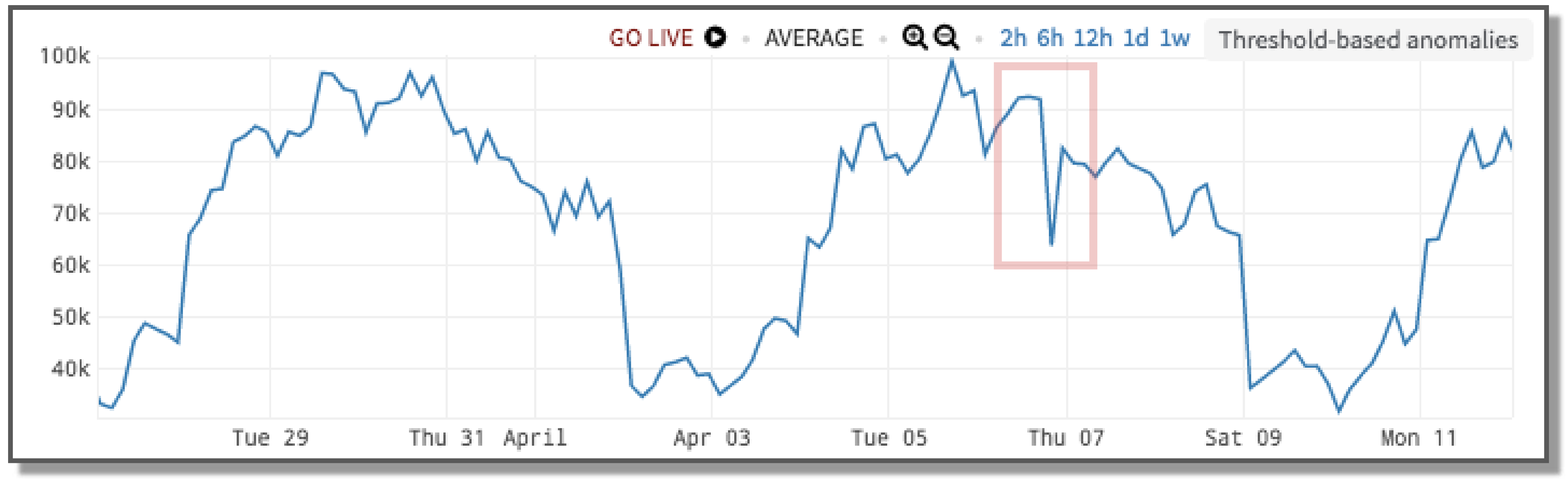
By looking at how values deviate from their general historical range, you can see where the anomaly is. In this case, the deviation within a time window is the baseline, and the departures from that baseline make up the anomaly.

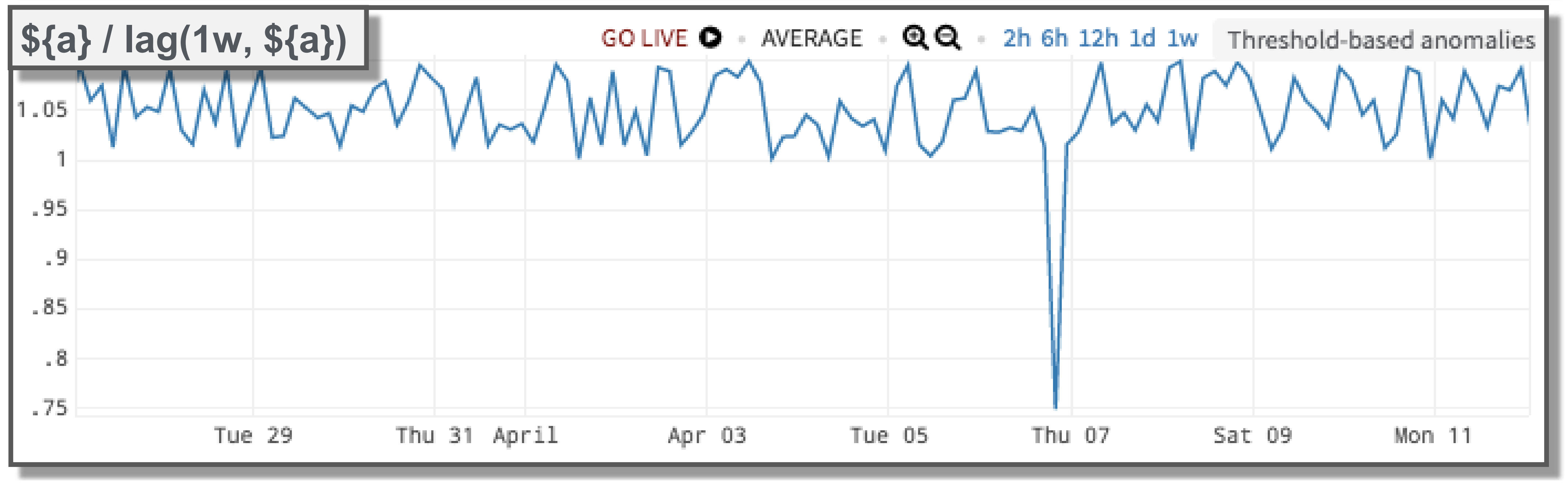


```
${a} - mmedian(4h,${a}) /
(mpercentile(4h,75,${a}) - mpercentile(4h,25,${a}))
```

3. Seasonal ratio

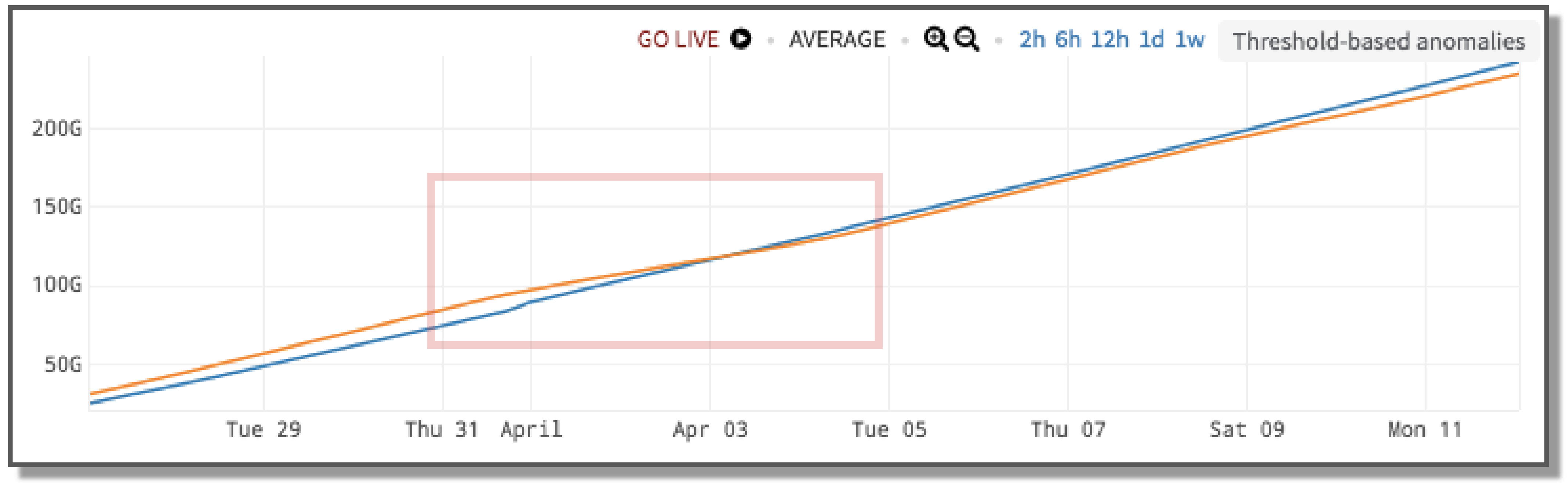
In the top chart below, you see a relatively irregular pattern. Why is the portion enclosed in the square any different from the other jags? You can see this much more easily by comparing each point to the identical point exactly one week earlier; now the anomaly stands out. Here that seasonal ratio establishes the baseline, and the anomaly deviates from that.

