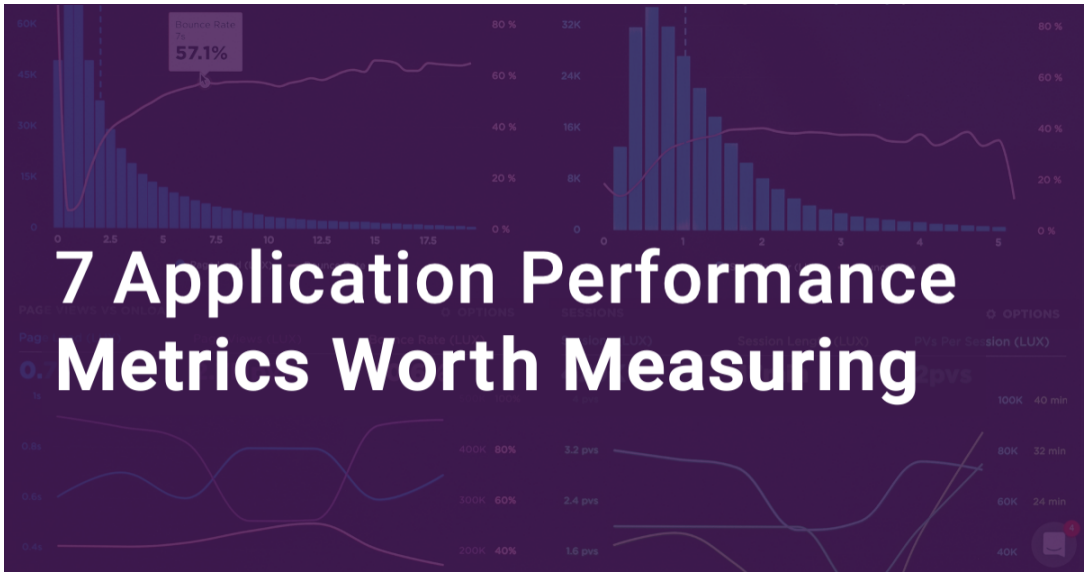
scoutapm.com

The 7 Application Performance Metrics You Need to Measure and Why

Jack Rothrock

9-12 minutes



When it comes to releasing new features out into the wild, developers and managers alike need to toe the line between speed and performance.

Take an all too common situation, where sales submits a high priority ticket to the development team that's a blocker for an enterprise deal coming through the pipeline. This blocker/feature requires a database query that is suspected to slow page speed for some customers, but the developer(s) isn't 100% certain when these instances occur or if the difference is more than negligible.

This is where a team can leverage an APM system to quicken development times without neglecting performance. With an APM

system, developers can deploy their code and watch for changes in performance. If a newly released feature is causing performance issues, the team can go back and address these as needed. Reducing premature optimization and increasing feature deployments.

Just installing an APM system is a major step forward for a development team. In minutes you and your team will start to see and evaluate both high and lower-level metrics. Once those metrics have been gleaned over, the real magic lies in understanding how to effectively use an APM system and what the metrics mean both independently and together.

Take the above example - after deploying the new feature for the sale, the authoring developer(s) saw a slight uptick in average response time but saw quite a significant increase in 95th percentile response time. What the developer feared has materialized. In some cases, the introduced SQL query (or queries) has significantly impacted a portion of customers.

Below, we are going to take a look at a few metrics, both independently and in conjunction with one another, which we believe can help you make the most of an APM system.

Here are the metrics that are most important to keep an eye on when measuring app performance:

- Throughput
- Average Response Time
- Queue Time
- 95th Percentile Response Time
- Apdex
- Errors
- Memory

How to Quantify Application Performance

As the great Jack Welch once put it, “if it isn’t measurable, it isn’t manageable.”

Development teams need insights into what their code is doing under the hood and how its impacting performance.

What the end-user interacts with is layers upon layers of features that have built up over time. It’s important for the development teams to see not only the high-level overview metrics of how their app is performing, but also being able to dig down and see which lines of code are causing slowness for either a single endpoint or background job.

Application Performance Monitoring Tools

Being able to generate quantifiable metrics for your app is as easy as registering with Scout, adding a few lines of code, and deploying.

In just a few minutes after you deploy, you will be able to visualize the data right before your eyes.

With our highly configurable charts and in-depth tracking and analysis, you and your team are one step closer to diagnosing your application’s bottlenecks.

The Most Important Application Performance Metrics

The ability to add an APM solution in just a few minutes enables developers to get insights into the application immediately.

The major question still hasn’t been answered though, how do we interpret the data?

Below we will discuss some of the metrics that Scout quantifies, as

well as how to read them both independently as well as in connection with one another.