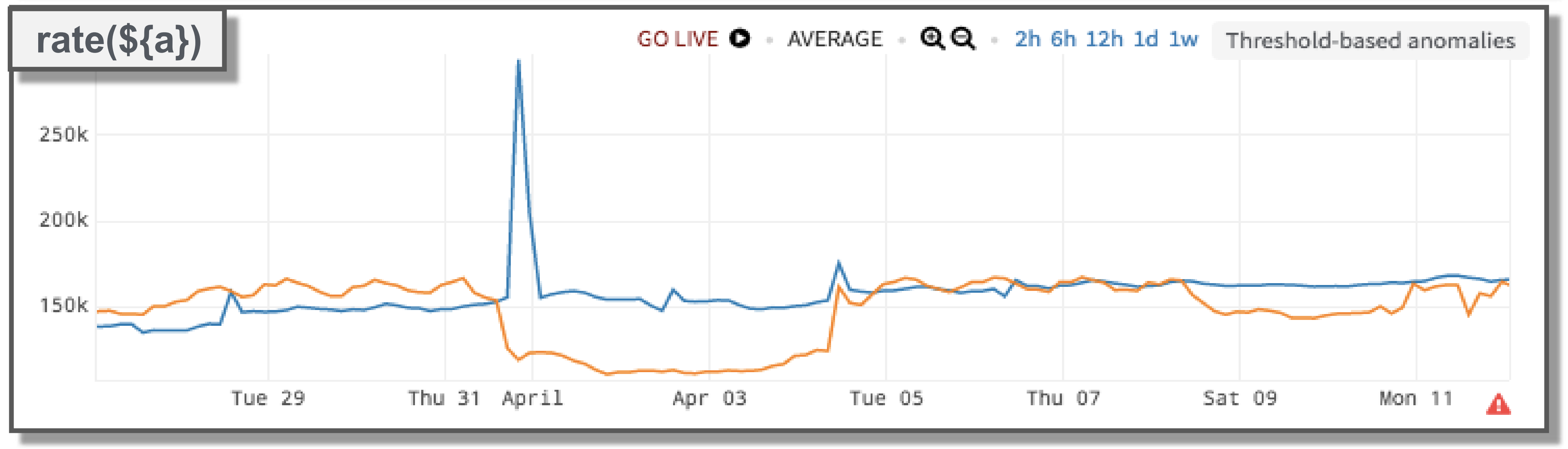




4. Constant rate

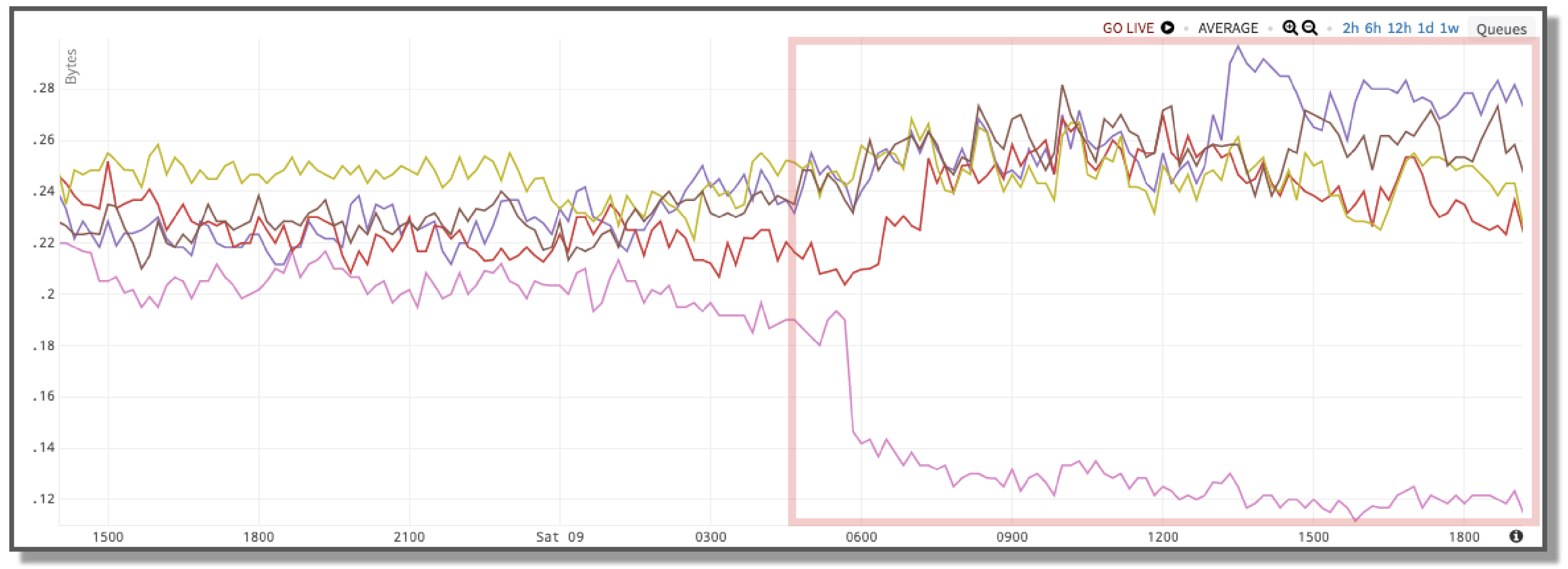
An anomaly is very hard to see in the top chart below, and yet one lurks. If you look at the rate of change, the baseline is constant. What appears to be a subtle shift showing nothing of concern becomes much more obvious looking at the rate.

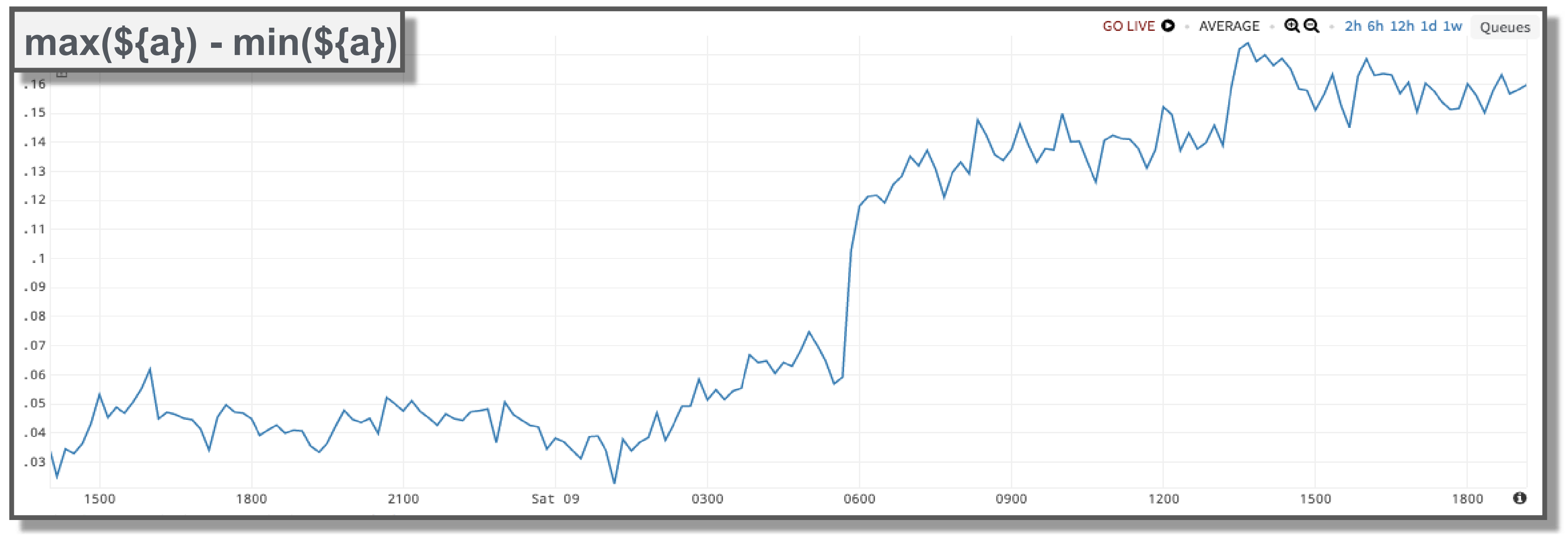




5. Group range

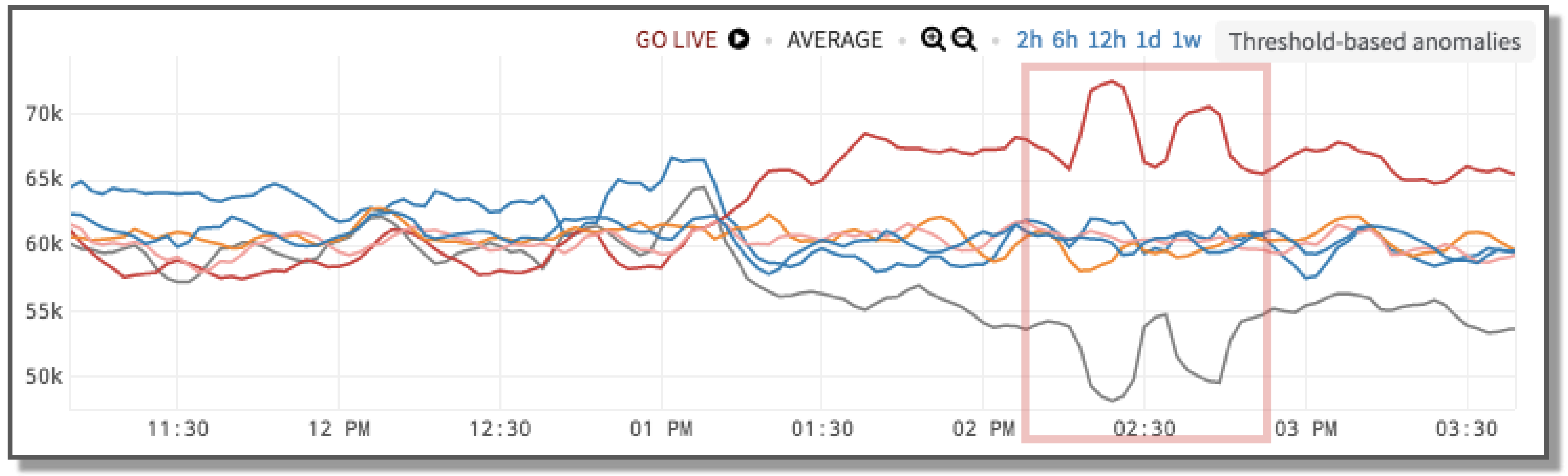
The next chart depicts a data pipeline. The bottom series in the top chart shows the output of the pipeline. The output rate slows way down, even while the throughputs of the intermediate stages actually go up. This suggests a stall in the pipeline. It can be more easily seen by looking at the range of the group of series.