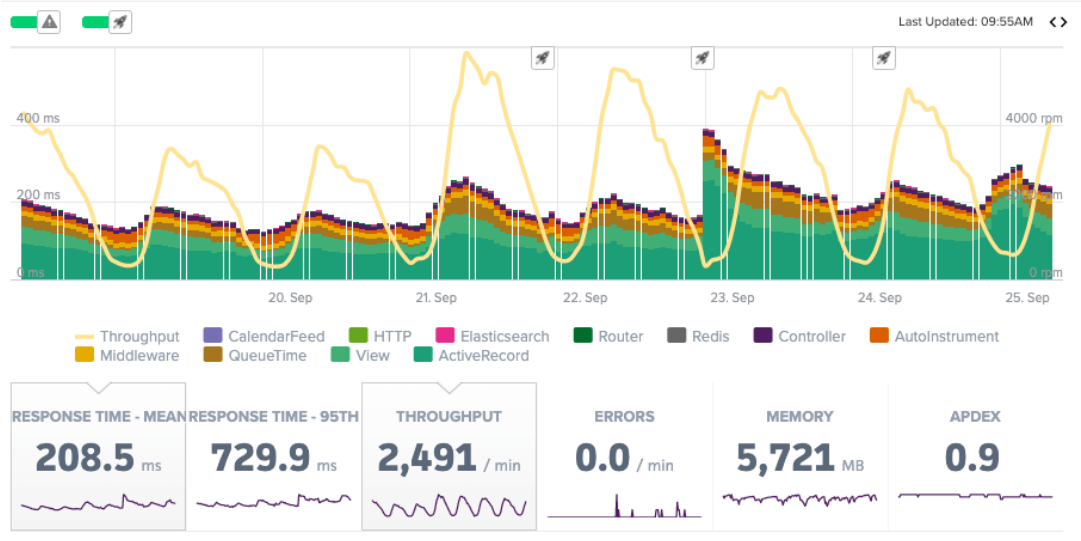
1. Throughput

At first glance this may be a metric that some would say is more of a KPI, but throughput is still the quintessential APM metric.

If I were to tell you (or your on-call team) that your average throughput was going to 20x at 3 A.M. this morning, you or your team would probably think twice about going to bed a little earlier than usual.

Being able to see what one's throughput usually is, and how much it varies, can help engineers gauge how many servers they need to support the traffic.

Additionally, throughput can help engineers assess how users will perceive their site. Throughput also plays a major role in calculating Little's Law. A very important, and not so often talked about, APM metric. We will discuss this a little more in the Queue Time section.



2. Average Response Time

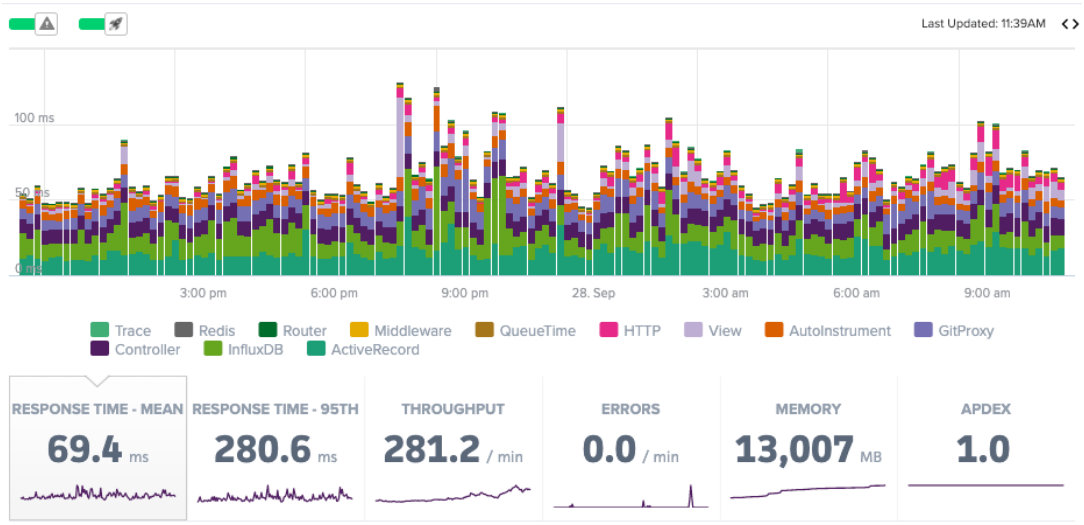
While maybe not the shiniest metric on the block, average response time is a cornerstone metric that helps gauge how one's application is performing as a whole.

This metric, combined with deploy tracking, can be extremely useful for developers.

Average response time can show whether a new deploy has increased average load times for all users.

This metric, combined with drilling down on specific endpoints, can help developers understand whether an area of code needs to be addressed immediately.

We certainly don't want /cart or /checkout average response times going up.



3. Queue Time

When we talk about Queue Time, we have to recognize that this is a measurable side effect of Little's Law.

Little's law is a queueing theory/law that can be used to understand potential bottlenecks in said queues.

There are a few queues that most developers need to be cognizant about when it comes to an application's performance: requests flowing from load balancers to web servers and background job

worker queues.

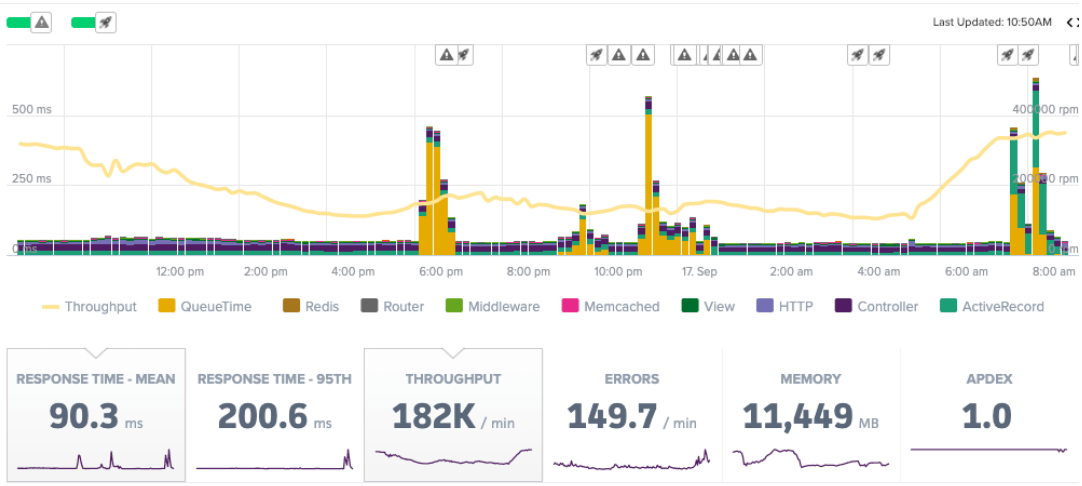
Little's Law is an algebraic expression that describes, in relation to web applications, the amount of transactions in a system is equal to the arrival rate (throughput) multiplied by the average time in the system (average response time).

Using this we can rework the equation to figure out what happens when either throughput goes up or response time goes up.

With load balancers, seeing an increase in queue time is an indication that the servers are running out of resources.

Since queue time is just a measurable effect of Little's Law, we can use Little's Law to cure queue time.

One way is to reduce average response time via increasing performance or doing less work per request, the other is to have a higher throughput tolerance via adding more servers.



4. 95% Response Time

Average response time is great for looking at things from a high-level perspective.

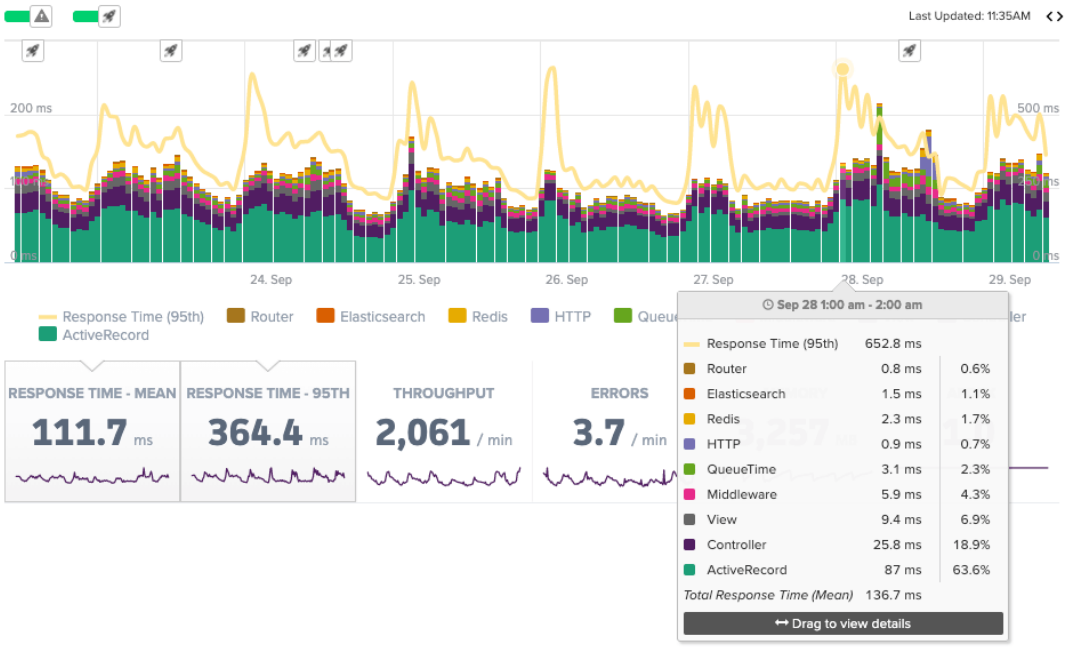
The only issue with averages, in general, is that they don't do a great job capturing what happens to outliers.