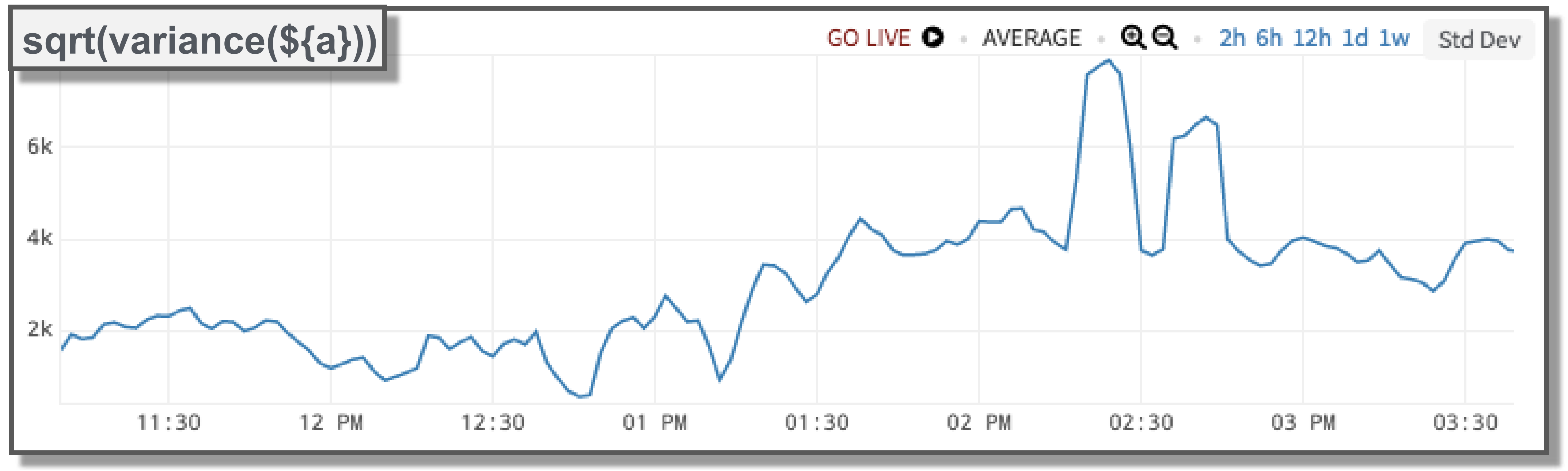




6. Group variance

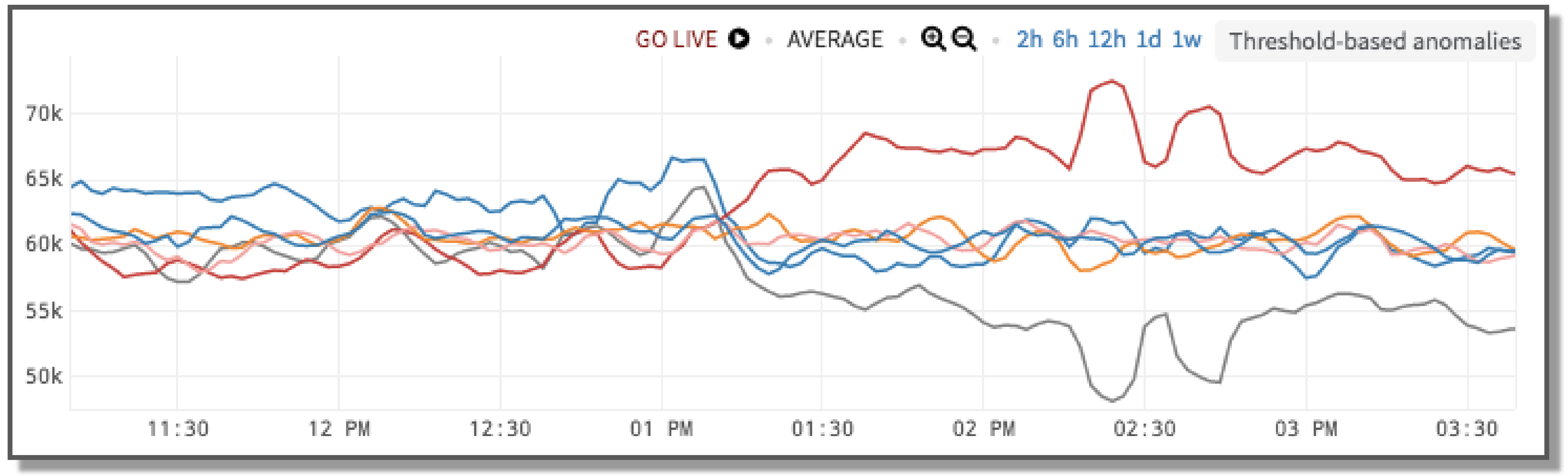
The top chart below shows the loading on various servers as allocated by a load balancer. What's important here is not just the load on any individual server, but the range of loading—if too high, then the servers are out of balance. Using a variance, the anomaly is clearly distinguished from baseline in the bottom chart.

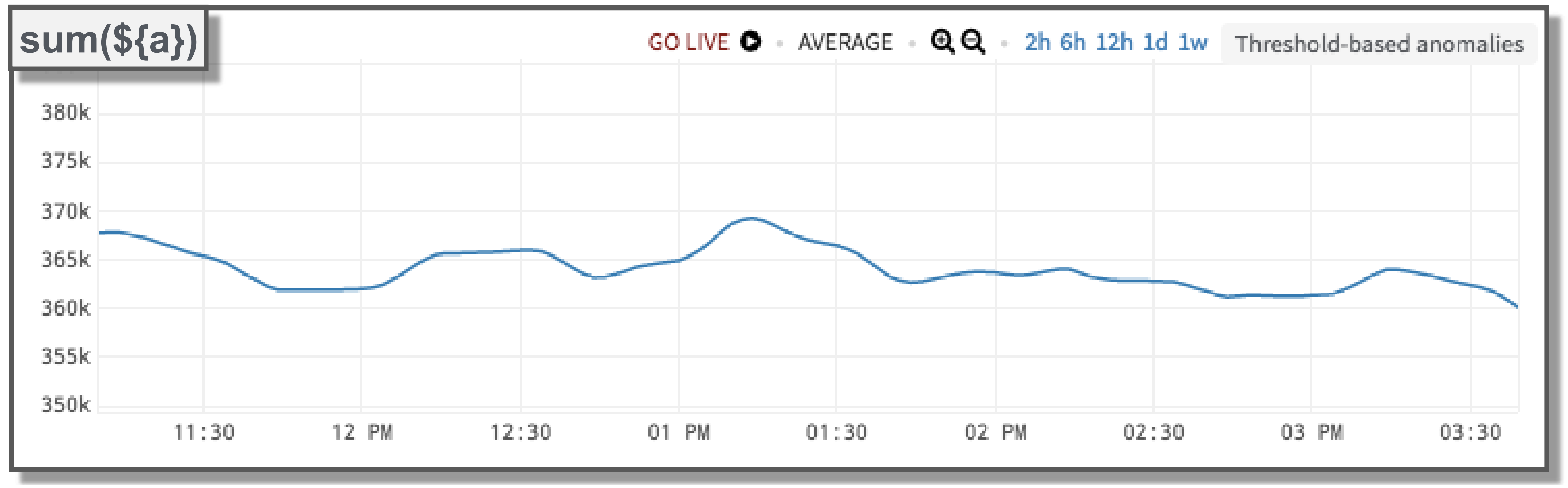




7. Group sum

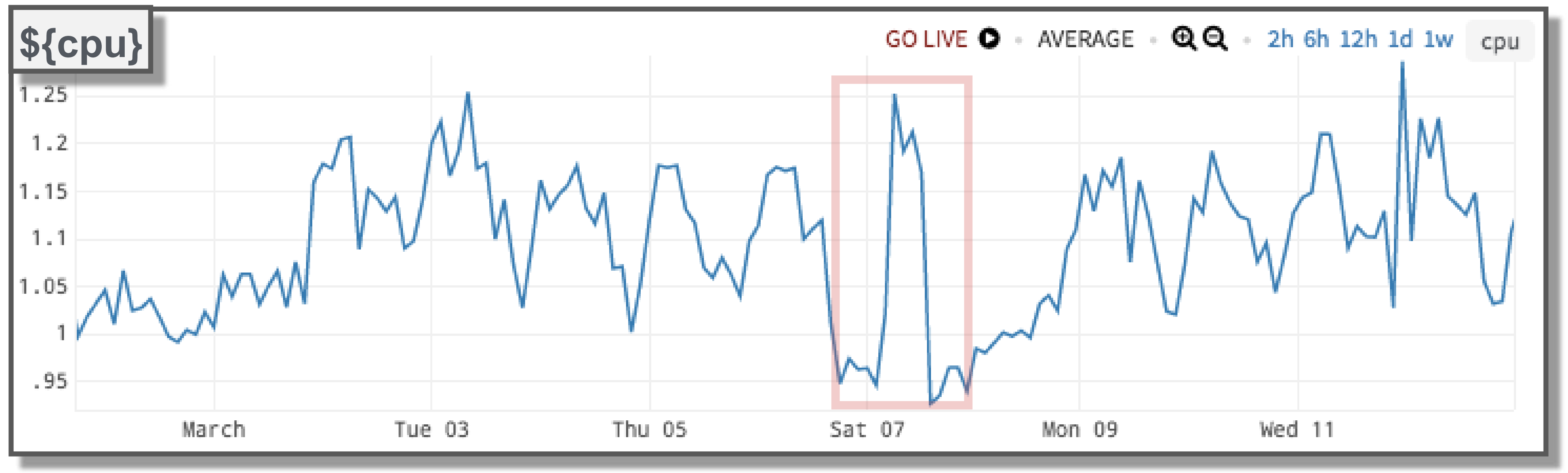
Anomalies may be in the eye of the beholder. The consumer of the server farm output sees only that the total number of tasks processed appears to be normal. Because they're not concerned about the inner workings of the balancer, to them there is no anomaly.

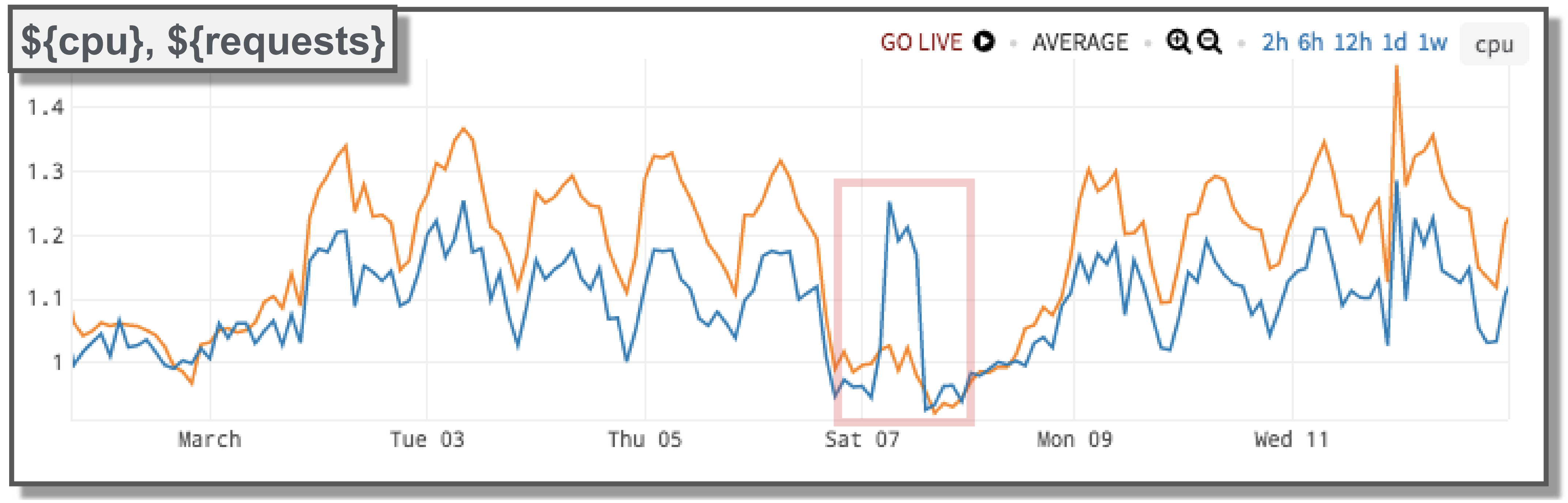




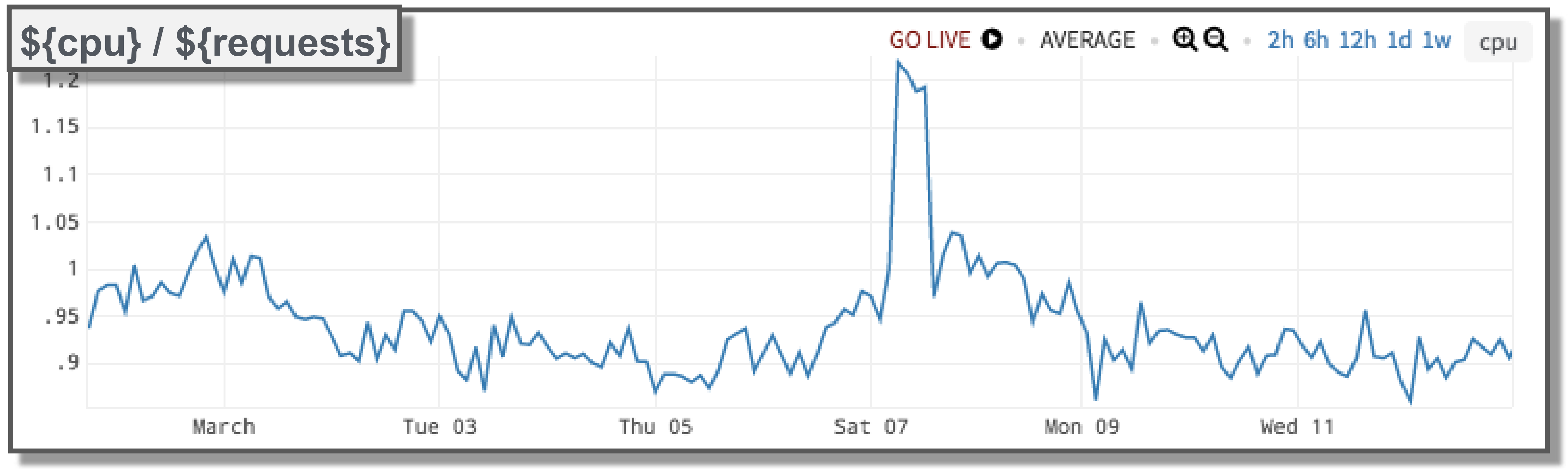
8. Ratio of two series (correlation)

Here we look at CPU loading in the top chart. There would appear to be nothing unusual going on, but, in fact, there is an anomaly. It's easier to see if we look not just at the CPU load, but also at the number of requests sent to the CPU. Ordinarily, CPU load moves in synchrony with the number of requests. But at the point of the anomaly, that isn't the case.