The average response rate could drop a little for most customers, increase for outliers, and it looks like nothing has happened.

That's where the 95th percentile response time shines.

When looking out past two sigmas, we can find endpoints that look normal from a quick glance, but underneath trouble brews for some of our users.

With the example we talked about above, this is the metric (and potentially Apdex) that would spot the issue.



5. Apdex

When people talk about APM metrics, people usually refer to Apdex. So what is Apdex?

Apdex is a measurement of a user's general level of satisfaction when using an application.

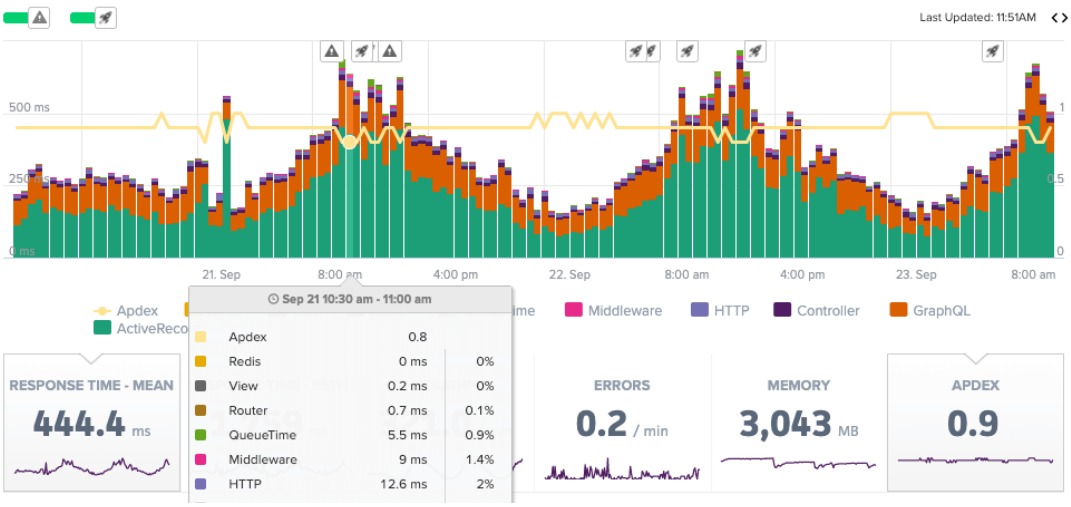
The value of the Apdex metric can range between 0 and 1, with an Apdex score of 1 indicating that all of our customers are fully satisfied, whereas an Apdex score of 0 tells us that none of our customers are happy.

This is done by categorizing requests into three buckets -- satisfied,

tolerating, and frustrated. If everyone is satisfied then one's Apdex score will be 1, and on the other hand, if everyone is frustrated, then one's Apdex score will be 0.

This can be a useful metric to look at when deploying new features. For example, if part of the system takes a long time to respond after a feature update, then the user naturally starts to become frustrated.

What constitutes a request being marked as frustrated depends on the app, and is a threshold that can be configured in Scout APM. By default, we set the threshold at 500ms with frustrated requests being 4x this threshold.