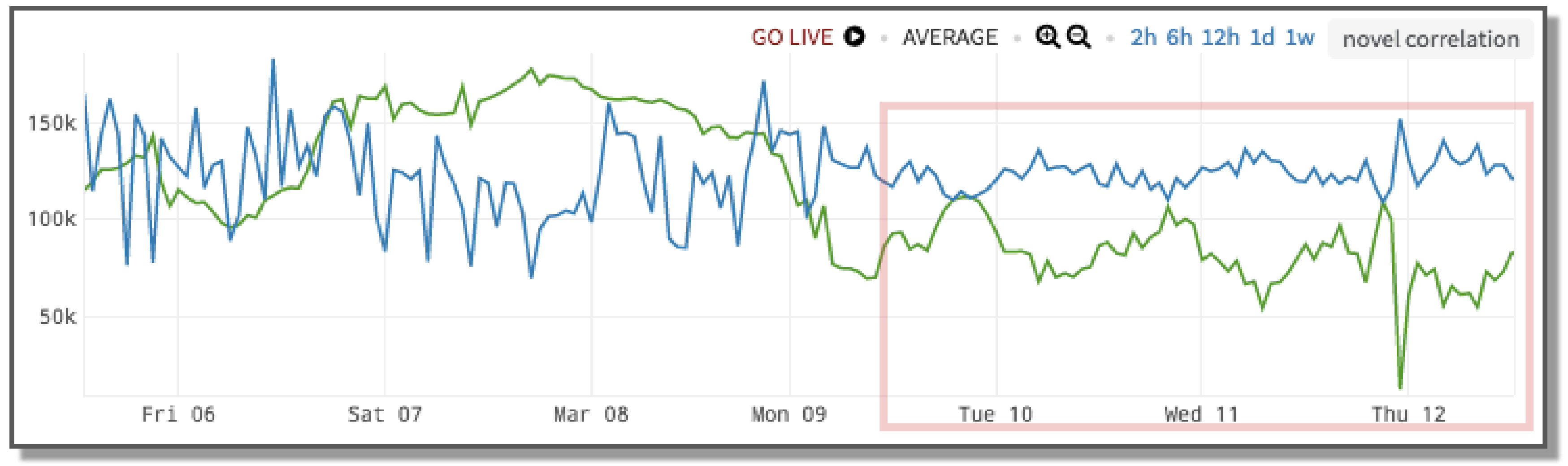


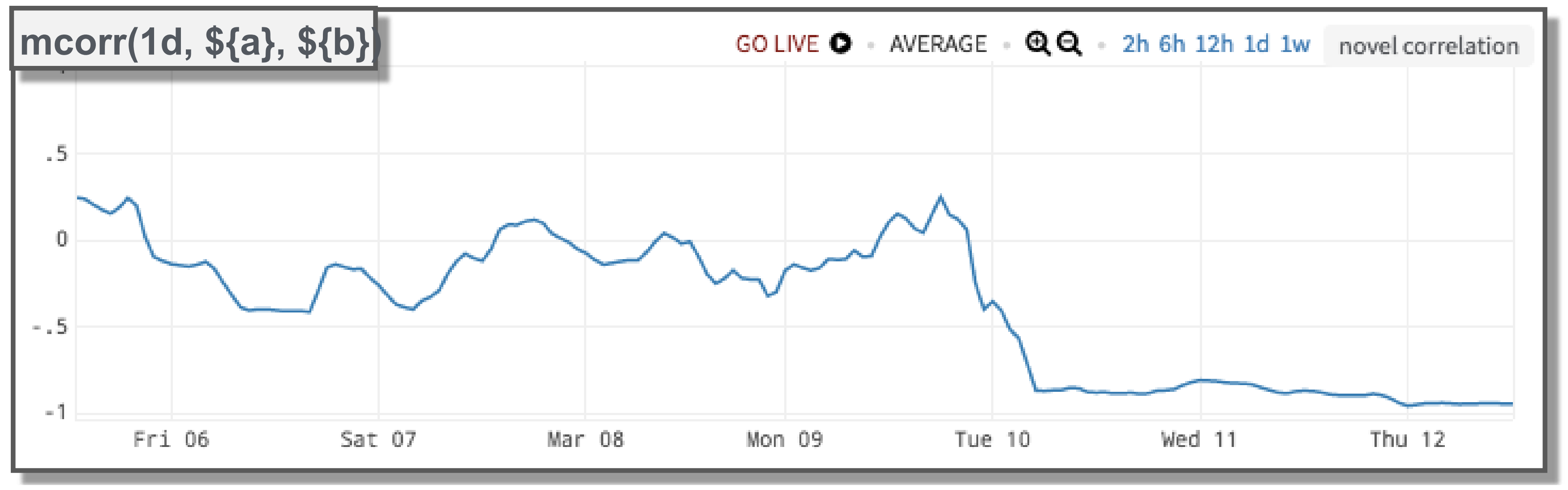
We can define a metric in this case as the ratio of loading to requests. The deviation from baseline shows an obvious anomaly.



9. Inverse-correlation

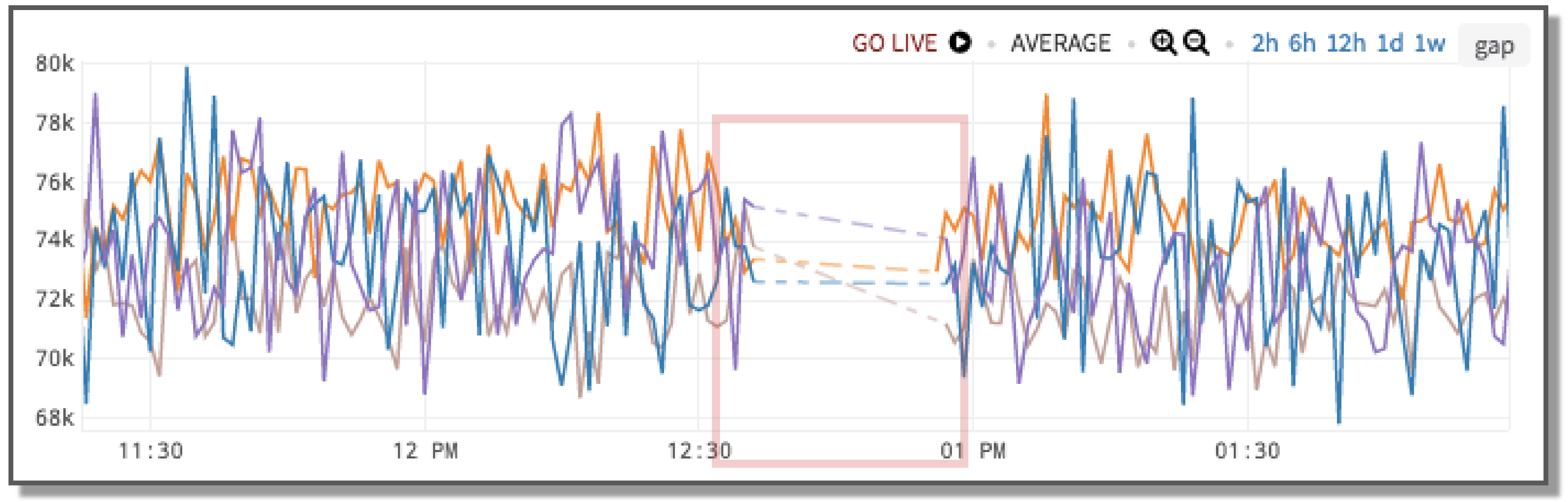
The top chart below shows two series. For the early part of the trace, they appear to behave completely independently, showing no correlation. But towards the end, they behave in opposition to each other—when activity on one goes up, the other drops. This inverse- or anti-correlation happens to be the result of a resource that the tasks from the two series both utilize. As long as that resource is plentiful, then both can use it, each unaffected by the other. But when that resource becomes saturated, then use by one makes the resource unavailable to the other. Measuring the inverse-correlation makes the anomaly clear.

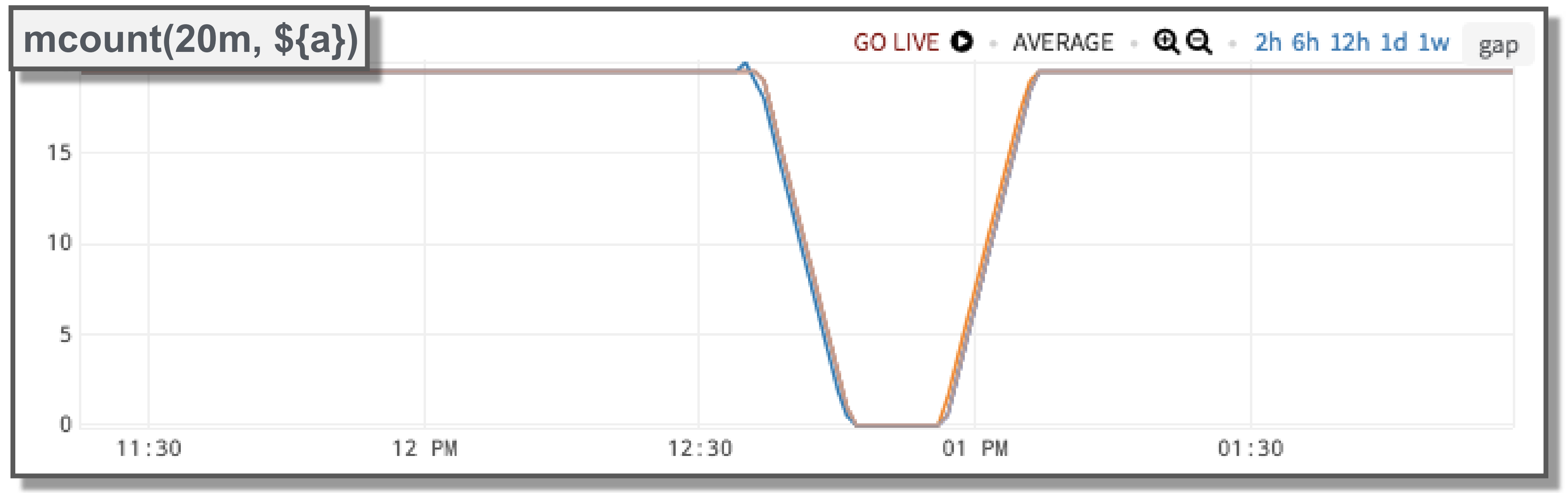



mcorr(1d, ${a}, ${b})

10. Consistent frequency

In this example, metrics reporting stops for a period of time. It's clear to see visually, but how do you define the anomaly mathematically for automatic detection? You can do this by counting the number of data points within a sliding window. You'll notice that this anomaly appears shifted right as compared to the cessation of reporting. That's because it takes some time for the sliding window to contain only empty data, and then for it to contain only real data after the anomaly ceases.




mcount(20m, ${a})

Any Boolean combination or functional composition.

The previous 10 examples have focused on single functions to detect anomalies. But, in reality, any functions—logical or mathematical—can be combined to build up sophisticated metrics. Definitions can be long and complex, going on for pages. Once your low-hanging metrics have been addressed, this is where your sophistication can help you differentiate your service to your customers in ways that might not be obvious to your competitors. Two examples are shown below.

```
default(0, 1m, 100*
(sum(rate(ts(httpstatus.api.daemon._sshDaemonId_.pushdata._workUnitId_.POST.406.count,
tag=prod and (tag="*-primary" or tag="*-secondary") and not tag=customer)), hosttags)) /
((sum(rate(ts(httpstatus.api.daemon._sshDaemonId_.pushdata._workUnitId_.POST.406.count,
tag="*-primary" or tag="*-secondary")), hosttags)) +
(sum(rate(ts(httpstatus.api.daemon._sshDaemonId_.pushdata._workUnitId_.POST.2*.count,
tag="*-primary" or tag="*-secondary")), hosttags)))) > 10
```

```
mavg(10m, avg(last(4w, ts(custom.globalAllowedRate, tag="*-primary" or tag="*-secondary")), hosttags)
as globalAllowedRate *5/6 - sum(rate(ts(dataingester.report-points, (tag=retired or tag=prod) and
(tag="*-primary" or tag="*-secondary" or tag="*-tertiary"))), hosttags) + 0 *
sum(rate(ts(dataingester.report-points, tag="*-primary" or tag="*-secondary" or tag="*-tertiary")),
hosttags)) - $globalAllowedRate *5/6 * 0.02 < 0
```

So What’s the Shared Theme Across These Examples?

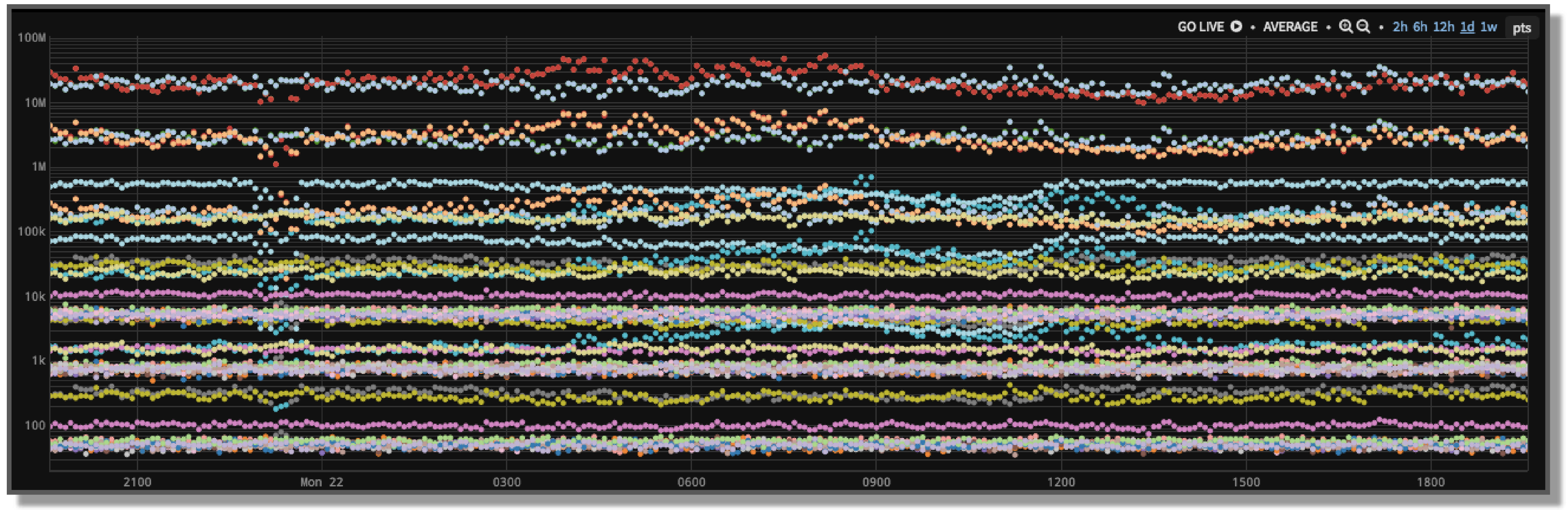
Given all of the mathematical power available to isolate these anomalies, notice the transformation that all of these analytic functions lead us to: a range of a value.

Why is that? It’s because, psychologically, we have evolved to be able to notice value differences easily. Without these functional transformations, the anomalies in the data streams aren’t available to us in such a built-in manner.

If we look at the two images below, we can easily detect which is the largest animal in each picture. We know instinctively. Even the other owls are all looking at the biggest one.

Most Anomalies Reflect Unexpected or Non-Normal Behavior, Not Values.

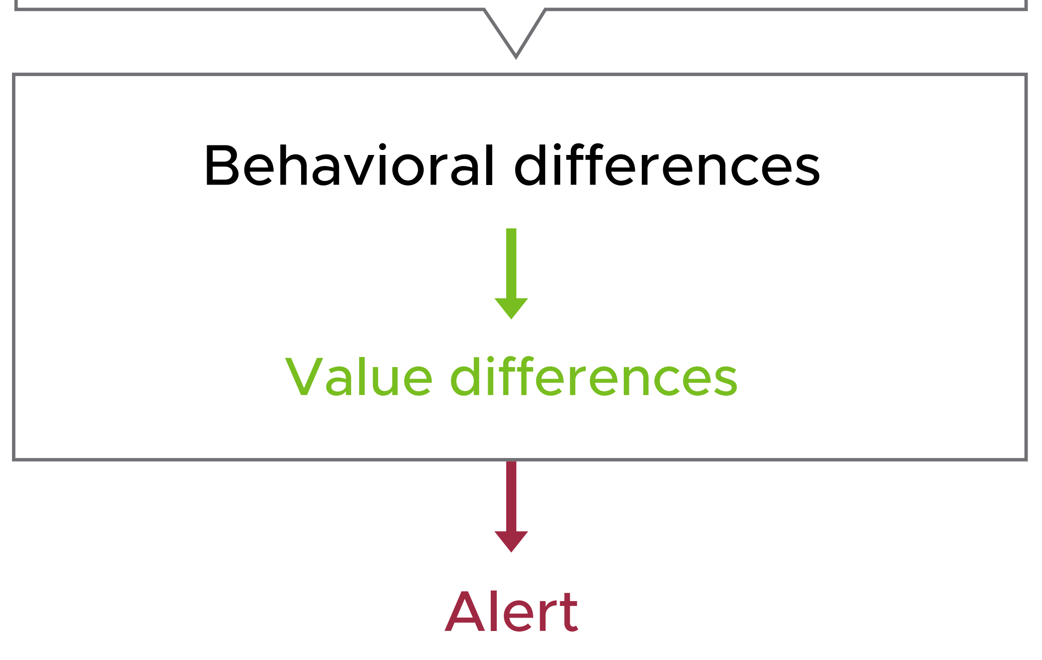
If you look at the following image and ask yourself which of the series below is showing anomalous behavior by reporting less frequently than the others, there's no way you're going to be able to figure that out by inspection. But analytics can transform and isolate hidden anomalies clearly.