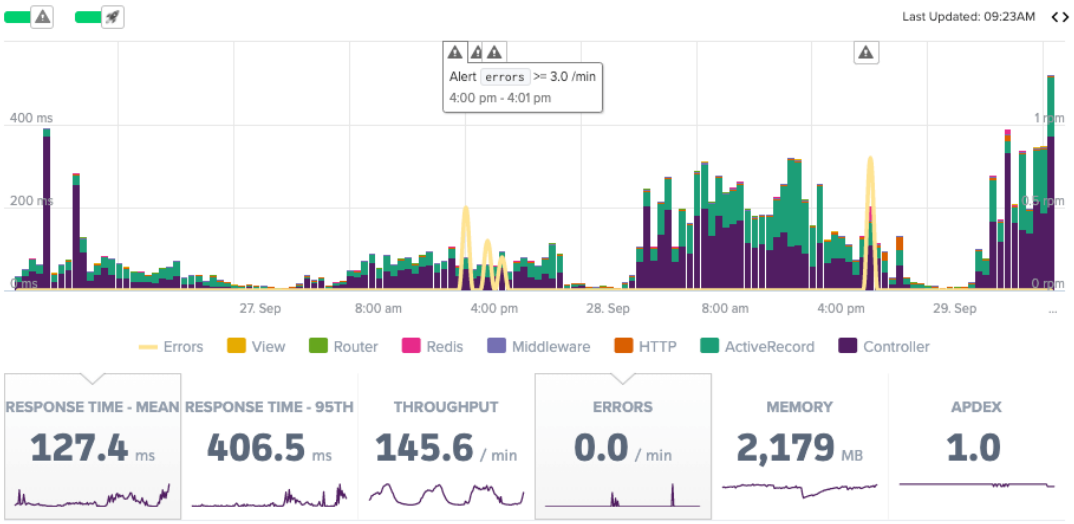


6. Errors

Whether it be 404s, 408s, or 500s, no user ever wants to encounter an error.

Being able to track errors and pinning down how the errors occurred is a powerful tool to have in one's APM system.

Having the ability to receive notifications right when errors occur, as well as seeing the number of times the error has occurred that day, is powerful in knowing whether this is an issue that needs to be addressed immediately, and/or if these errors are part of something larger.



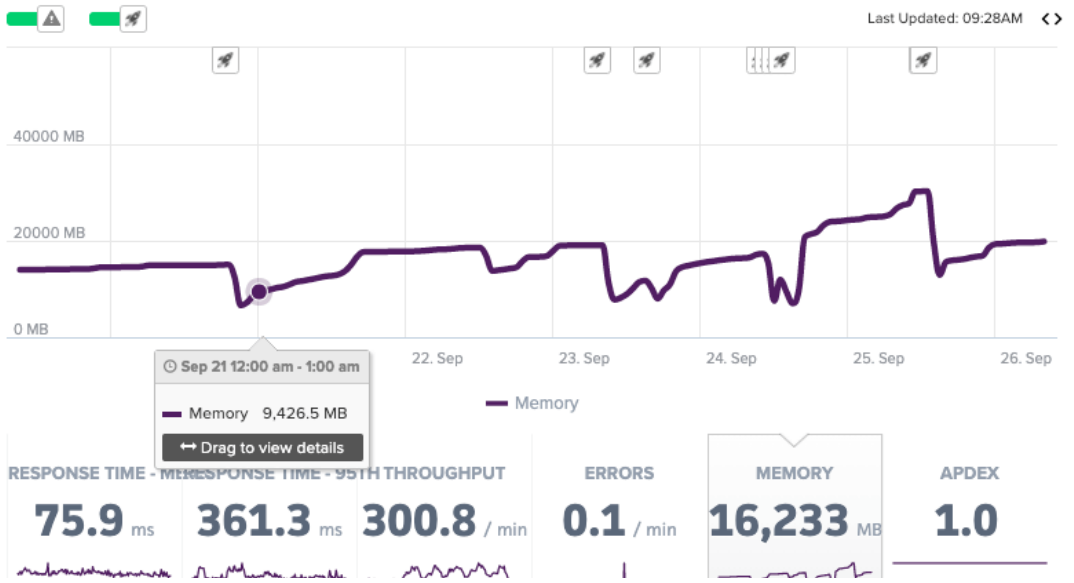
7. Memory

Different languages allocate memory differently, and purchasing high memory servers is expensive.

Knowing how and when your language both allocates and cleans up memory is essential to scaling - from both a technical and financial perspective.

Ruby, for example, likes to hang on to allocated memory as it expects further use of this memory to be used shortly.

This can be an issue depending on if one's application is seeing a massive throughput increase in different areas.



Summary

The old saying “a stitch in time saves nine” is an important adage that is super applicable for performance monitoring.

Scout understands that development teams are always pressed for time, and sometimes these performance issues are time-consuming to investigate.

Yet being able to spot bottlenecks as they are released is a lot easier to deal with than when multiple bottlenecks have started to pile upon one another.

We give development teams bandwidth back by being able to gain fast insights into how their code is performing from both a micro and macro perspective.

Additionally, Scout enables teams to spot fires on the horizon before they lead to high priority incidents.

Are you interested in bringing your application monitoring to a new level? Sign up for a free trial with Scout today, and see why Scout is trusted by 1000s of engineers at leading-edge companies.