The Secret to Smarter Alerts: Convert Behaviors to Values

2. Windowed deviation
3. Seasonal ratio
4. Rate
5. Group range
6. Group variance
7. Group sum
8. Correlation (of two series)
9. Non-correlation (of two series)
10. Consistent frequency

and any **Boolean combination** or **functional composition** of these



The way to create smarter alerts is by taking the behaviors and converting them into value-based metrics that we can then see easily. That's what happened with the 10 examples in the previous pages—except for the first one, which was already a range.

Once you have converted behaviors into values, you can readily define alerts that key off of those value differences when anomalies occur. Said another way, it's not about making alerts themselves more sophisticated; it's about defining more sophisticated metrics that can then trigger simple alerts.

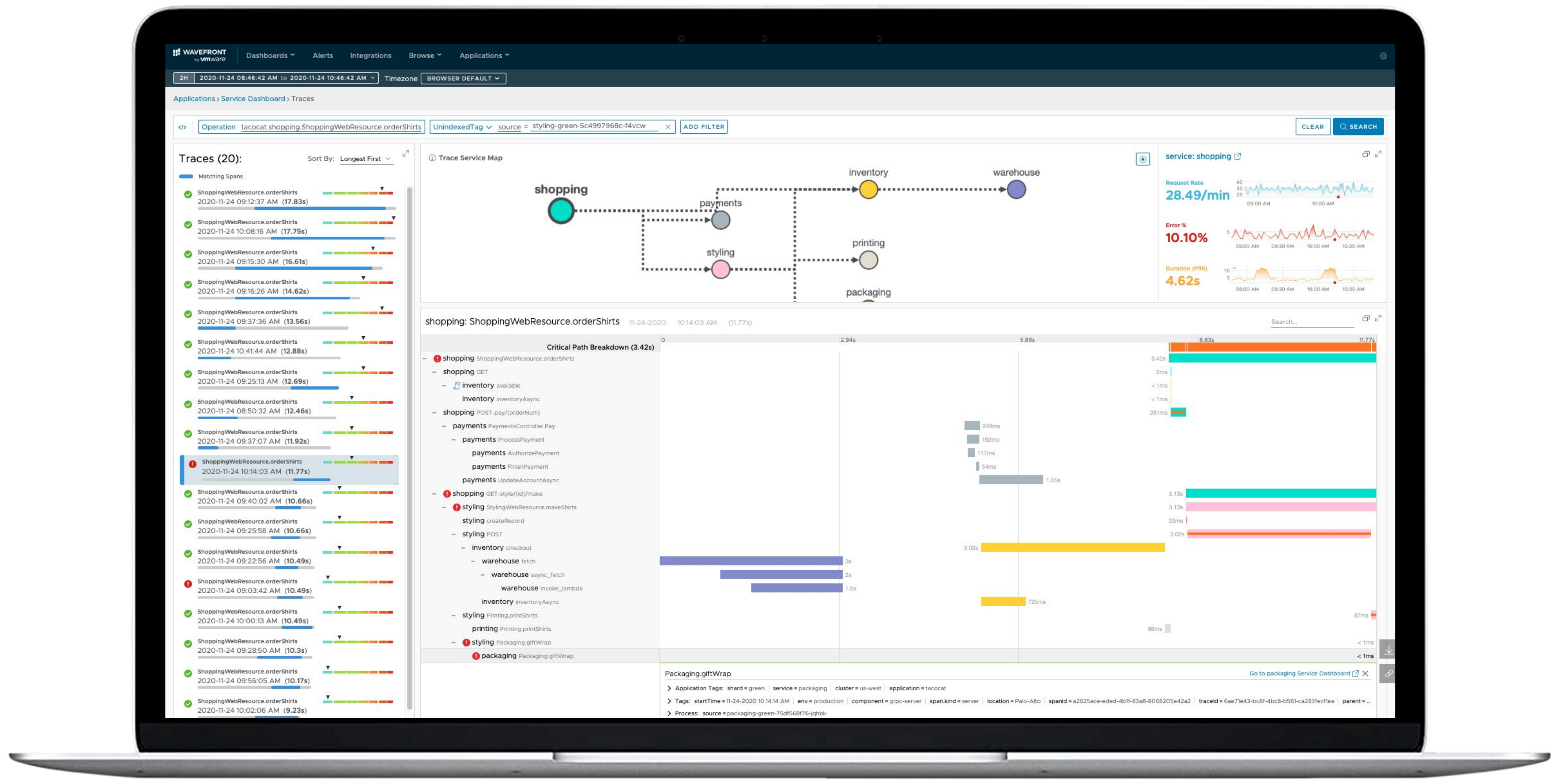
Introduction to the Tanzu Observability Platform

Tanzu Observability is the leading cloud-delivered observability platform for everyone needing to know how well code is running in production (and impacting the business). Dev and Ops teams and the world's leading SaaS and digital enterprise companies use Tanzu Observability for alerting, troubleshooting, reporting and optimizing their workloads running on multiple clouds.

What makes Tanzu Observability truly different is that it gets you to the "why" faster using AI-driven analytics on your unified repository of full-stack performance data—metrics, traces, histograms, span logs, events.

The resulting benefit is that you can, in the fastest and most economical way possible, detect and resolve unknown error states that ever more frequently occur in your dynamic cloud application environment.

One window into health of every app, on every cloud



Alert

Visualize

Troubleshoot

Predict
Automate

Unified telemetry repository with AI and analytics

Business KPIs >
Applications >
App Services >
Databases >

Kubernetes >
Containers >
Service Meshes >

Cloud App PaaS >
Serverless >
IoT >

Public Clouds >
Private Clouds >
Multi-Clouds >

Servers >
Networks >
Storage >
SDDC >

First source of truth giving you answers, not data

Metrics

Histograms

Traces

Span Logs

Smart Alerting Features in Tanzu Observability.

Here is a summary of some of the smart alerting capabilities in Tanzu. For more details, refer to Tanzu Observability's public documentation page at https://docs.wavefront.com/alerts.html

Use Analytics to Set Alerts based on Behaviours

Set alert conditions based on any analytical expression with the Tanzu Observability query language, directly from a visualization or dashboard.

Set Alerts on Aggregate or Multiple Metrics

Set alert conditions based on different metrics types or aggregate metrics across an enitre system, e.g. p95 latencies as histograms across all your app servers.

Set Alerts on Missing Data

Know when your monitoring pipeline stops reporting data. Create alerts to detect missing telemetry data, so you can identify potential failures and resolve them before too much data is lost.

Set One or More Targets for Each Alert

Associate each alert with one or more custom alert targets, like PagerDuty, Moogsoft, ServiceNow, and automated remediation tools. Specify to include charts with forwarded alert notifications.

Backtest New Alerts

Fine tune new or existing alert conditions before you save them. display actual or hypothetical alert-generated events using backtesting.

Manage All Your Alerts

Easlily manage all the alerts from all your teams. View all alerts and their status. View alert history and details. Set role-based access controls on alerts. Snooze alerts during maintenance hours.

Wrap-Up and Key Takeaways

- Smarter alerts are simple alerts derived from smarter metrics-based anomalies.
- Smarter metrics take complex behaviors and transform them into values.
- The transformation is made possible through functional and Boolean combinations of a variety of analytics functions using a powerful, expressive language.

Smart alerts let you derive greater insight into how your systems are behaving. They can give you more advanced notice when something appears to be amiss.

By catching anomalies ahead of an outright failure, you can ensure an excellent experience for your customers.

Try Tanzu Observability for free:

tanzu.vmware.com/observability-trial

Join us online:

vmware®